

# MDE for publishing data on the Semantic Web.

Guillaume Hillairet, Frédéric Bertrand, Jean Yves Lafaye

{guillaume.hillairet01, fbertran, jylafaye}@univ-lr.fr

Laboratoire L3I,

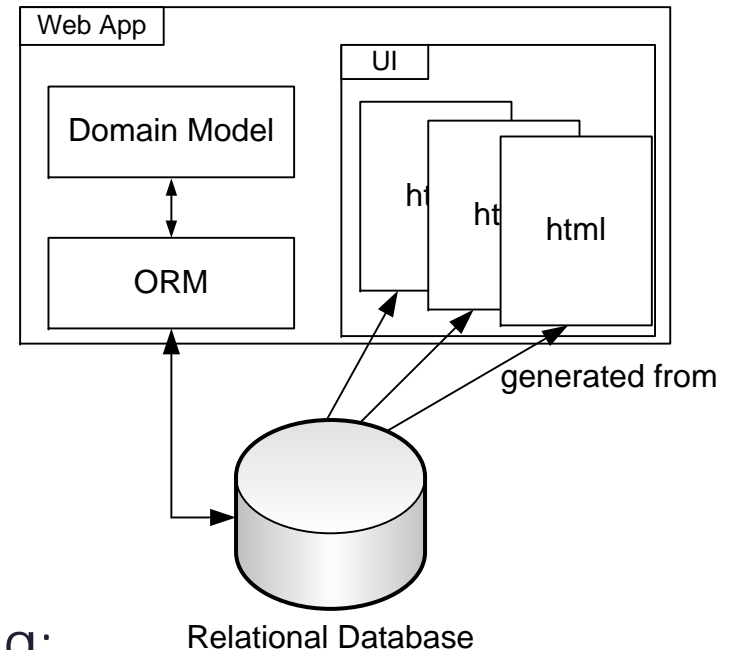
Université de La Rochelle, France.

# Outline

- Context of this work
  - Web applications.
  - Semantic Web applications.
  - Publishing Semantic Web data.
- Contribution
  - Overall approach.
  - Object Ontology Mapping.
  - Data transformation.
- Perspectives

# Database Driven Web applications

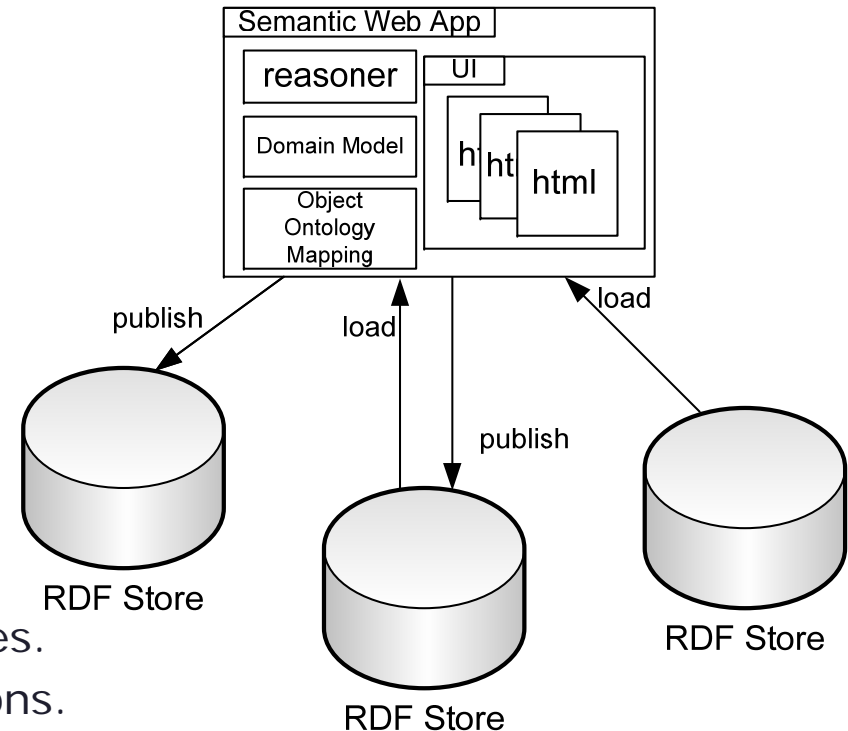
- Web applications:
  - Data are stored in relational databases.
  - Html pages are generated from database content.
  - Web applications are driven by an object-oriented domain model (better cases).



