Institut de Recherche en Informatique de Toulouse
INPT-ENSEEIHT
2, Rue Charles Camichel - BP 7122 - 31071 Toulouse cedex 7 - France

Extending Persistent Meta-Modeling Systems to Handle Behavioral Semantics

Yamine AIT AMEUR[1] and Youness BAZHAR[2]

[1]IRIT/ENSEEIHT, [2]LIAS/ISAE-ENSMA

yamine@n7.fr youness.bazhar@ensma.fr

Constantine November 10th, 2012
IWAISE Conference

Introduction

Models and modelling languages

- Modelling languages manipulate structures
  - Classes, properties,
  - Entities, associations,
  - State-transitions,
  - Data flow
  - Constraints
  - ...
- Model real world concepts
  - Databases,
  - Ontologies
  - Services,
  - Programs,
  - …

Introduction

Models and modelling languages

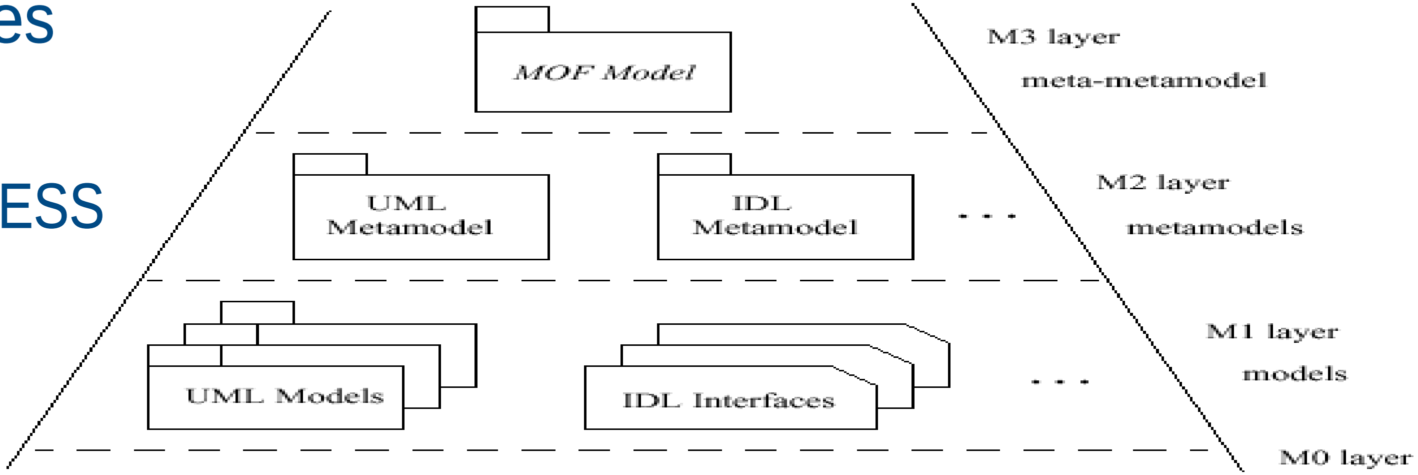
- Support of different analyses
  - Code generation,
  - Type checking,
  - Proofs and model checking,
  - Constraints solving
  - Test generation,
  - Various V & V techniques
- Analyses are defined by programs that run on models
  - They manipulate models.
- Different modelling languages
  - UML and its family
  - functional, state based modelling languages
  - ...
- Different semantic representations
  - Formal models
  - Tools give an operational semantics

