# Comparing M2T & M2M Complementary Approaches

Hugo Bruneliere, Richard Paige & Goran Olsen

INRIA, York & SINTEF

# Context of this work

- The present courseware has been elaborated in the context of the MODELPLEX European IST FP6 project (http://www.modelplex.org/).
- Co-funded by the European Commission, the MODELPLEX project involves 21 partners from 8 different countries.
- MODELPLEX aims at defining and developing a coherent infrastructure specifically for the application of MDE to the development and subsequent management of complex systems within a variety of industrial domains.
- To achieve the goal of large-scale adoption of MDE, MODELPLEX promotes the idea of a collaborative development of courseware dedicated to this domain.
- The MDE courseware provided here with the status of open-source software is produced under the EPL 1.0 license.

# Outline

- Presentation of model transformation
  - Overview
  - Other kinds of transformation (not model-based)
- What is M2T?
  - Principles
  - Existing solutions (MOFScript & Epsilon EGL)
- What is M2M?
  - Principles
  - Existing solutions (ATL & Epsilon ETL)
- Differences between M2T & M2M
- Combining both approaches in an MDE process
  - Application on a concrete use case: UML2 to Java
  - Advantages of such a solution

# Presentation of model transformation

## Overview

- Model-Driven Engineering (MDE) technique
- Consume/produce models as inputs/outputs
  - Each model conforms to a given metamodel
- Two kinds of model transformation:
  - Model-to-Text transformation (M2T)
  - Model-to-Model transformation (M2M)
- Two different possible implementations:
  - Use a model transformation Domain-Specific Language (DSL)
    - ATL, MOFScript, Epsilon, etc.
  - Use a General Purpose Language (GPL)
    - Java, C#, etc.

# Presentation of model transformation

## Other kinds of transformations (not model-based)

- XSLT transformation
  - XML document-to-XML document transformation
  - Each XML document conforms to a given XML schema

  → Directly translatable to the MDE paradigm
- Compilation transformation
  - Text-to-Binary transformation
  - Each source program conforms to a given grammar
  - Each target compiled program conforms to a given binary format

  → Also adaptable to the MDE paradigm
- Model transformation is a generic abstraction of all these techniques

# What is M2T?

## Principles

- To be completed (York & SINTEF)

# What is M2T?

## Existing solutions: MOFScript

- To be completed (SINTEF)

# What is M2T?

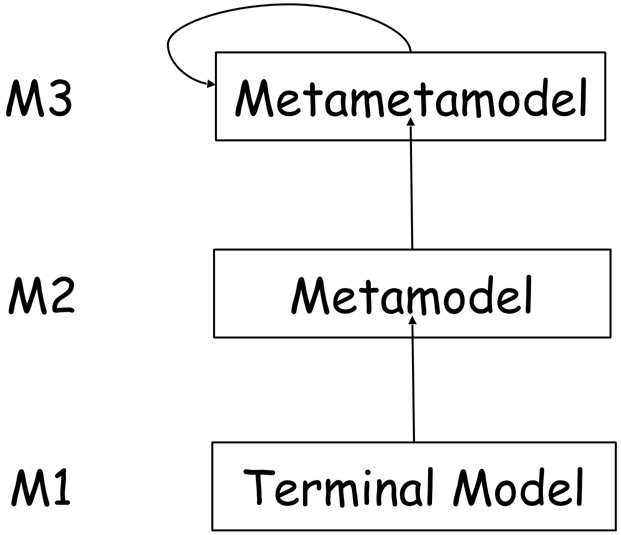
## Existing solutions: Epsilon EGL

- To be completed (York)