- Need for an object-relational mapping:
  - *impedance mismatch* between object and relational.
  - Mapping specification between object-oriented domain model and relational schema.
  - Several tools already here:
    - Hibernate, TopLink, ActiveRecord, ADO.NET, etc...

# Semantic Web applications

- Semantic Web:
  - Data => RDF.
  - Schema => Ontology (OWL).
  - Query Language => SPARQL.
  - Reasoning capabilities.
- Semantic Web application:
  - Provides data in RDF format.
  - Loads RDF from various RDF databases.
  - Shares data with others SW applications.
- SW application development with Object-oriented languages:
  - Uses an object-oriented domain model.
  - Eases evolution of existing applications.
  - Needs for object ontology mapping solutions.



# Publishing data as RDF

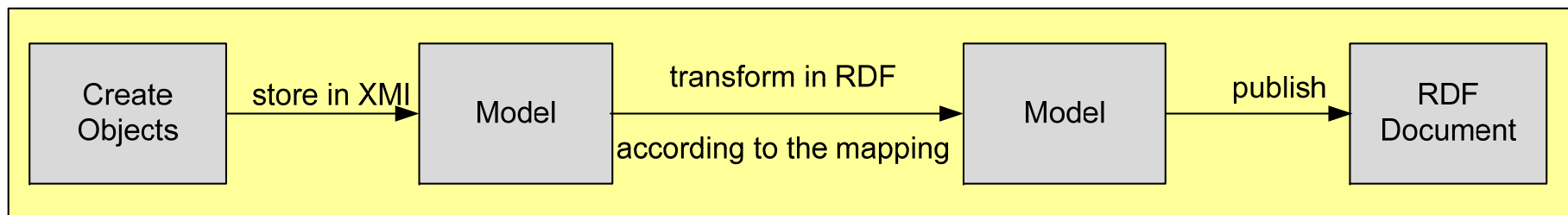
- Relational to RDF mapping solutions have been proposed:
  - D2RQ, R2O, Virtuoso, etc...
- Relational schemas:
  - Poor semantic (flat model).
  - Hard to apprehend (and so to map).
- Why not using object domain model (data model)
  - Object relational mapping solutions exist.
  - Object model has more semantics.
  - More closer to an ontology than a relational schema.

# Publishing objects as RDF

- Use the object model as pivot between persistence layer and ontology.
- Publishing relational data as RDF => 2 steps mapping:
  - Use existing object relational mapping solutions.
  - Use object ontology mapping.
- The work presented here focus on:
  - Object model and ontology mapping.
  - Object to RDF transformation (Publish objects as RDF).

# Basic Scenario

- Prerequisite:
  - object-oriented domain model.
  - Users vocabulary (ontologies).
- Goal:
  - publish objects according to users vocabulary.
- Approach:
  - Map object domain model and ontologies.
  - Generate transformation between objects and RDF.