Introduction
Models as cake machines

- Models and model instances
- Models are described as instances of meta-models

- Exchange format for instance, model and meta-model representations

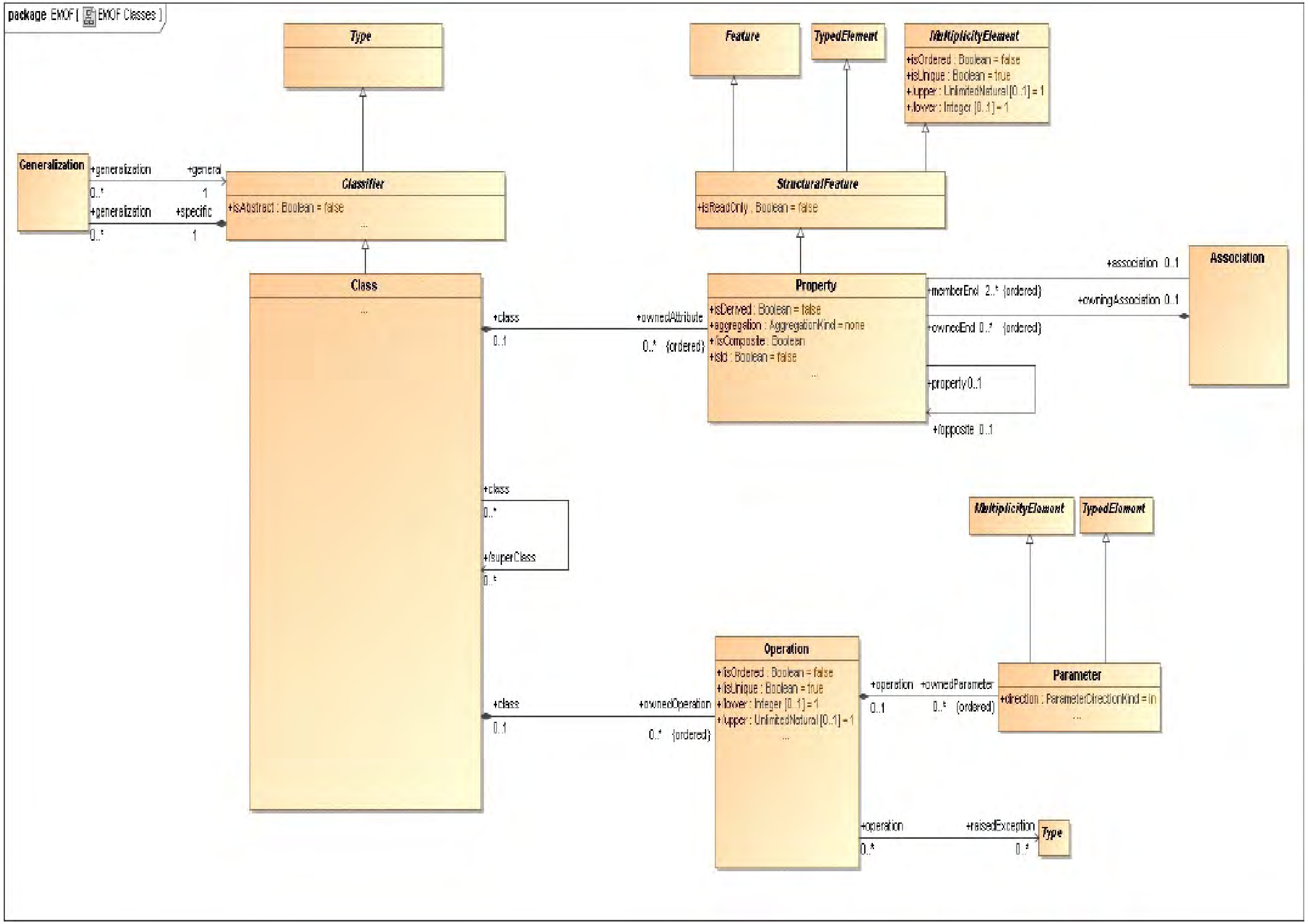
- Examples
  - MOF
  - EXPRESS



MOF Metadata Architecture

Extending persistent meta-modelling systems to handle behavioural semantics
Yamine AIT AMEUR and Youness BAZHAR – Constantine November 10th, 2012

Introduction
The MOF



Introduction
Model manipulation

- Models become objects that can be manipulated
- Meta-models and meta-modelling are good candidates for model representation
- Model management systems [Bernstein]
  - An algebra of operators for model management
  - Rondo System
- Define operators that operate on models and model concepts

Introduction
MMS

- A Meta-Modelling System
  - Representation of instances, models and meta-models (the M0, M1, M2 and M3 layers)
  - Support of manipulation operators
    - Access, Creation code generation, etc.
    - APIs play a crucial role
  - Example
    - EMF for the MOF
    - ECO Toolkit for EXPRESS
    - ...

Introduction

MMS manipulation

- Need to manipulate models
  - Extract, Query, Search,
  - Transform, Integrate, Compose, Annotate
  - Store, Retrieve,
  - Different model analyses
  - Etc.