# What is M2M?

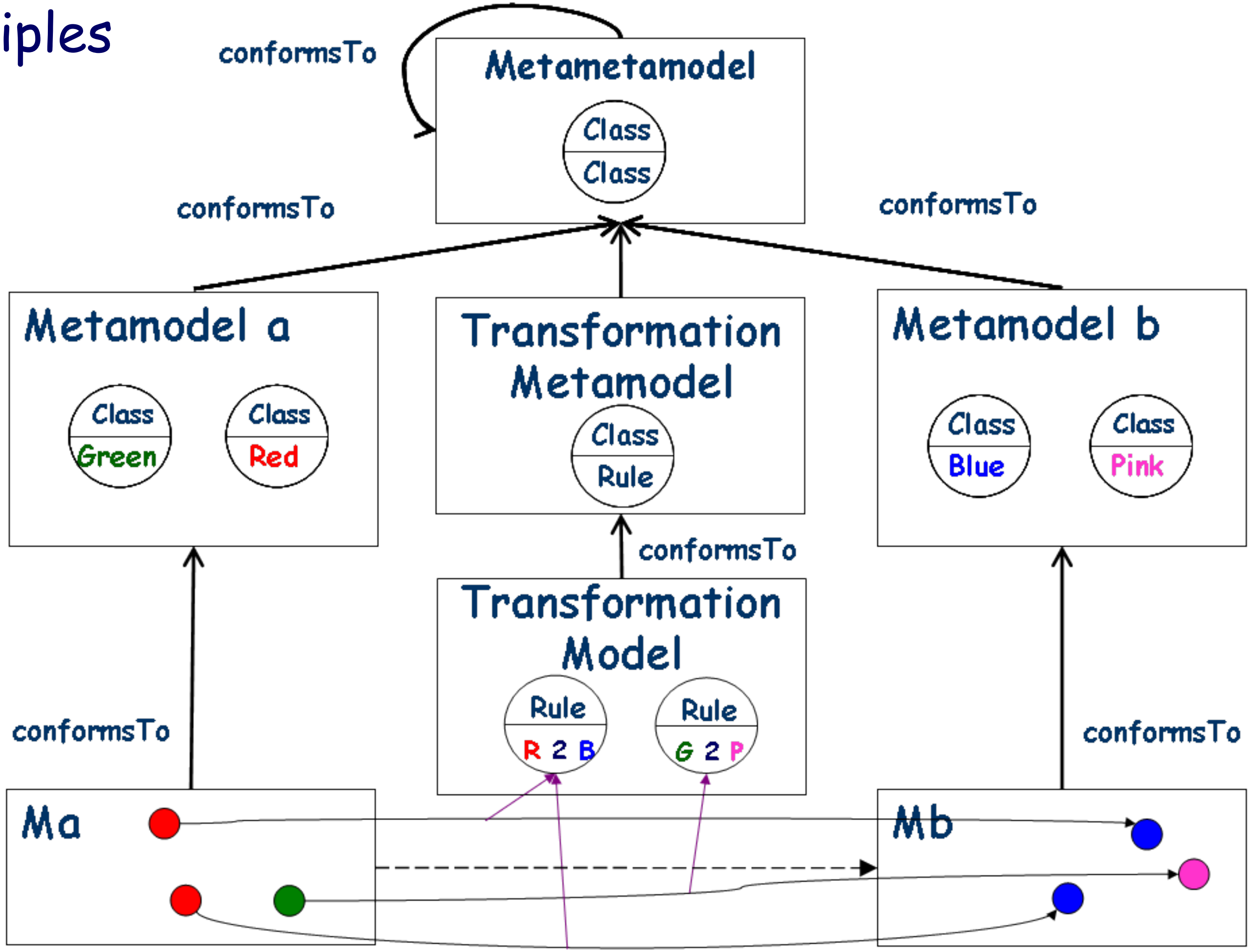
## Principles

- A M2M transformation is the automated creation of *m* target models from *n* source models
  - Each model conforms to a given reference model (i.e., a metamodel or metametamodel), which can be the same for several models
- M2M transformation is not only about M1 to M1 transformations:
  - M1 to M2: promotion
    - E.g., UML to MOF
  - M2 to M1: demotion
    - E.g., MOF to Metrics
  - M2 to M2
    - E.g., metamodel refactoring
  - etc.