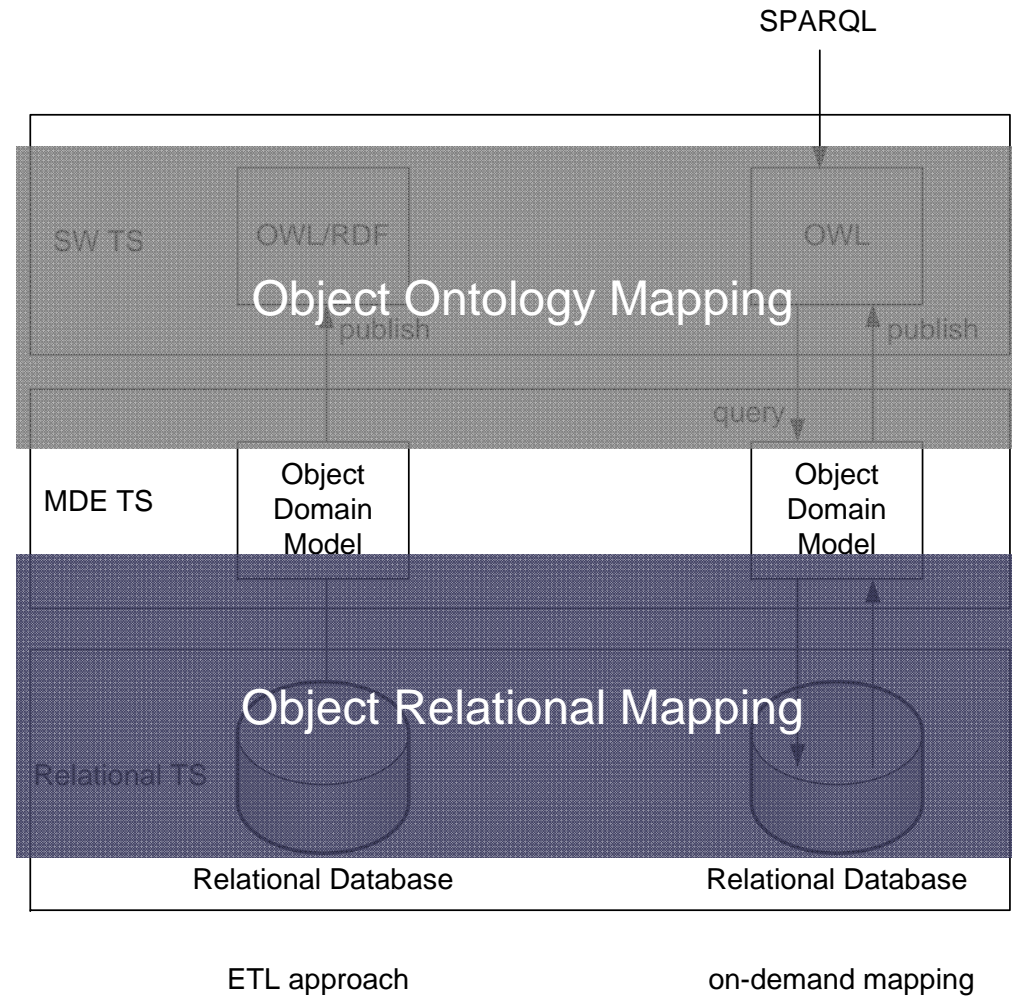
# Outline

- Context of this work
  - Web applications.
  - Semantic Web applications.
  - Publishing Semantic Web data.
- Contribution
  - Proposal.
  - Object Ontology Mapping.
  - Data transformation.
- Perspectives

