ATL: A Model Transformation Tool

Frédéric Jouault, Freddy Allilaire, Jean Bézivin, Ivan Kurtev

Presented by Bairong Lei

Mar 12, 2012

Outline

- Problem
- ATL introduction with modeling
- Case study with ATL
- ATL Development Tools
- Evaluation

Problem

- In MDE developing model transformation definitions is a common task.
- Model transformation needs to ensure that models are consistent in a precise way that developers can define.
- Mature MDE tools are required to support this operation.

Introduction to ATL with MDE

- ATLAS Transformation Language (ATL) is a DSL for specifying model-to-model transformation
- ATL is a hybrid language providing a mix of declarative and imperative constructs.
- It builds on the Object Constraint Language formalism.

Introduction to ATL with MDE

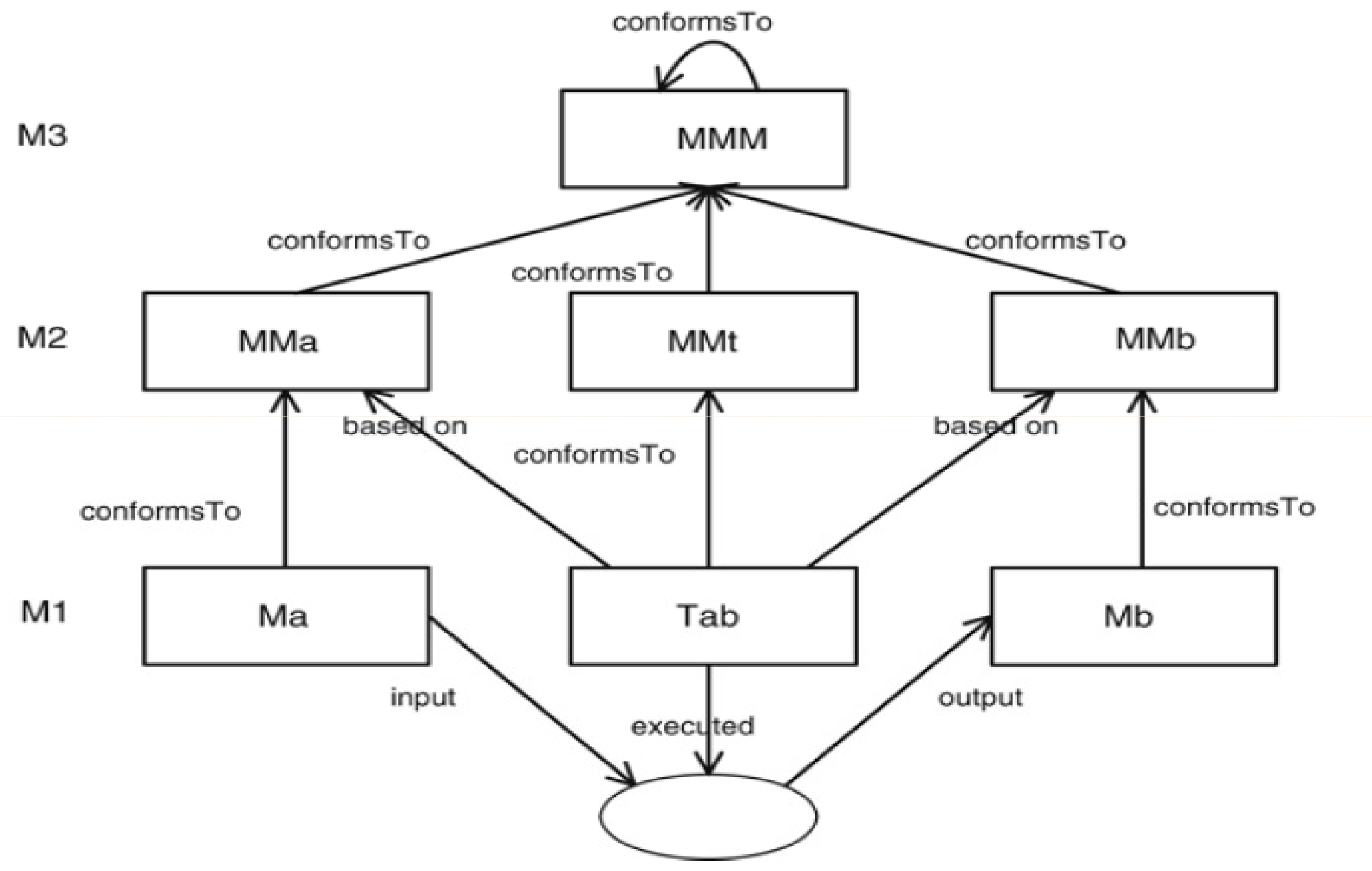


Fig. 1. Model transformation pattern.