# What is M2M?

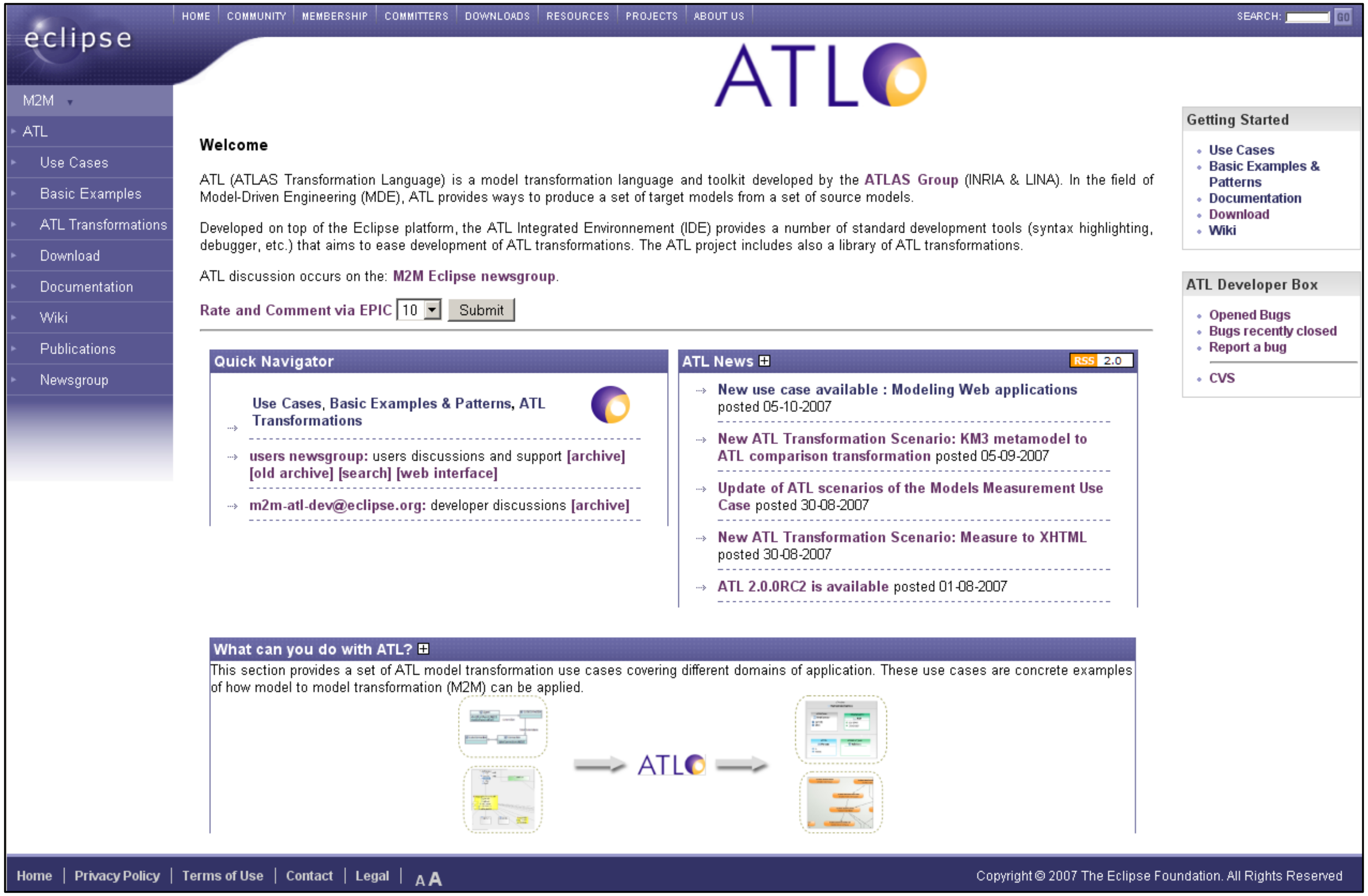
## Principles



# What is M2M?

## Existing solutions: Eclipse-M2M ATL

- Website → http://www.eclipse.org/m2m/atl/



**Welcome**

ATL (ATLAS Transformation Language) is a model transformation language and toolkit developed by the ATLAS Group (INRIA & LINA). In the field of Model-Driven Engineering (MDE), ATL provides ways to produce a set of target models from a set of source models.

Developed on top of the Eclipse platform, the ATL Integrated Environnement (IDE) provides a number of standard development tools (syntax highlighting, debugger, etc.) that aims to ease development of ATL transformations. The ATL project includes also a library of ATL transformations.

ATL discussion occurs on the: M2M Eclipse newsgroup.

Rate and Comment via EPIC 10 Submit

**Quick Navigator**

- Use Cases, Basic Examples & Patterns, ATL Transformations
- users newsgroup: users discussions and support [archive] [old archive] [search] [web interface]
- m2m-atl-dev@eclipse.org: developer discussions [archive]

**ATL News**

- New use case available : Modeling Web applications posted 05-10-2007
- New ATL Transformation Scenario: KM3 metamodel to ATL comparison transformation posted 05-09-2007
- Update of ATL scenarios of the Models Measurement Use Case posted 30-08-2007
- New ATL Transformation Scenario: Measure to XHTML posted 30-08-2007
- ATL 2.0.0RC2 is available posted 01-08-2007

**What can you do with ATL?**

This section provides a set of ATL model transformation use cases covering different domains of application. These use cases are concrete examples of how model to model transformation (M2M) can be applied.