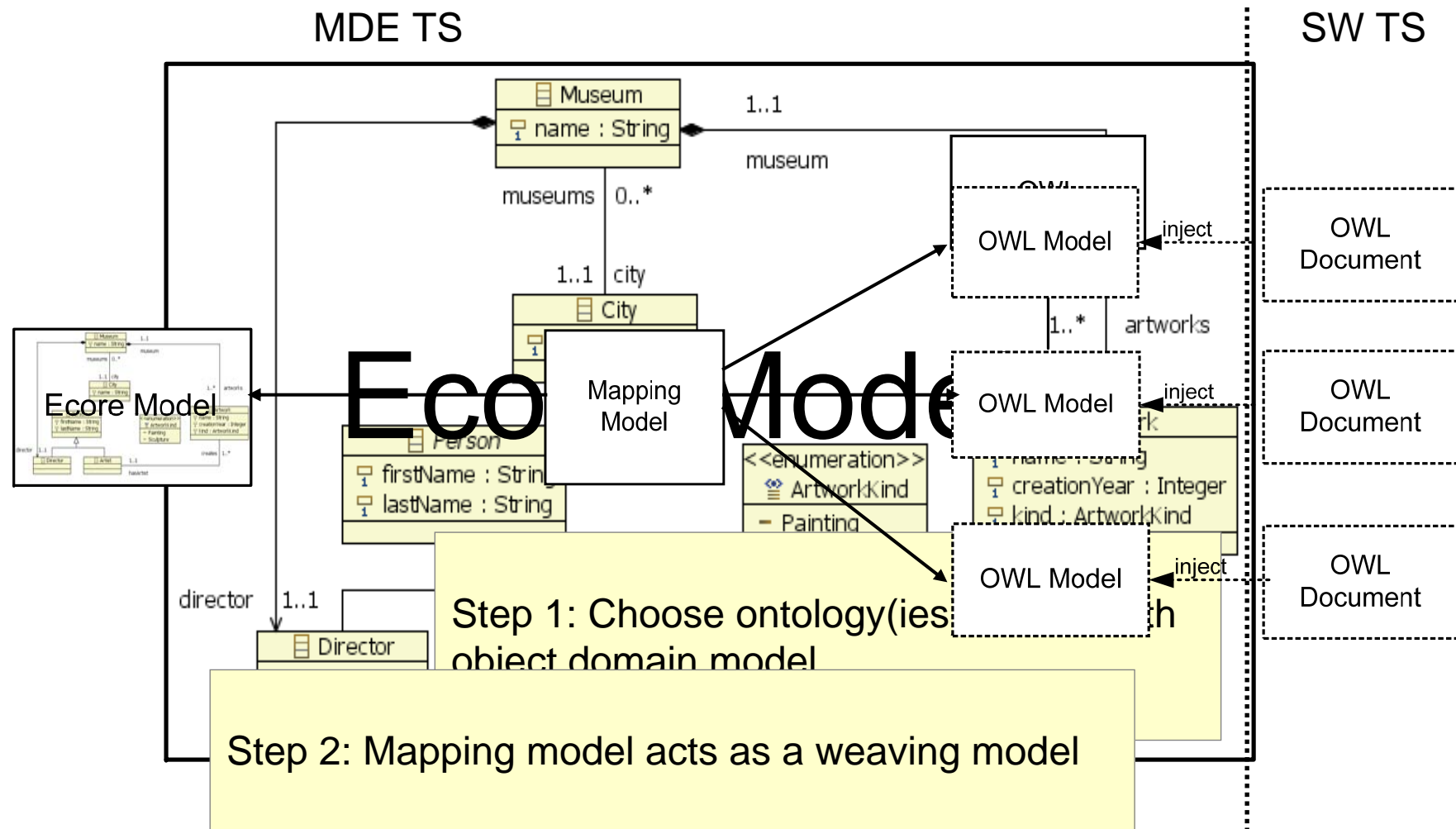
# Proposal

- Object model is a central issue between the persistence layer and the semantic layer.
- Two levels mapping:
  - Links between EMF Model and OWL Ontology
    - → weaving model.
  - Convert objects as RDF resources
    - → model transformation.