Introduction to ATL with MDE

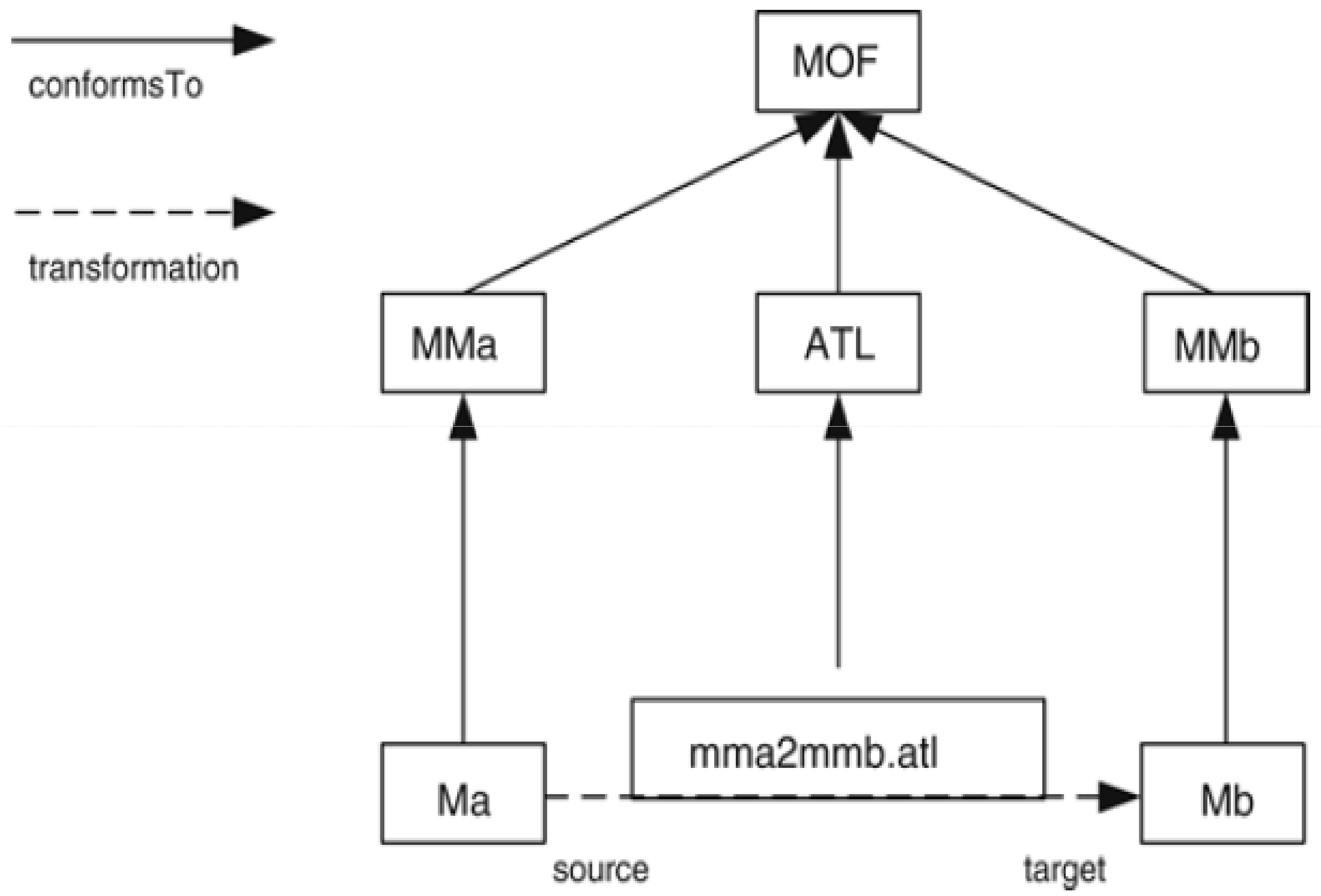


Fig. 2. Overview of the ATL transformational approach.

Introduction to ATL with MDE

- ATL transformations
  - Unidirectional
  - Operating on read-only source models
  - Producing write-only target models
  - Source and target models expressed in XMI OMG serialization format.
  - Meta-models expressed in XMI or KM3 notation.

Introduction to ATL with MDE

- ATL language contains declarative and imperative constructs.
- Transformation definitions
  - Header section
  - Import section
  - Helpers
  - Transformation rules