Copyright © 2007 The Eclipse Foundation. All Rights Reserved

# What is M2M?

## Existing solutions: Eclipse-M2M ATL

- Available resources (1/2)
  - Use cases → 24 complete transformation scenarios covering many different domains of application
  - Basic examples → very first transformation examples which are interesting when starting with ATL (for beginners)
  - ATL Transformations → ATL Transformation Zoo which gathers more than a hundred of various and varied transformations implemented using ATL
  - Download → different binary builds of ATL available and also additional information for using the ATL update site

# What is M2M?

## Existing solutions: Eclipse-M2M ATL

- Available resources (2/2)
  - Documentation → various kinds of ATL documents including a reference manual, a user manual, installation instructions, etc
  - Publications → non-exhaustive list of papers presenting different works involving or using (directly or indirectly) ATL
  - Wiki → an open section dedicated to ATL on the Eclipse Wiki which allows the community to consult or/and add information about ATL
  - Newsgroup → a link to the Eclipse newsgroup dedicated to the M2M project components (posts concerning ATL are prefixed with the [ATL] tag)

# What is M2M?

## Existing solutions: Eclipse-M2M ATL

- How to get the plugins:
  - Download the latest binary builds (frequently updated): http://www.eclipse.org/modeling/m2m/downloads/?project=atl
  - Use the M2M update site (M2M ATL SDK): http://www.eclipse.org/modeling/m2m/updates/
  - Install ATL sources from CVS (stable HEAD): http://wiki.eclipse.org/ATL/How_Install_ATL_From_CVS/
  - Install ATL sources from CVS (development branch): http://wiki.eclipse.org/ATL/How_Install_ATL_(Dev)_From_CVS

# What is M2M?

## Existing solutions: Epsilon ETL

- To be completed (York)

# Differences between M2T & M2M

- M2T transformation bridges the MDE technical space with the Grammarware technical space
  - Consumes/produces models to/from text files
  - Requires both reference models (i.e., metamodels or metametamodels) and text formats (e.g., grammars)
  - Handles both model elements and text

  → Heterogeneity

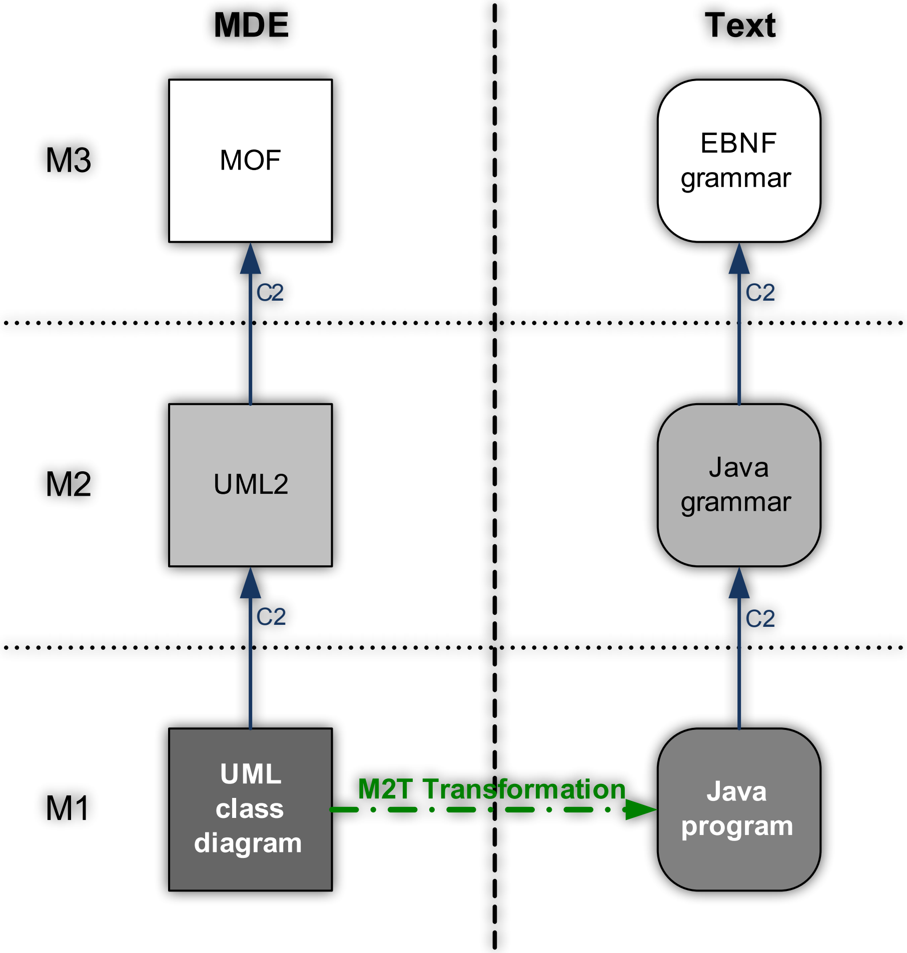
- M2M transformation concerns only the MDE technical space
  - Consumes/produces only models
  - Requires only reference models (i.e., metamodels or metametamodels)
  - Handles only model elements