# Proposal



# Object Ontology mapping principle

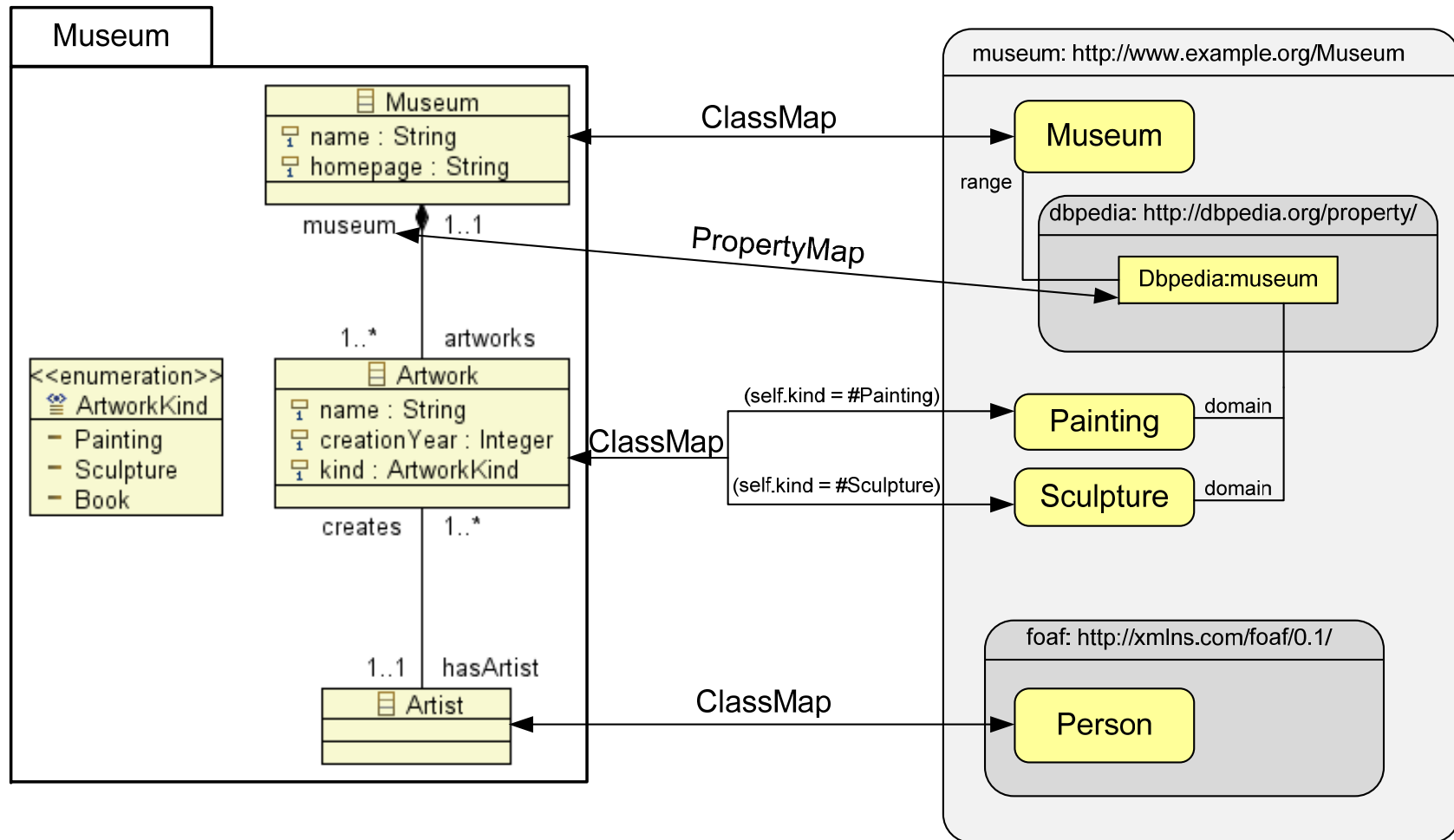


# Object Ontology mapping

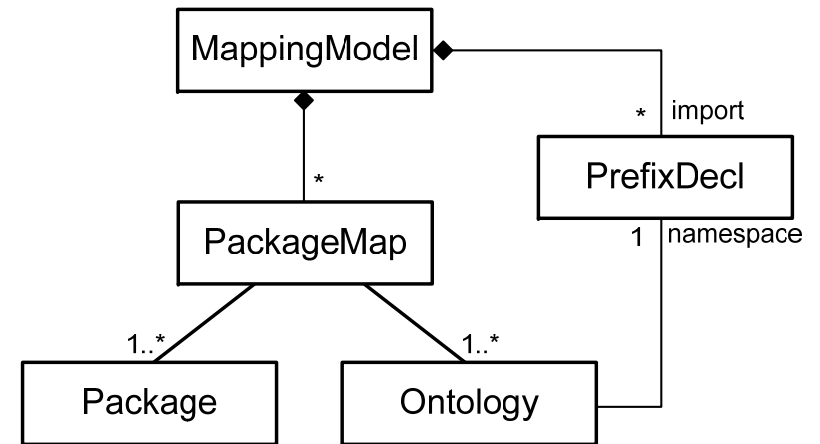
## Overview

- Declarative language for mapping EMF model concepts to OWL ontology concepts.
  - Mapping links are explicit declared.
  - Mapping language supports a subset of OCL for EMF model navigation and mapping constraints definition.
- Weaving EMF models and OWL models:
  - A mapping is a model (conform to a metamodel)
  - A mapping model holds semantic links between EMF models and OWL models.
- Weaving eases generation of the model transformation for translating objects to RDF.