Case study: Class to relational

Class Meta-model

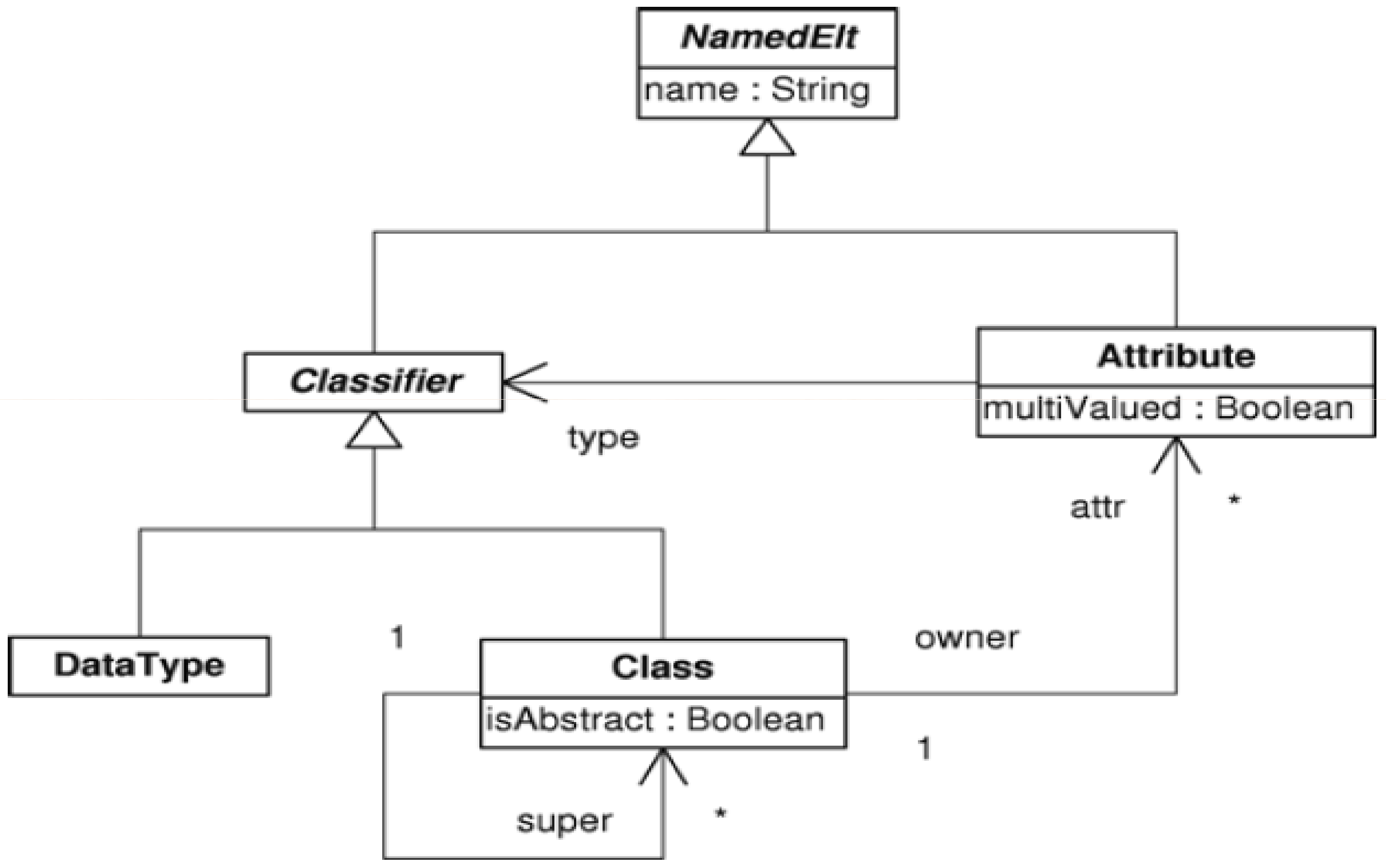


Fig. 3. Class metamodel.

Class Model

- Have zero or more attributes
- May specialize other classes
- Type of attributes is either a primitive or a class