- Several approaches have addressed
  - Structural and descriptive knowledge for models.
  - Hard encoded operators
  - No possibility of extension
  - Based on static APIs

Introduction

Scalability

- Engineering make an extensive use of modelling
- A big quantity of models are produced every day
  - Origin of the work
  - Limitations of the current approaches
    - Model loading
- Classical meta-modelling systems fail to support oversized models
  - Example: Eclipse EMF framework
- **Towards a repository of models**
  - **Persistent solutions are required**

Plan

- Introduction
- **Persistent Meta-Modelling systems**
- Handling behavioural semantics
- The case of ontology concepts
- Conclusion

Persistent MMS

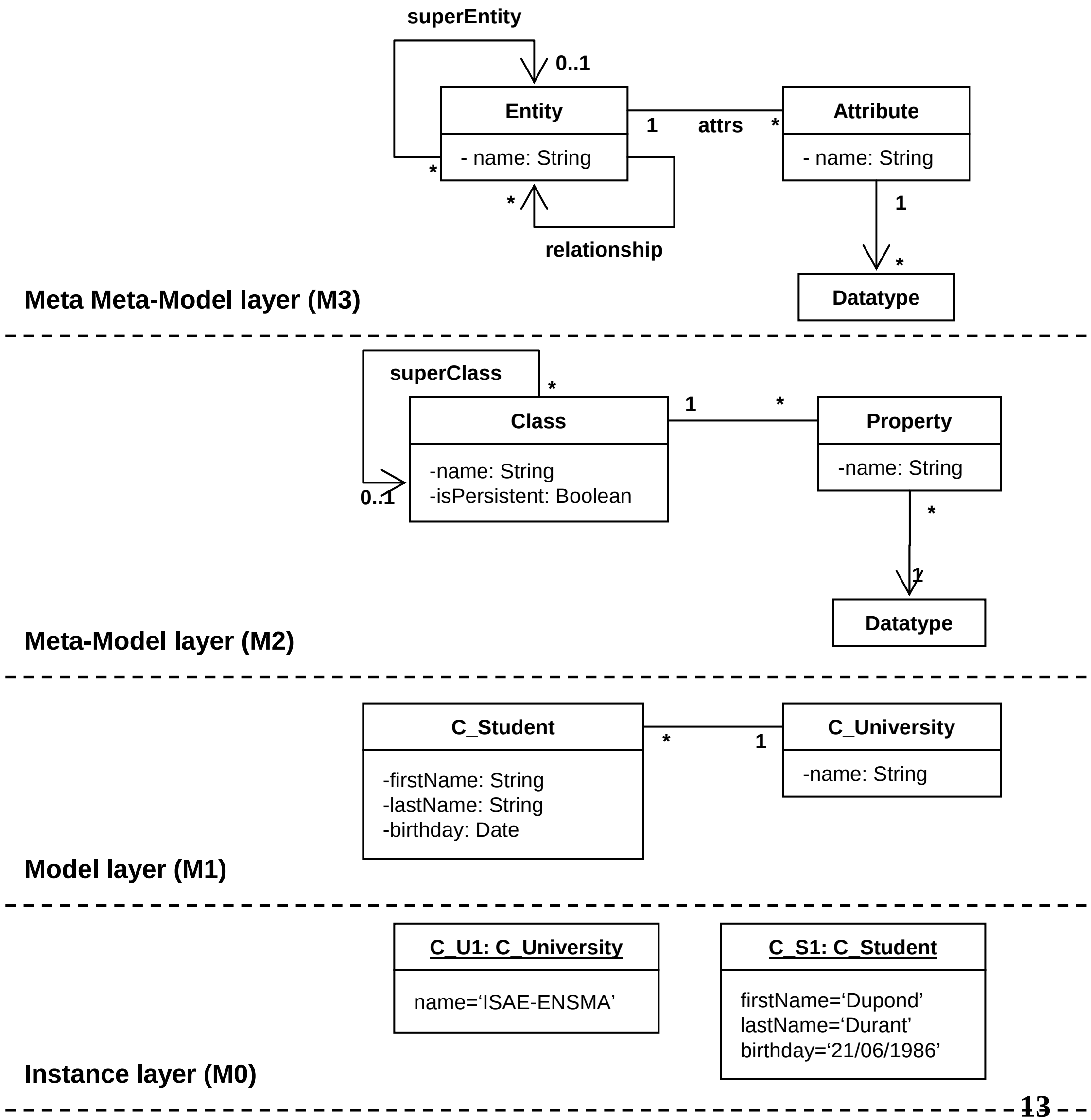
- A persistent MMS
  - is a MMS where
  - the
    - **Models** and
    - the whole **meta-modelling architecture**
  - are stored in a database
  - and
    - an **exploitation language** is available for this database
  - Power of the language ?