# Mapping Overview



- MappingModel composed of:
  - PackageMap
    - EPackage  $\Leftrightarrow$  OWLOntology.
  - Prefix declaration
    - Ontology namespace.
    - declare ontologies to import.



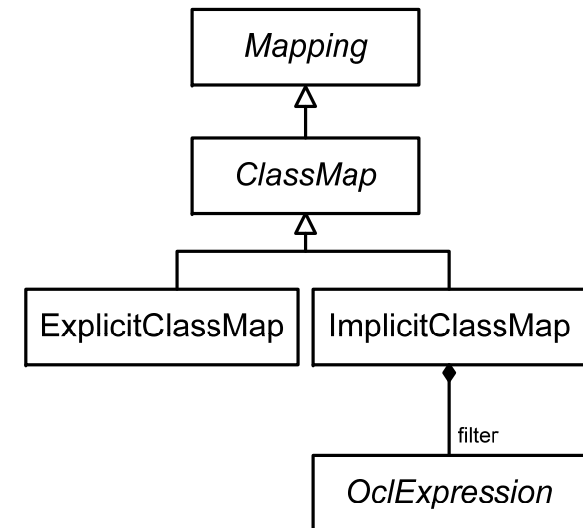
```
1  prefix dbpedia: "http://dbpedia.org/property/";
2  map package Museum with museum:"http://www.example.org/museum#" def = {
3    map class Museum with museum:Museum def = {
4      uriPattern = "http://www.example.org/museum/" + self.name;
5      properties = {
6        map attr name with museum:name;
7        map ref artworks with museum:artworks;
10     }
11 }
```



- ClassMap
  - Explicit Class Map => one to one.
  - Implicit Class Map => one to many.

```
1  map class Artwork (self.kind = #Painting) with museum:Painting def = {  
2    uriPattern = "http://www.example.org/museum/painting/" + self.name;  
3    properties = {  
4      map attr name with dbpedia:title;  
5      map attr creationYear with museum:creationYear;  
6      map ref museum with dbpedia:museum;  
7      map ref hasArtist with dbpedia:artist;  
8    }  
9  }
```

```
10 map class Artwork (self.kind = #Sculpture) with museum:Sculpture def = {  
11   uriPattern = "http://www.example.org/museum/sculpture/" + self.name;  
12   properties = { ... }
```