  → Homogeneity

# Combining both approaches in an MDE process

## Application on a concrete use case: UML2 to Java

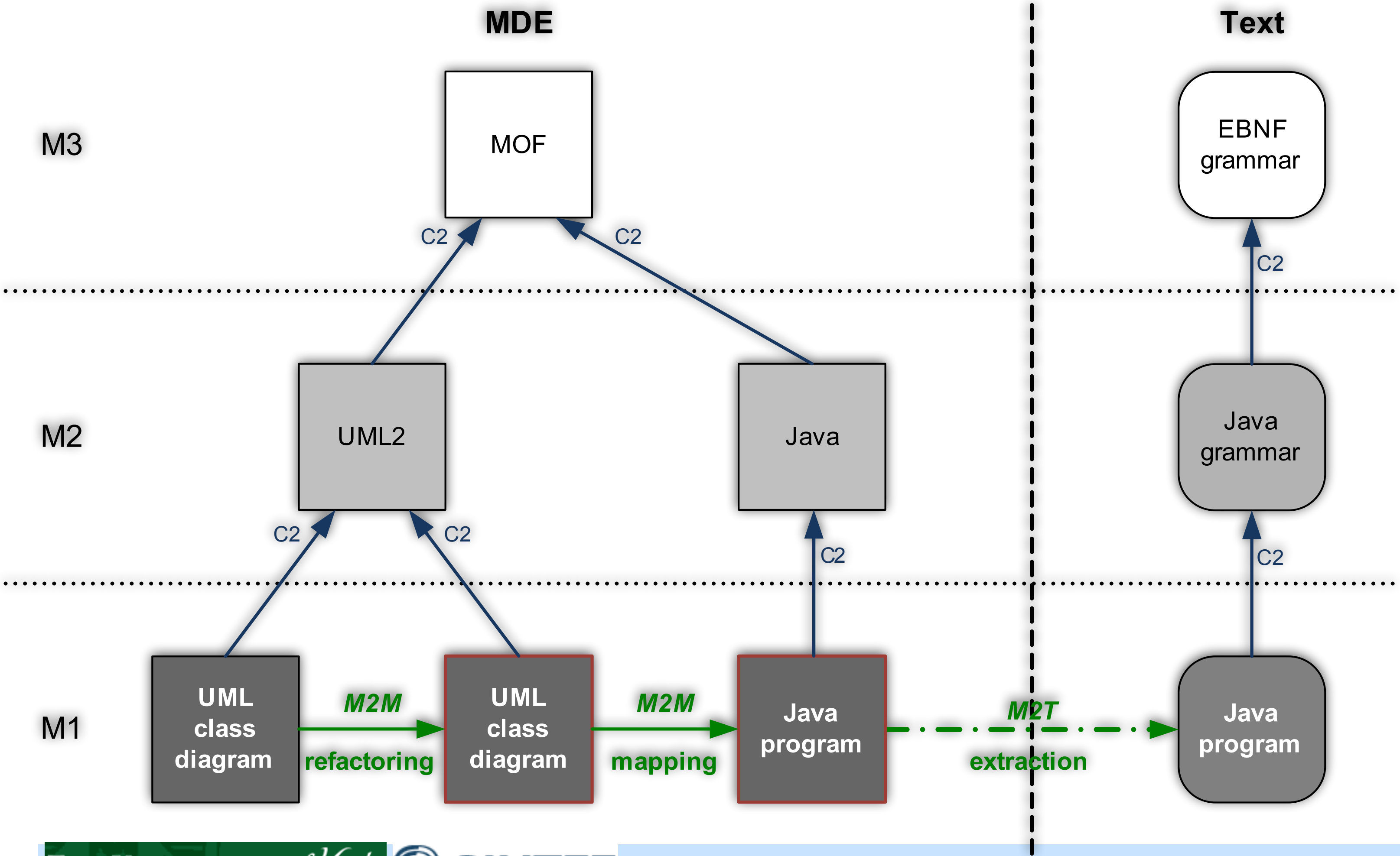
- An M2T solution
- A single transformation performing at the same time:
  - Refactoring (e.g. delete of multiple inheritance)
  - Mapping (UML2 concepts to Java concepts)
  - Extraction to a concrete syntax (conforming to the Java grammar)