Persistent MMS

An illustrative example

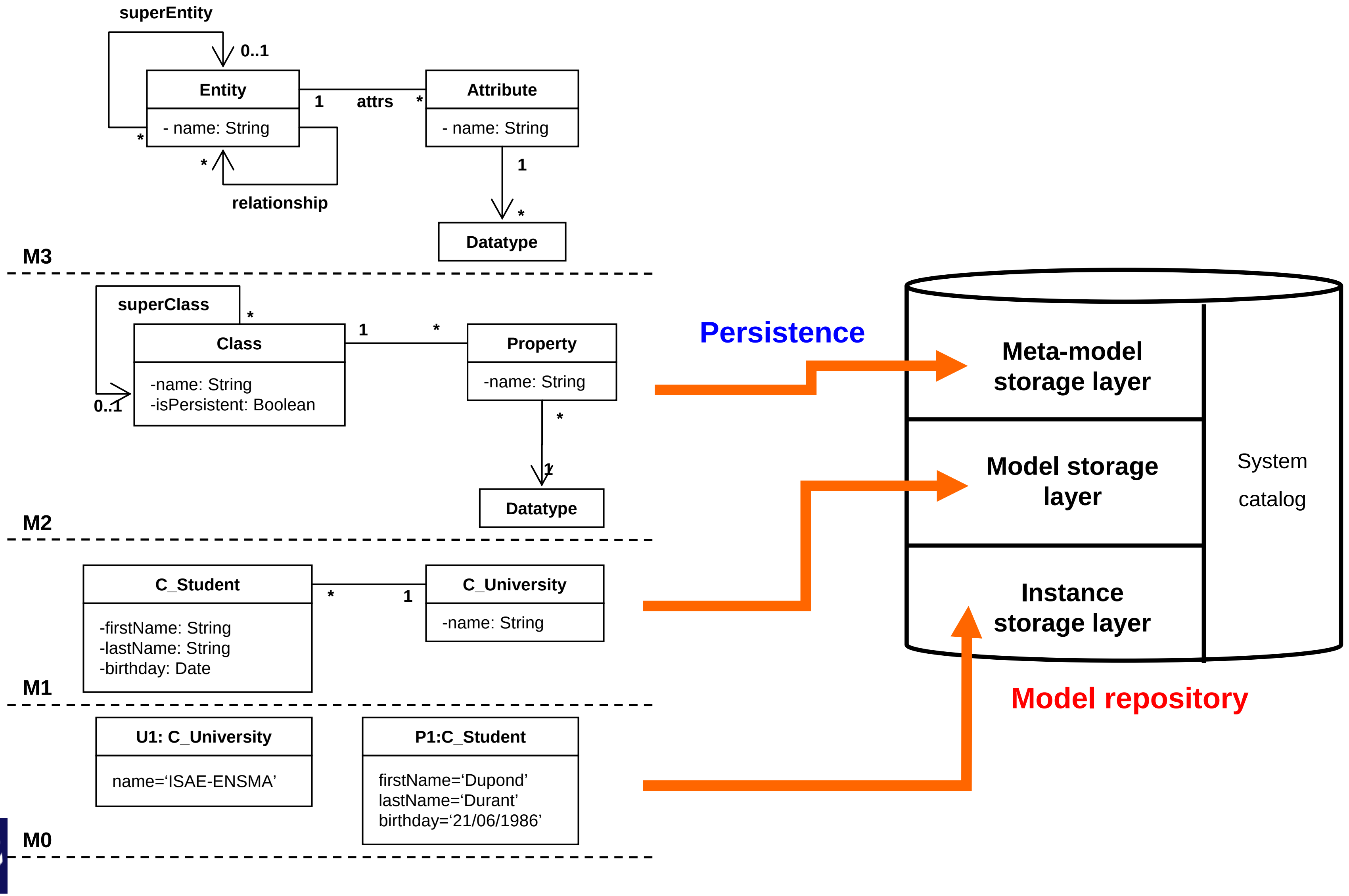
- A PMMS
  - ONDTODB[Dehainsala]
- An exploitation language
  - ONTOQL [Jean]
- Dynamic instanciation of the M3 level