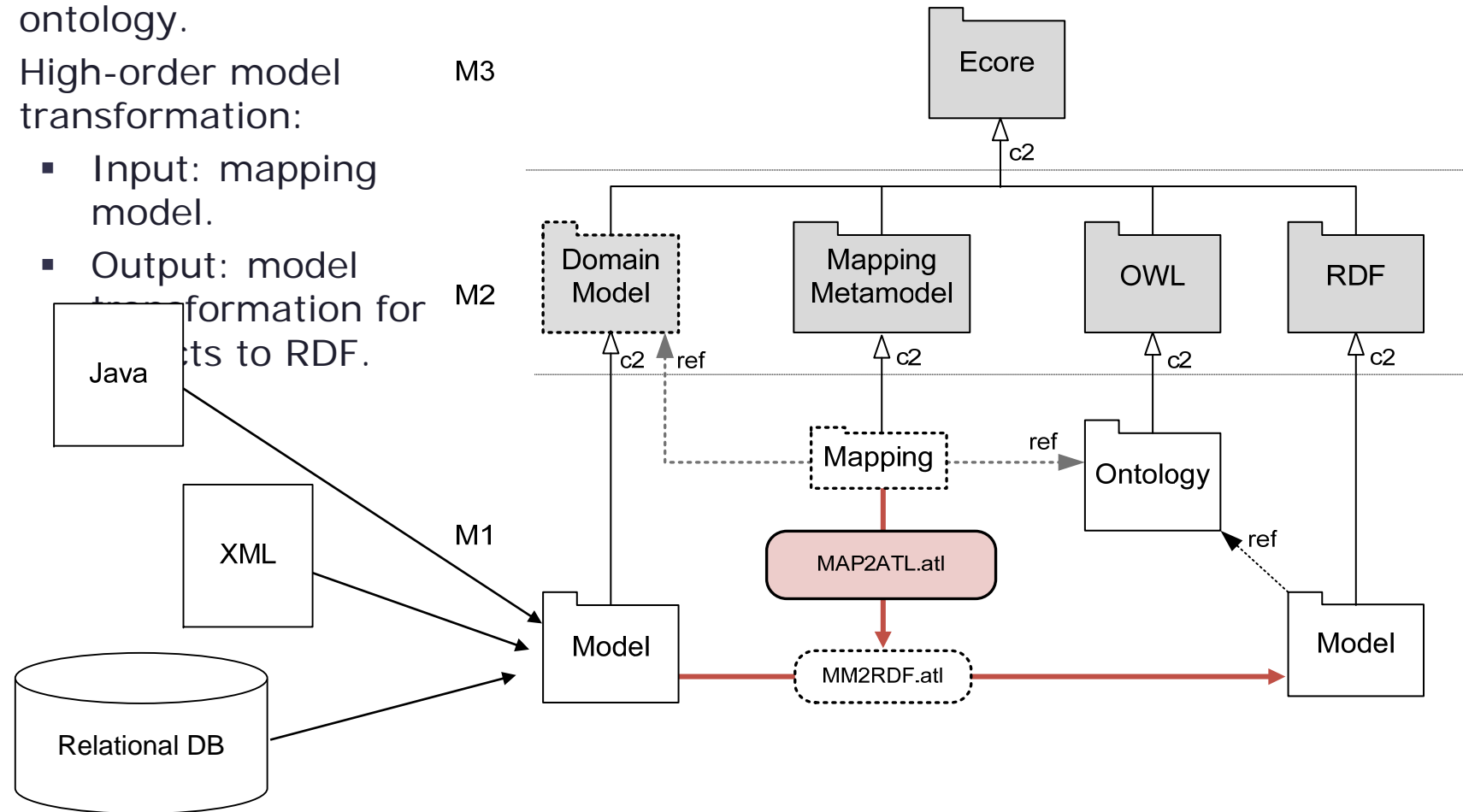
# Data Transformation (From Object to RDF)

## Solutions:

- Extract, transform and publish:
  - Obtain model (objects representation)
  - Generate Model Transformation according to the mapping.
  - Transform all objects into RDF resources.
  - Publish resulting RDF.
- On-demand mapping:
  - Define SPARQL queries over ontology.
  - Rewrite SPARQL into HQL according to the mapping.
  - Execute HQL query over object domain model.
  - Transform resulting objects into RDF Resources.

# Extract, transform and publish RDF

- Mapping weaves EMF domain model and OWL ontology.
- High-order model transformation:
  - Input: mapping model.
  - Output: model transformation for objects to RDF.



# ATL rule for Artwork mapping

```
<museums name="Le Louvre" city="//@cities.0">
  <director firstName="Henri" lastName="Loyrette"/>
  <artworks name="Mona Lisa"
    creationYear="1642"
    exhibitedIn="//@museums.0" hasArtist="//@persons.0"/>
  <artworks name="David"
    creationYear="1504"
    kind="Sculpture"
    exhibitedIn="//@museums.0/@rooms.0" hasArtist="//@persons.1"/>
</museums>
```

```
uri <- m.getURI(),
```

```
subject <- m.getSubject(),
```