Relational Meta-model

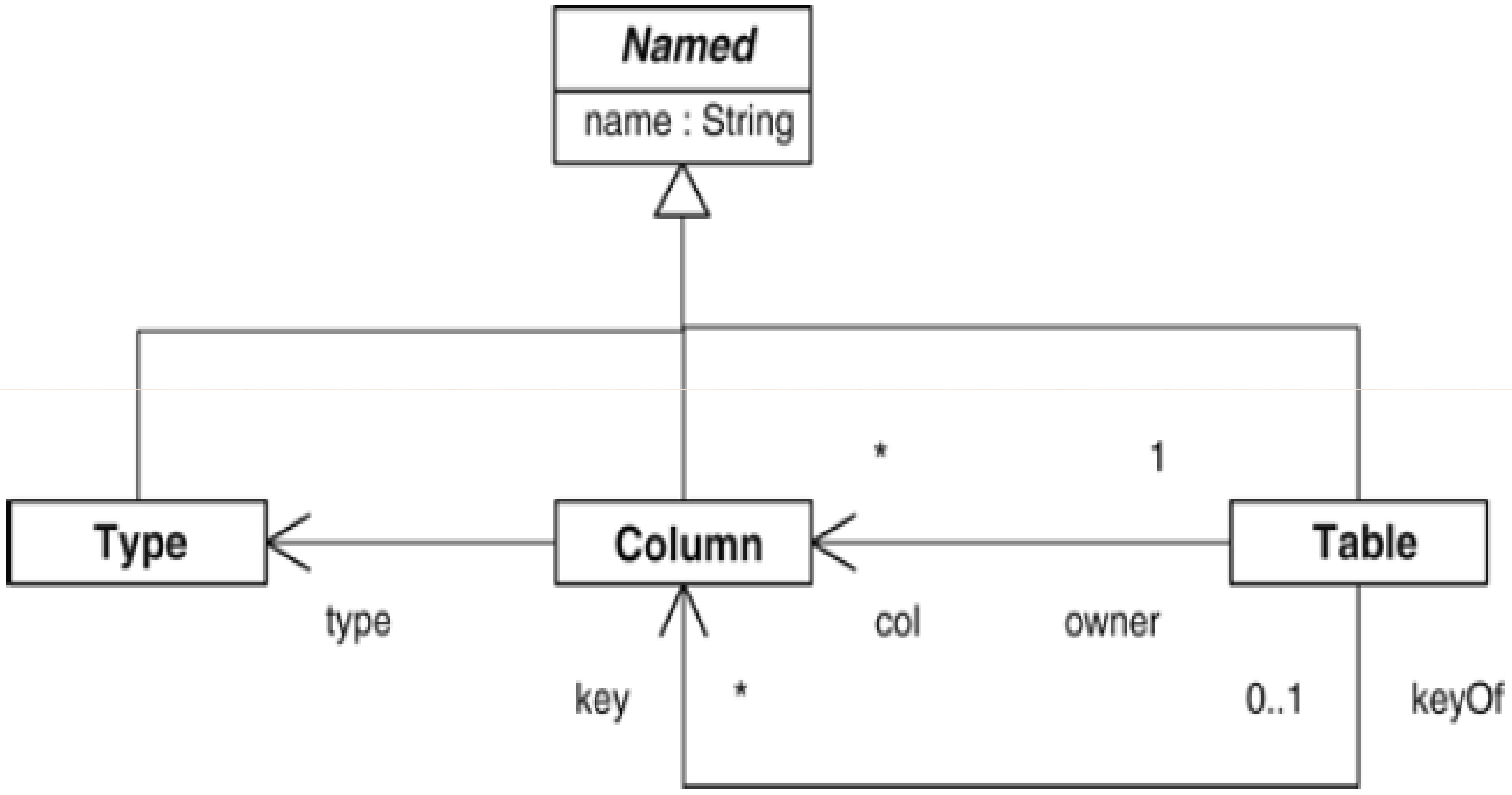


Fig. 4. Relational metamodel.

Relational Model

- Contains a set of tables
- Every table has zero or more columns
- Some columns are keys

Transformation Rules

- ATL rules may be specified either in a declarative style or imperative style
- Matched rules (known as declarative rules)
  - Source pattern
    - A set of source types
    - A guard
  - Target pattern
    - A set of elements
      - A target type
      - A set of bindings

Examples of Matched Rule

```
1. rule Class2Table {
2.   from
3.     c : Class!Class                   // source pattern
4.   to
5.     out : Relational !Table (                       // target pattern
6.             name <- c.name,
7.             col <- Sequence {key}->union(c.attr->select(e | not e.multiValued)),
8.             key <- Set {key}
9.     ),
10.
11.     key : Relational!Column (
12.             name <- ’objectId’,
13.             type <- thisModule.objectIdType
14.     )
15. }
```

Line 7 indicates that obtaining all the columns derived from non multi-valued attributes and uniting them with the *key* column created in the same rule.

Rule that creates columns from non multi-valued attributes

```
1. rule ClassAttribute2Column {
2.         from
3.             a : Class!Attribute (
4.                     a.type.oclIsKindOf(Class!Class) and not a.multiValued
5.             )
6.         to
7.             foreignKey : Relational !Column (
8.                 name <- a.name + ’Id’,
9.                 type <- thisModule.objectIdType
10.        )
11. }
```

Line 4 has a guard expression to ensure only non multi-valued attributes will be selected for transformation by this rule.