Persistent MMS
An illustrative example



Persistent MMS

An illustrative example

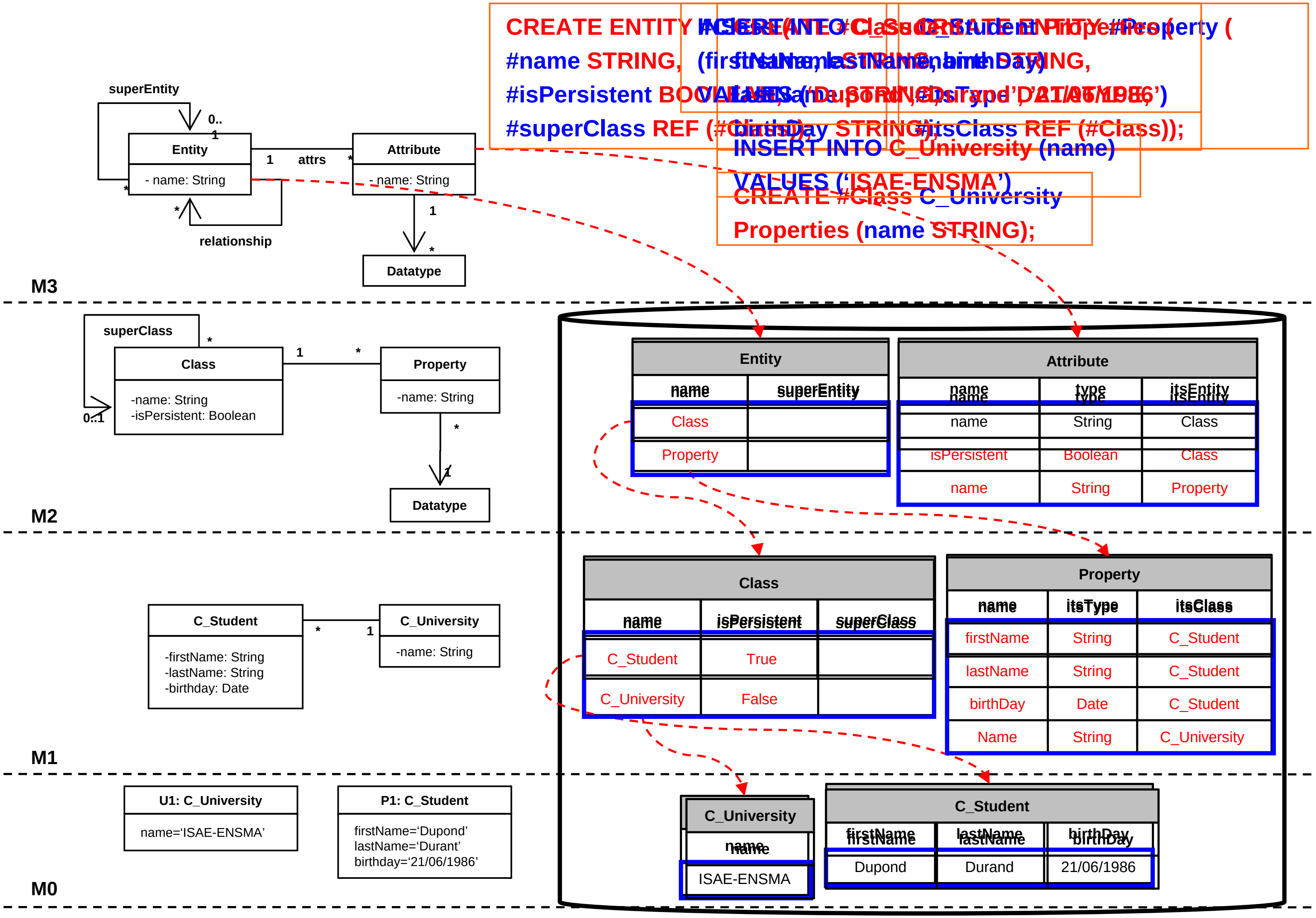


Persistent MMS

An illustrative example

The OntoQL exploitation language

- Introduction
- Persistent MMS
- Handling behaviors …
- The case of ontologies
- Conclusion



CREATE ENTITY #Class (
#name STRING,
#isPersistent BOOLEAN,
#superClass REF (#Class));

CREATE ENTITY #Property (
#name STRING,
#itsType STRING,
#itsClass REF (#Class));

CREATE #Class C_Student
Properties (firstName STRING, lastName STRING,
birthDay STRING);

INSERT INTO C_Student
(firstName, lastName, birthDay)
VALUES ('Dupond', 'Durant', '21/06/1986')