**Model2RDF.atl**

```
thisModule.makeDataStatement(l, l.name, 'title'),
```

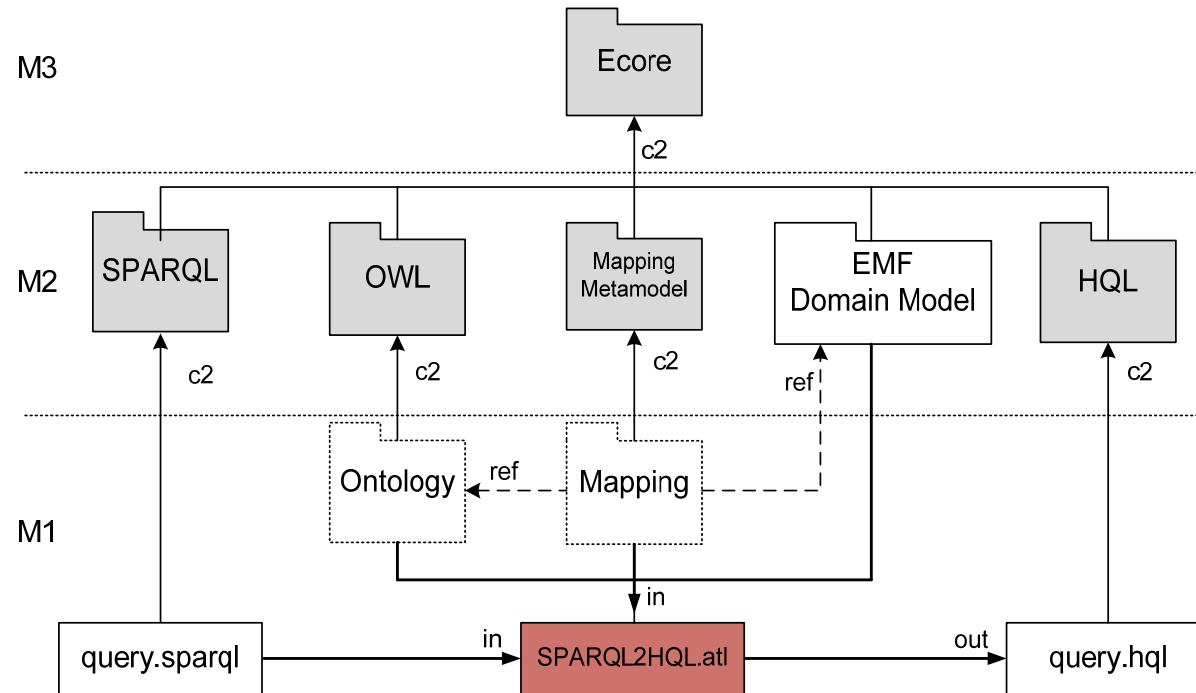
```
thisModule.makeDataStatement(l, l.creationYear, 'creationYear'),
```

```
thisModule.makeObjectStatement(l, l.museum, 'museum'),
```

```
<rdf:Description rdf:about = 'http://www.example.org/museum/LeLouvre'>
  <rdf:type rdf:resource = 'http://www.example.org/museum#Museum' />
  <museum:name rdf:datatype = '&xsd:string'>Le Louvre</museum:name>
  <museum:artworks rdf:resource = 'http://www.example.org/museum/painting/MonaLisa' />
  <museum:artworks rdf:resource = 'http://www.example.org/museum/sculpture/David' />
</rdf:Description>
<rdf:Description rdf:about = 'http://www.example.org/museum/painting/MonaLisa'>
  <rdf:type rdf:resource = 'http://www.example.org/museum#Painting' />
  <dbpedia:title rdf:datatype = '&xsd:string'>Mona Lisa</dbpedia:title>
  <museum:creationYear rdf:datatype = '&xsd:int'>1642</museum:creationYear>
  <dbpedia:museum rdf:resource = 'http://www.example.org/museum/LeLouvre' />
  <dbpedia:artist rdf:resource = 'http://www.example.org/museum/person/LeonardoDaVinci' />
</rdf:Description>
```

# On-demand mapping

- Translate SPARQL queries into object-oriented query language (HQL).
- Execute HQL query on domain model.



# Conclusions

- Publishing objects as RDF Resources is an important issue for Semantic Web Application development.
- Using MDE helps us to reduce the complexity of implementing this approach.
- Generating transformations from the mapping model insures a generic approach.

# Perspectives

- The mapping language does not respond to all the impedance mismatch between object – RDF.
- Generation of bidirectional model transformation from mapping is possible.

Thank you,

Questions ?