ATL Development Tools

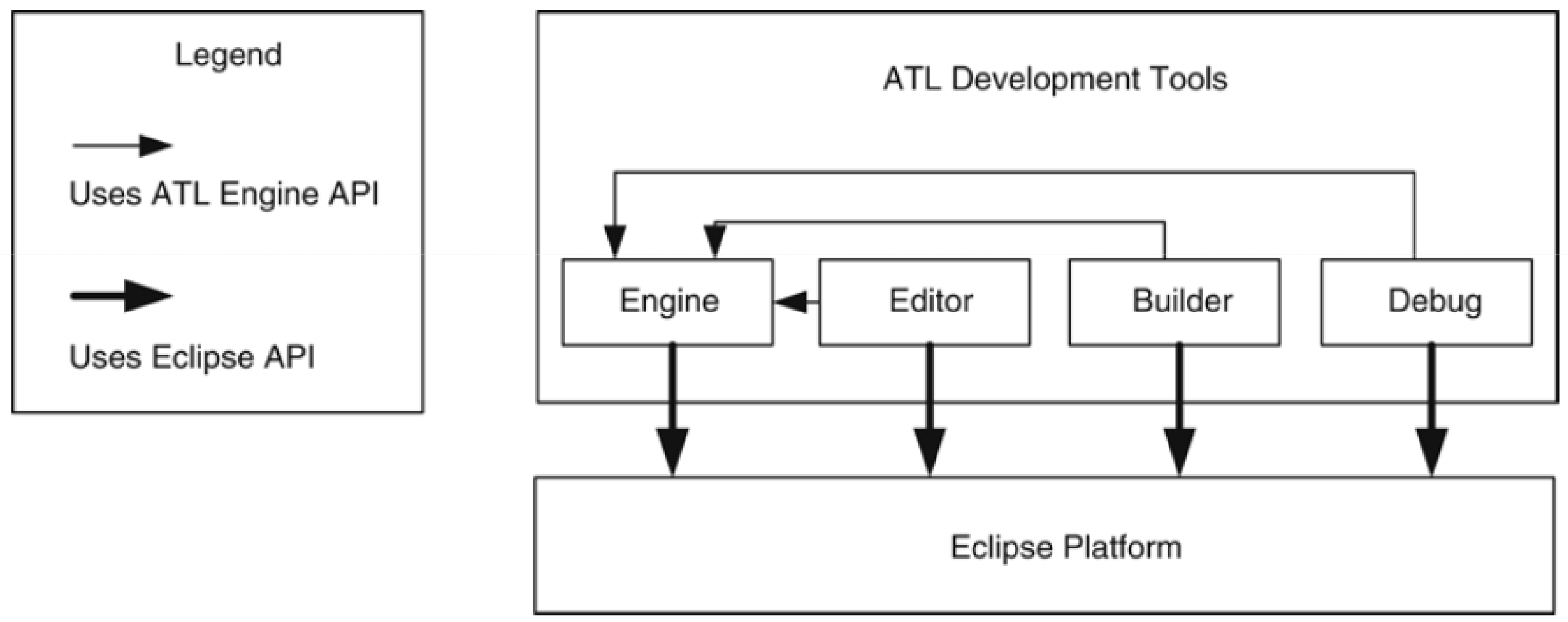


Fig. 5. ATL development tools architecture.

Engine

- Responsible for compilation and execution
- Transformations are compiled to byte-code programs run by ATL Virtual Machine.
- Model Handler Abstraction Layer translates the instructions of the VM for model manipulation to the instructions of a specific model handler such as EMF and MDR.