INSERT INTO C_University (name)
VALUES ('ISAE-ENSMA')

CREATE #Class C_University
Properties (name STRING);

Entity

name	superEntity
Class	
Property	

Attribute

name	type	itsEntity
name	String	Class
isPersistent	Boolean	Class
name	String	Property

Class

name	isPersistent	superClass
C_Student	True	
C_University	False	

Property

name	itsType	itsClass
firstName	String	C_Student
lastName	String	C_Student
birthDay	Date	C_Student
Name	String	C_University

C_University

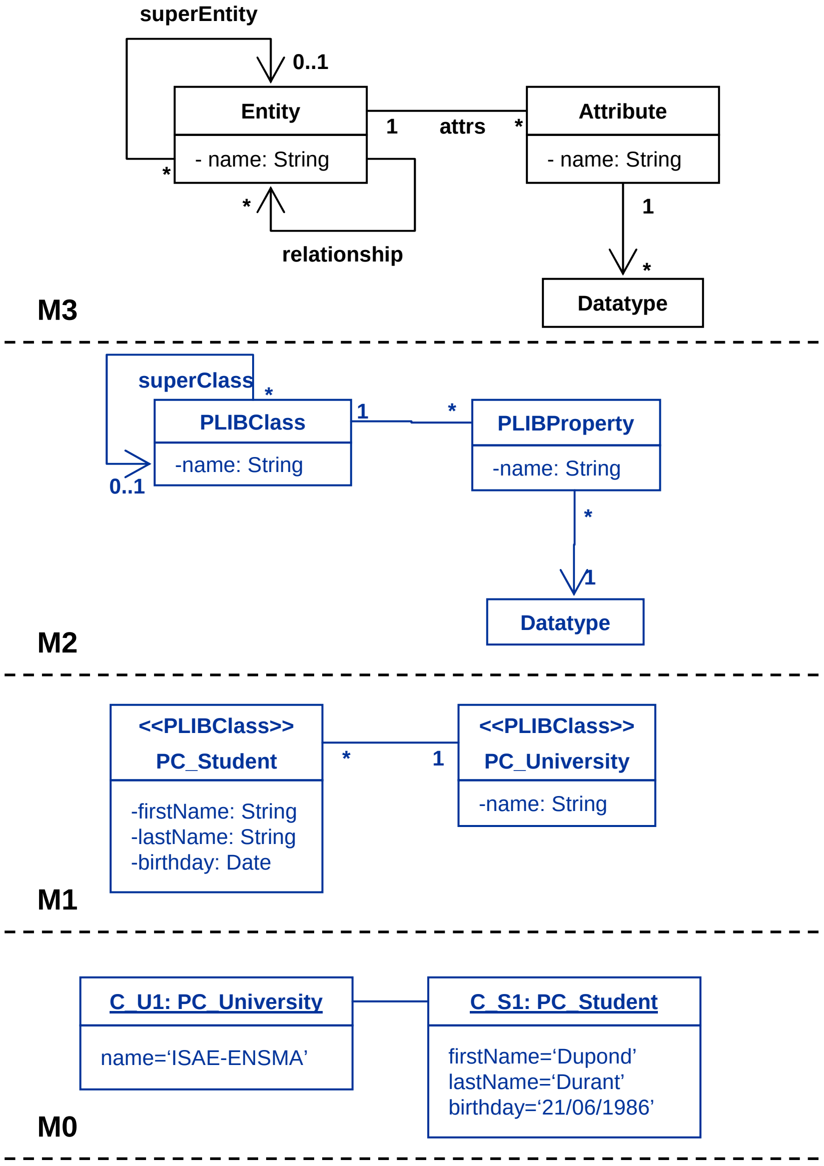
name
ISAE-ENSMA

C_Student

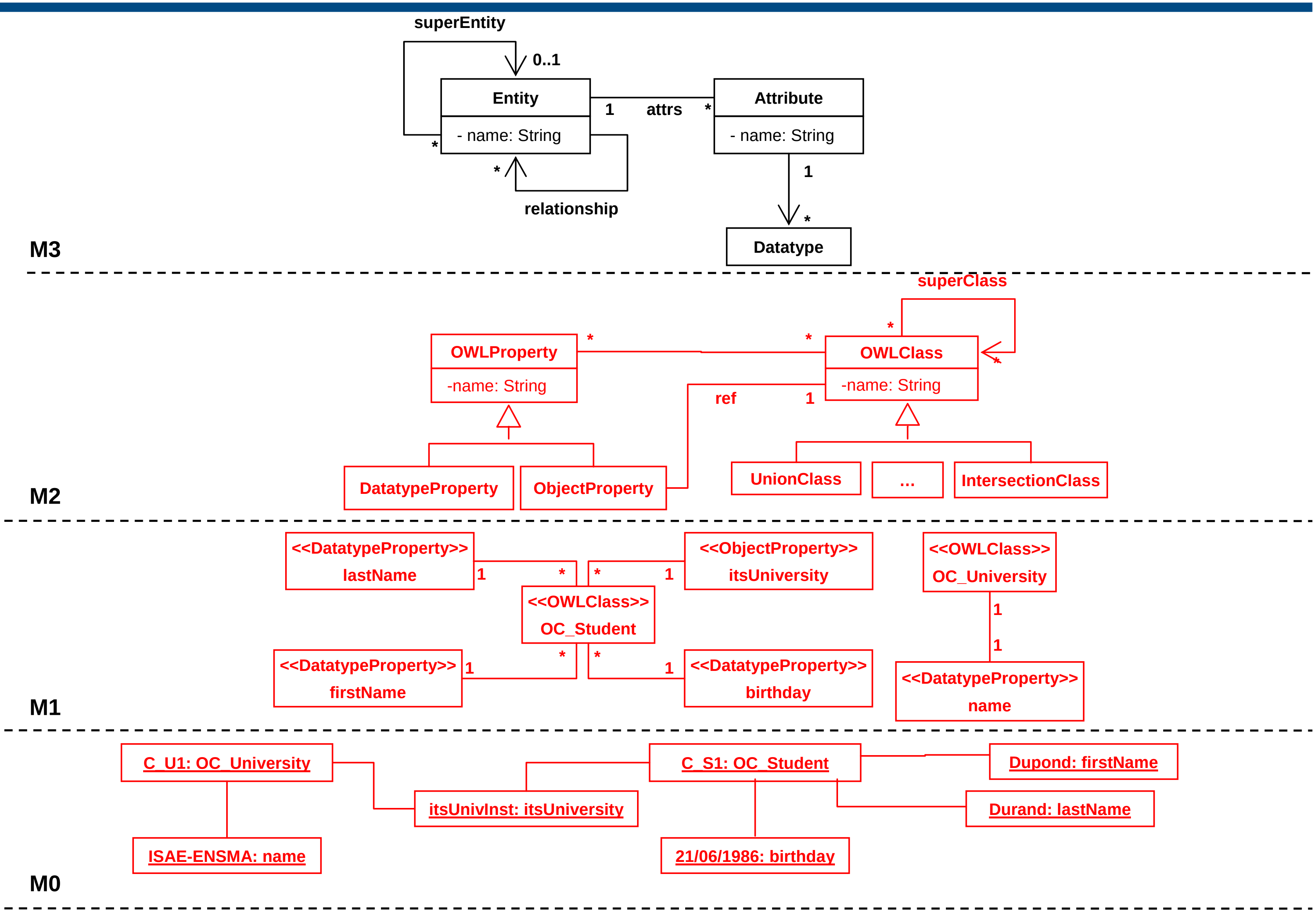
firstName	lastName	birthDay
Dupond	Durant	21/06/1986

Persistent MMS
Instantiation for the PLIB model

- Introduction
- Persistent MMS
- Handling behaviors …
- The case of ontologies
- Conclusion



Persistent MMS
Instantiation for the OWL model



M3

superEntity
0..1
Entity
- name: String
*
1
attrs
*
Attribute
- name: String
*
relationship
1
*
Datatype

M2

superClass
*
*
*
OWLProperty
-name: String
OWLClass
-name: String
*
ref
1
DatatypeProperty
ObjectProperty
UnionClass
…
IntersectionClass

M1

<<DatatypeProperty>>
lastName
1
*
*
1
<<ObjectProperty>>
itsUniversity
<<OWLClass>>
OC_University
<<OWLClass>>
OC_Student
1
1
<<DatatypeProperty>>
firstName
1
*
*
1
<<DatatypeProperty>>
birthday
<<DatatypeProperty>>
name

M0

C_U1: OC_University
C_S1: OC_Student
Dupond: firstName
itsUnivInst: itsUniversity
Durand: lastName
ISAE-ENSMA: name
21/06/1986: birthday