# Combining both approaches in an MDE process

## Application on a concrete use case: UML2 to Java

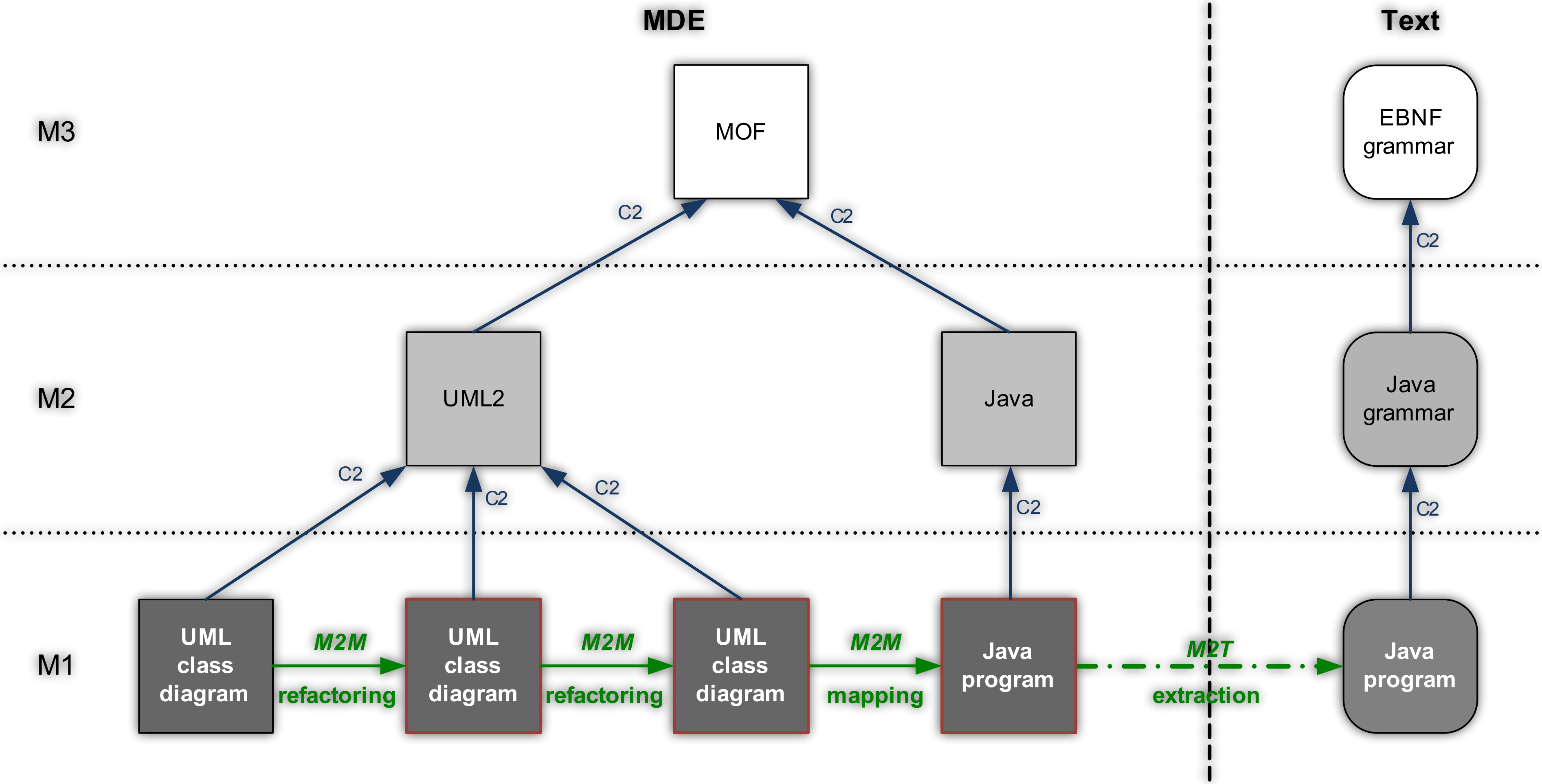
- Same case using an M2M+M2T solution