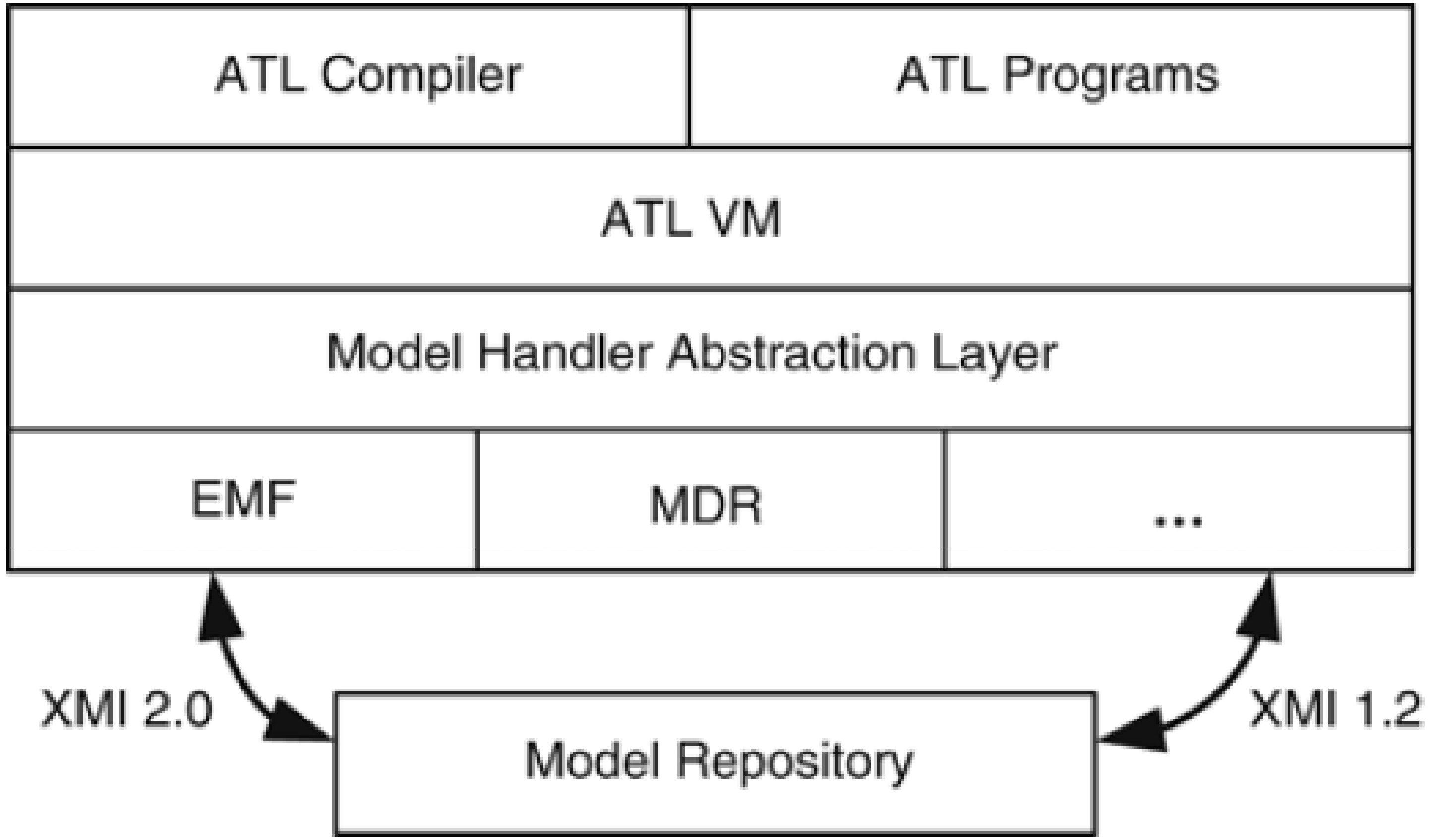
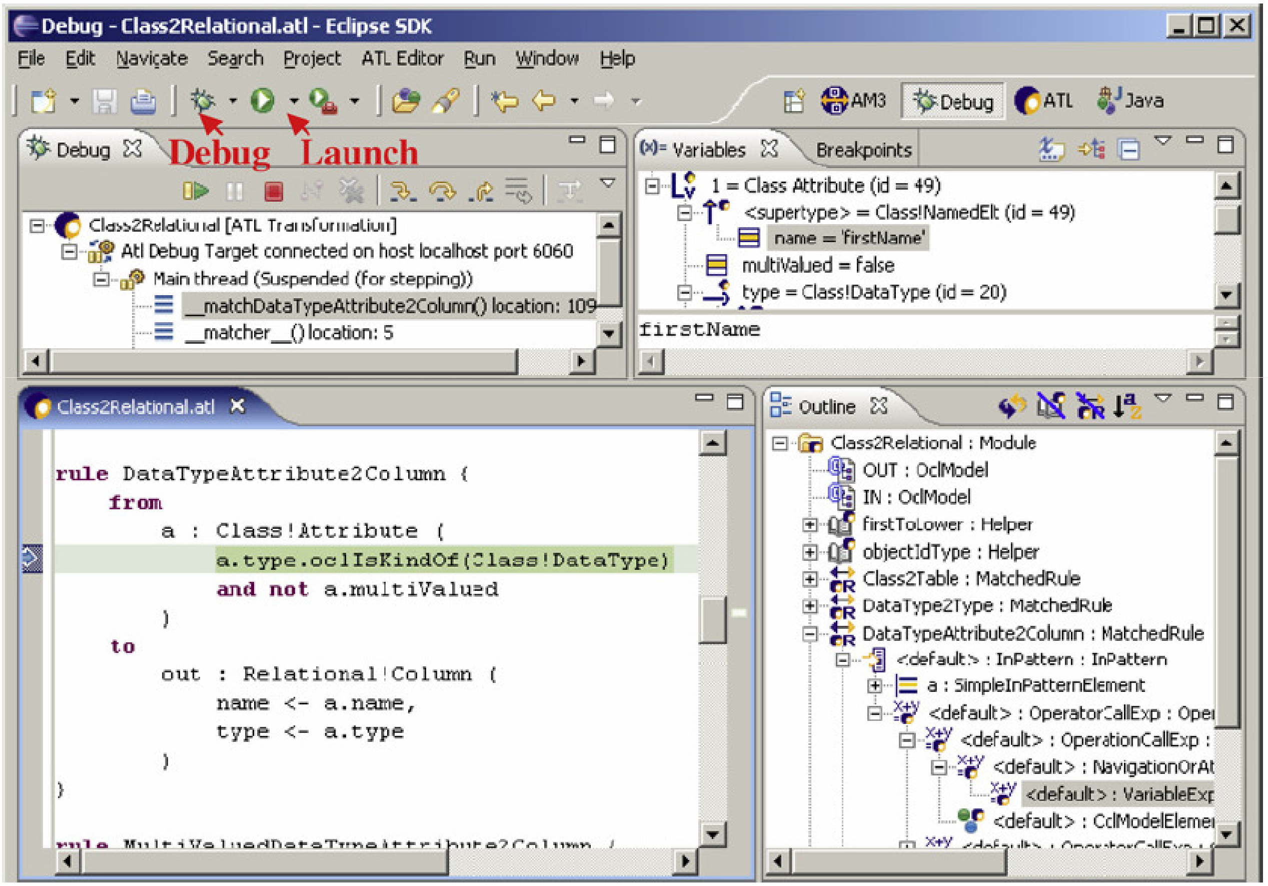


Fig. 6. The architecture of the ATL execution engine.

Editing and Debugging


Debug - Class2Relational.atl - Eclipse SDK
File Edit Navigate Search Project ATL Editor Run Window Help
AM3 Debug ATL Java
Debug
Debug
Launch
Variables
Breakpoints
Class2Relational.atl
Outline

```
Class2Relational [ATL Transformation]
  Atl Debug Target connected on host localhost port 6060
    Main thread (Suspended (for stepping))
      __matchDataTypeAttribute2Column() location: 109
      __matcher__() location: 5
```

```
1 = Class Attribute (id = 49)
  <supertype> = Class!NamedElt (id = 49)
    name = 'firstName'
  multiValued = false
  type = Class!DataType (id = 20)
firstName
```

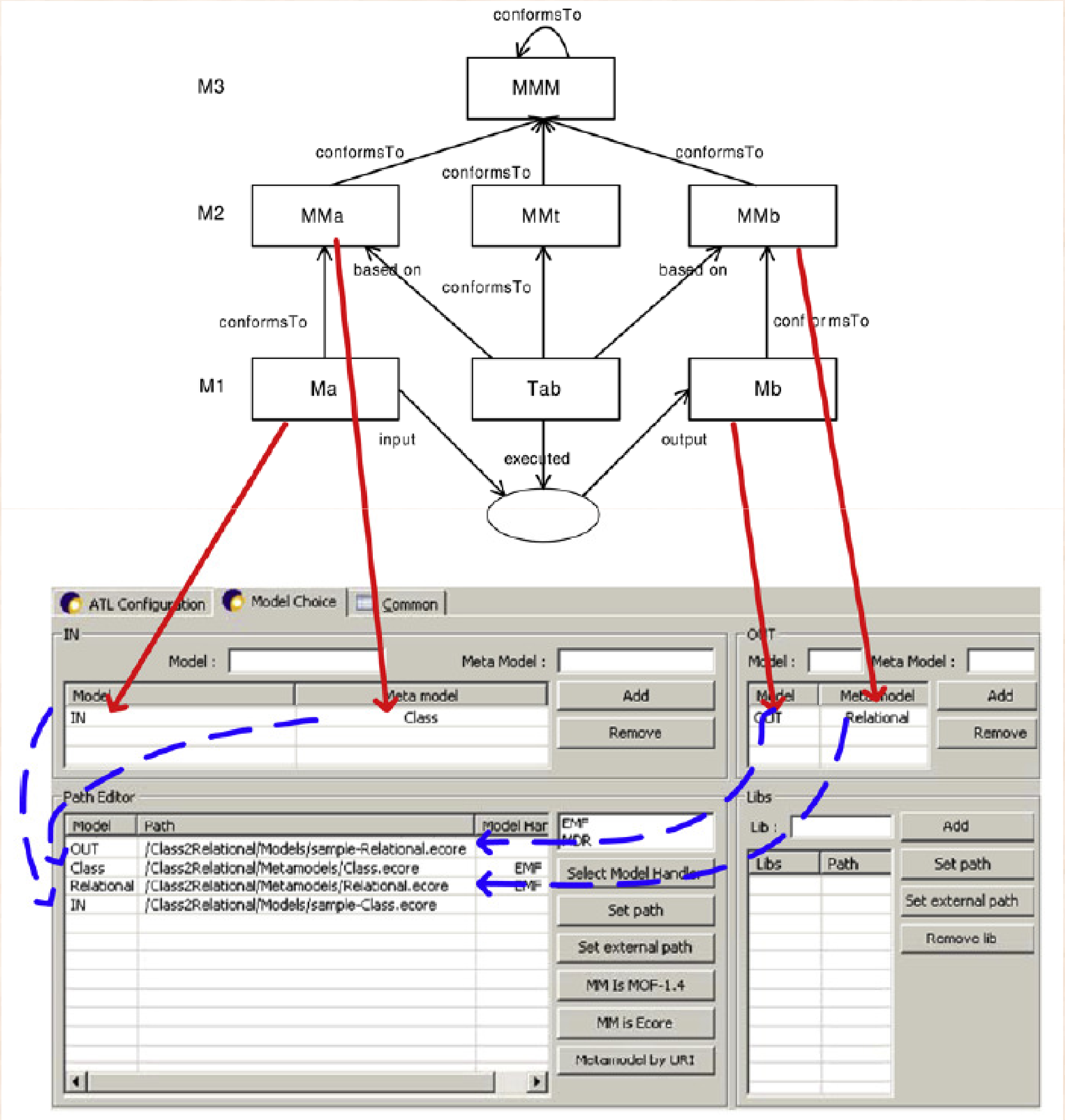
```
rule DataTypeAttribute2Column {
    from
        a : Class!Attribute (
            a.type.oclIsKindOf(Class!DataType)
            and not a.multiValued
        )
    to
        out : Relational!Column (
            name <- a.name,
            type <- a.type
        )
}

rule MultiValuedDataTypeAttribute2Column (
```

```
Class2Relational : Module
  OUT : OclModel
  IN : OclModel
  firstToLower : Helper
  objectIdType : Helper
  Class2Table : MatchedRule
  DataType2Type : MatchedRule
  DataTypeAttribute2Column : MatchedRule
    <default> : InPattern : InPattern
      a : SimpleInPatternElement
      <default> : OperatorCallExp : Oper
        <default> : OperationCallExp :
          <default> : NavigationOrAt
            <default> : VariableExp
          <default> : OclModelElemen
```

Fig. 7. ATL editor and debugger screenshot.