# Combining both approaches in an MDE process

## Application on a concrete use case: UML2 to Java

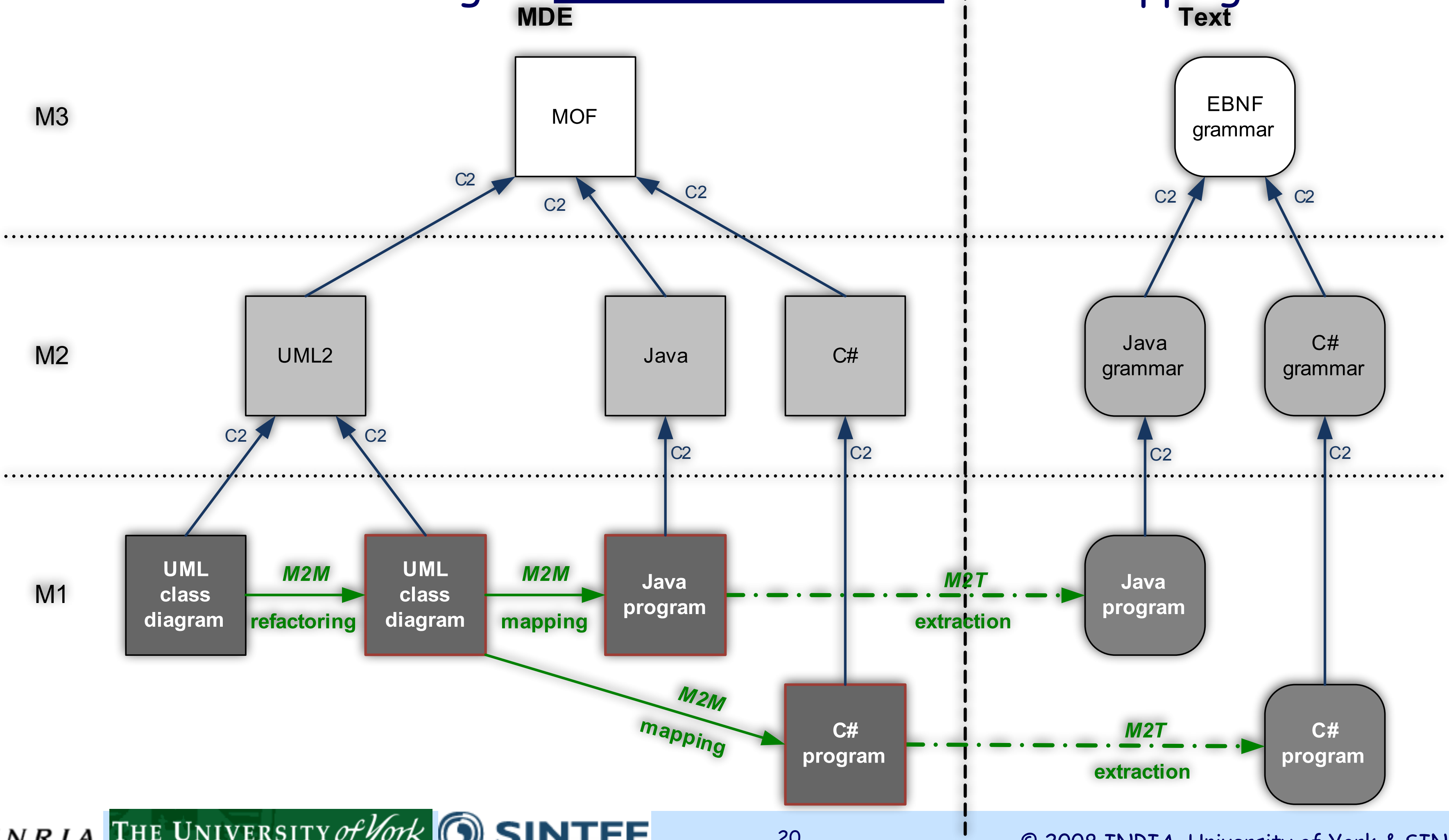
- Same case using an M2M+M2T solution + new refactoring