Building and Launching



IN

Model :

Meta Model :

Model	Meta model
IN	Class

OUT

Model :

Meta Model :

Model	Meta model
OUT	Relational

Path Editor

Model	Path	Model Har
OUT	/Class2Relational/Models/sample-Relational.ecore	
Class	/Class2Relational/Metamodels/Class.ecore	EMF
Relational	/Class2Relational/Metamodels/Relational.ecore	EMF
IN	/Class2Relational/Models/sample-Class.ecore	

EMF
MDR

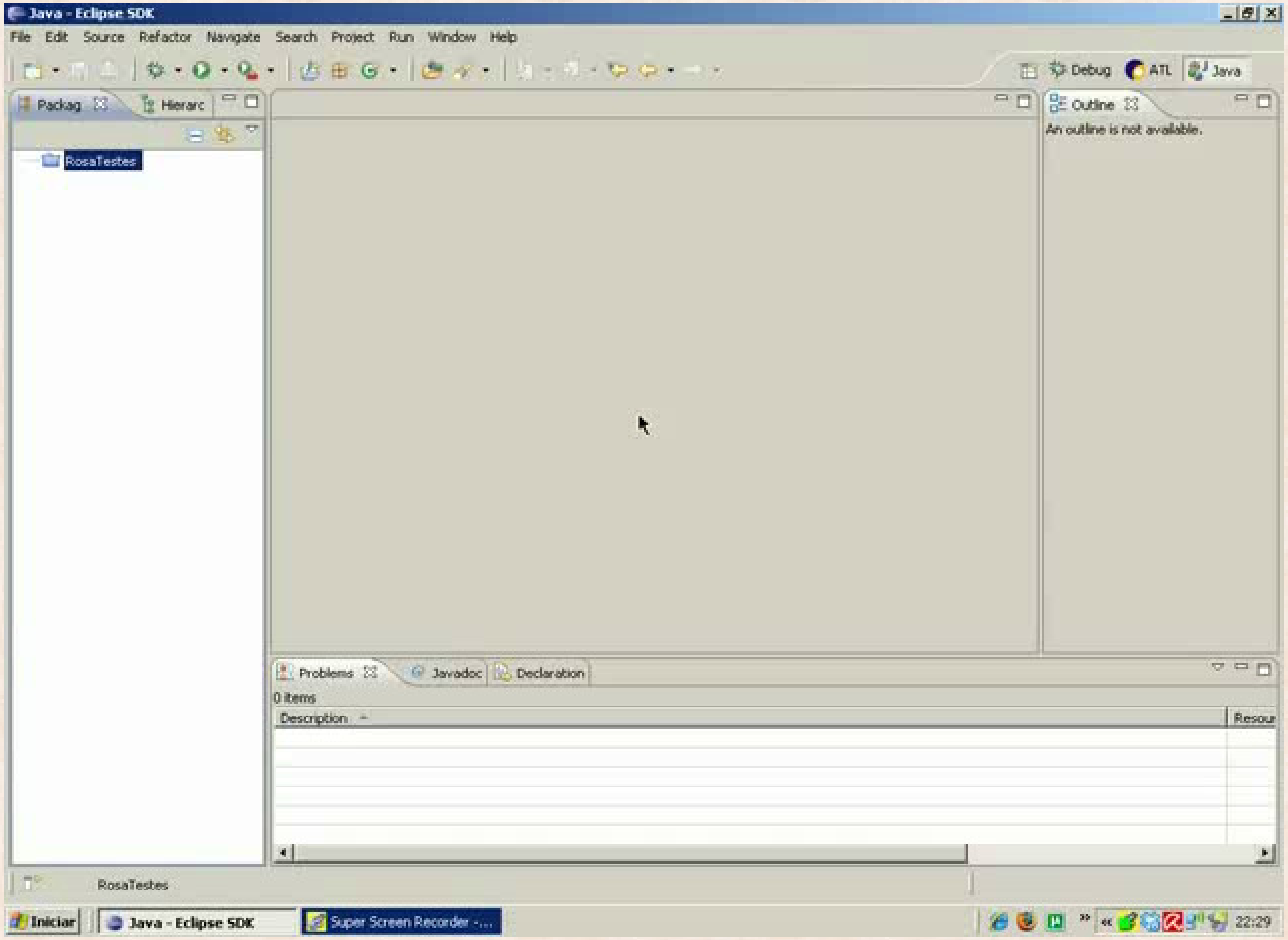
Libs

Lib :

Libs	Path

Fig. 8. ATL launch configuration.

- Source model Ma is declared as model IN with metamodel Class.
- The file corresponding to model IN is sample-Class.ecore.


Java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Debug ATL Java
Packag Hierarc
RosaTestes
Outline
An outline is not available.
Problems Javadoc Declaration
0 items
Description
Resou
RosaTestes
Iniciar
Java - Eclipse SDK
Super Screen Recorder -...
22:29

Evaluation

- The tool supports syntax highlighting and error reporting.
- Debugging is also available.
- Not support visual language.
- Limited language support for source and target models. (XMI OMG serialization format)

Summary

- ATL is support by a set of development tools built on top of the Eclipse environment.
- It allows both imperative and declarative approaches for transformation definitions.
- More than 160 individual transformations are available on ATL M2M website.

Thank You

Discussion

- Will you use ATL if it supports both textual and graphical notation in the future?

Discussion

- Would it be useful if reverse engineering is applied on ATL to support bidirectional transformation?

Discussion

- How to verify the correctness of model transformation using ATL?