# Combining both approaches in an MDE process

## Application on a concrete use case: UML2 to Java

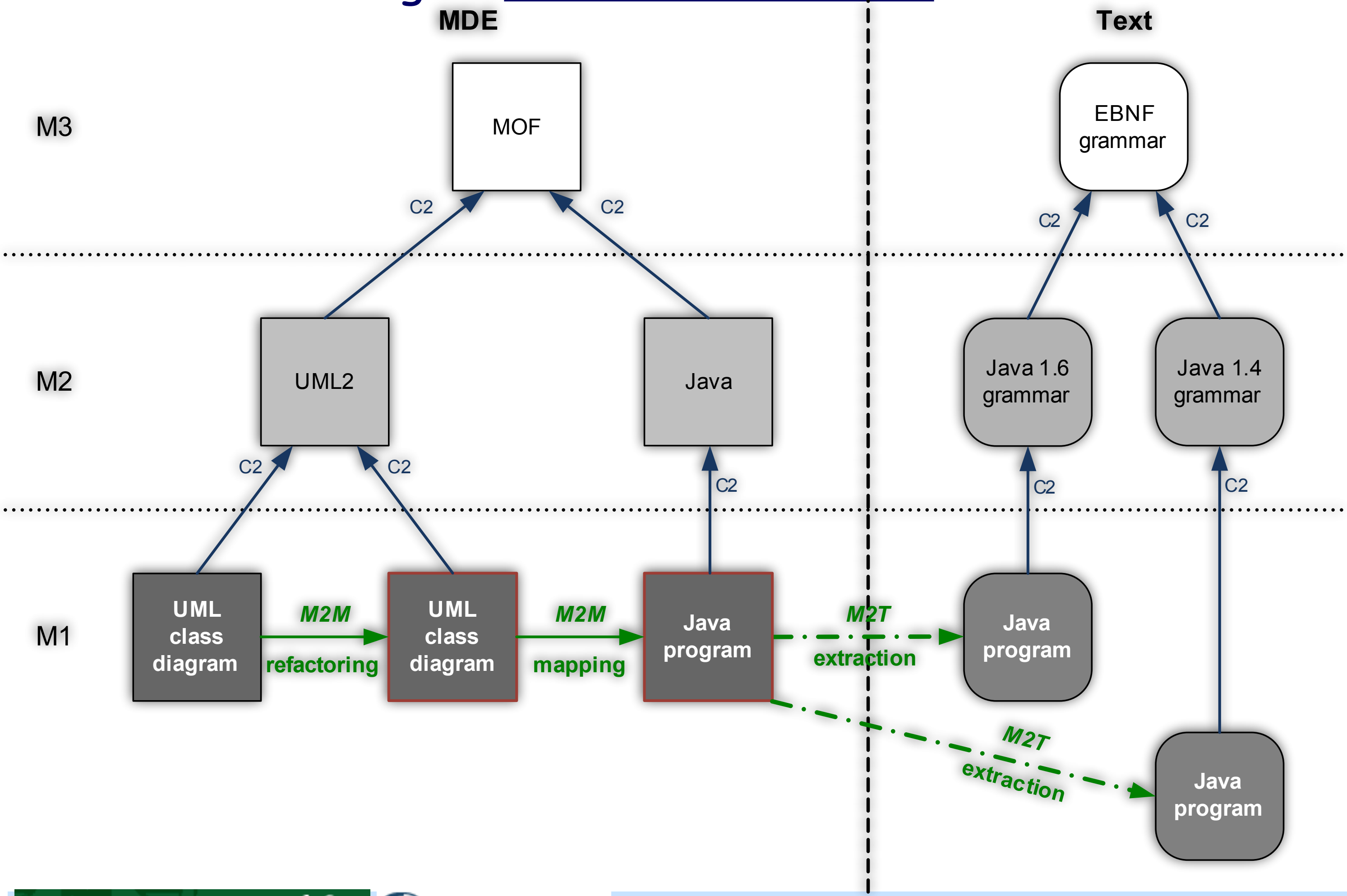
- Same case using an M2M+M2T solution + new mapping



# Combining both approaches in an MDE process

## Application on a concrete use case: UML2 to Java

- Same case using an M2M+M2T solution + new extraction



# Combining both approaches in an MDE process

## Advantages of such a generic M2M+M2T solution

- **Modularity**
  - Clearly separate the concerns (refactoring, mapping, extraction to a given syntax, etc)
- **Extensibility**
  - Easily add new features (additional refactoring, different mapping, other extraction to a textual or graphical syntax, etc)
- **Reusability**
  - Apply the same feature in different contexts (i.e., the same refactoring for targeting different languages)
- **Homogeneity**
  - Handle mostly models (extraction is just the final step)
- **Abstraction**
  - Focus is set only on the concepts (abstract syntax) and not on their various possible representations (concrete syntaxes)