Persistent MMS
Limitations of PMMS

- Limitations
  - Structural and descriptive knowledge representation
  - The exploitation language addresses structural and descriptive parts
  - Absence of behaviours
    - Constraints, derivations, etc.
- Is the PMMS capable to handle
  - Model transformation ?
    - Example: Class2Table

```
SELECT #name, #superClass
FROM #PLIBClass
```

```
SELECT #name, #type, #class
FROM #PLIBProperty
```

```
SELECT #name, #properties
FROM #OWLClass
```

```
SELECT #name, #type
FROM #Property
```

Model transformation

Limitations of PMMS exploitation languages

- Limitations
  - Structural and descriptive knowledge representation
  - The exploitation language addresses structural and descriptive parts
  - Absence of behaviours
- Is the PMMS capable to handle
  - Derivations ?
    - Example: computeAge(birthday)

```
SELECT firstName, lastName, birthday
FROM PC_Student
```

```
SELECT name
FROM PC_University
```

Compute derivations:

- **Age**
- **Complete name**
- **Etc.**

✗

Persistent MMS

Assessment

- Current PMMS handle
  - Structural and descriptive semantics
- OntoQL is an exploitation language for PMMS
  - Exploit both models of M0, M1, M2 and M3.
- Is the PMMS capable to handle
  - Model transformation ?
    - ❖Example: Class2Table
  - Derivations ?
    - ❖Example: computeAge(birthday)
  - …

Persistent MMS Assessment

- Insufficiencies
  - Constraints and derivations are not handled
  - Completeness of the language is not ensured
  - Heterogeneous languages need to be integrated
  - No capability of model manipulation nor analysis
- Needs to handle richer behaviours

Plan

- Introduction
- Persistent Meta-Modelling systems
- **Handling behavioural semantics**
- The case of ontology concepts
- Conclusion

Handling behavioral semantics

State of the art

- Embedded DB procedural languages
  - Support the expression of programs in a native language
  - Examples
    - PL/SQL, PL/PGSQL
- Limitations
  - No possibility to manipulate models and meta-models
  - Limited to the system catalog (tables, columns)
  - Mono-language
  - Weak expressive power compared to programming languages

Handling behavioral semantics

State of the art

- Definition of APIs
  - Data access API like JDBC
  - Based on the logical model
    - GetTableName
    - GetColumnType
    - …
- Limitations
  - No possibility to manipulate models and meta-models
  - Limited to the system catalog (tables, columns)
  - Mono-language
  - Unidirectional

Handling behavioral semantics

State of the art

- Introduction
- Persistent MMS
- Handling behaviors …
- The case of ontologies
- Conclusion

- Object-Relational mapping API
  - Data models are defined and APIs are generated
    - Example: Hibernate Framework
  - Hidden logical models

- Limitations
  - No possibility to manipulate meta-models
    - ==> Static models
  - Mono-language
  - Unidirectional

Handling behavioral semantics

State of the art

- Persistent programming languages
  - Definition of Transient objects
  - Example JPA, JDO Frameworks
  - Hidden logical model
  - The persistent model can be parameterized

- Limitations
  - No possibility to manipulate meta-models
    - ==> Static models
  - Mono-language
  - Unidirectional

Handling behavioral semantics

State of the art

- These solutions use internal and specific mechanisms for
  - Derivations
  - Model transformations
  - Constaints
  - ...
- Examples of such mechanisms
  - Views
  - Frozen and hard-encoded operators
  - ...
- **No conformance with MOF architecture**
- **Non extensible**
  - **Introduce new operators**
- **Procedural aspects are**
  - **Supported in the native language of the PMMS**
  - **Limited to the expressive power of the native language**

Handling behavioral semantics

Requirements (1/2)

- Handling different modeling features
  - Structural and descriptive semantics
  - Behavioral semantics
    - Model transformations
    - Derivations
    - …
  - Constraints definition and checking
- Need of persistence
  - Large-scale models and data
  - An algebra of operators for behavior persistence
    - Create
    - Update
    - Select
    - Delete
    - **Run**

Handling behavioral semantics

Requirements (2/2)

- Conformance with MOF
  - 4 abstraction levels
  - MOF meta-model support
- Provide powerful programming capabilities and flexibility
  - Use programming languages (e.g. Java, C++)
  - Remote services (e.g. web services)
  - ...
- Single access interface for programs, data and models
  - bidirectional

PMMS with behavioral semantics

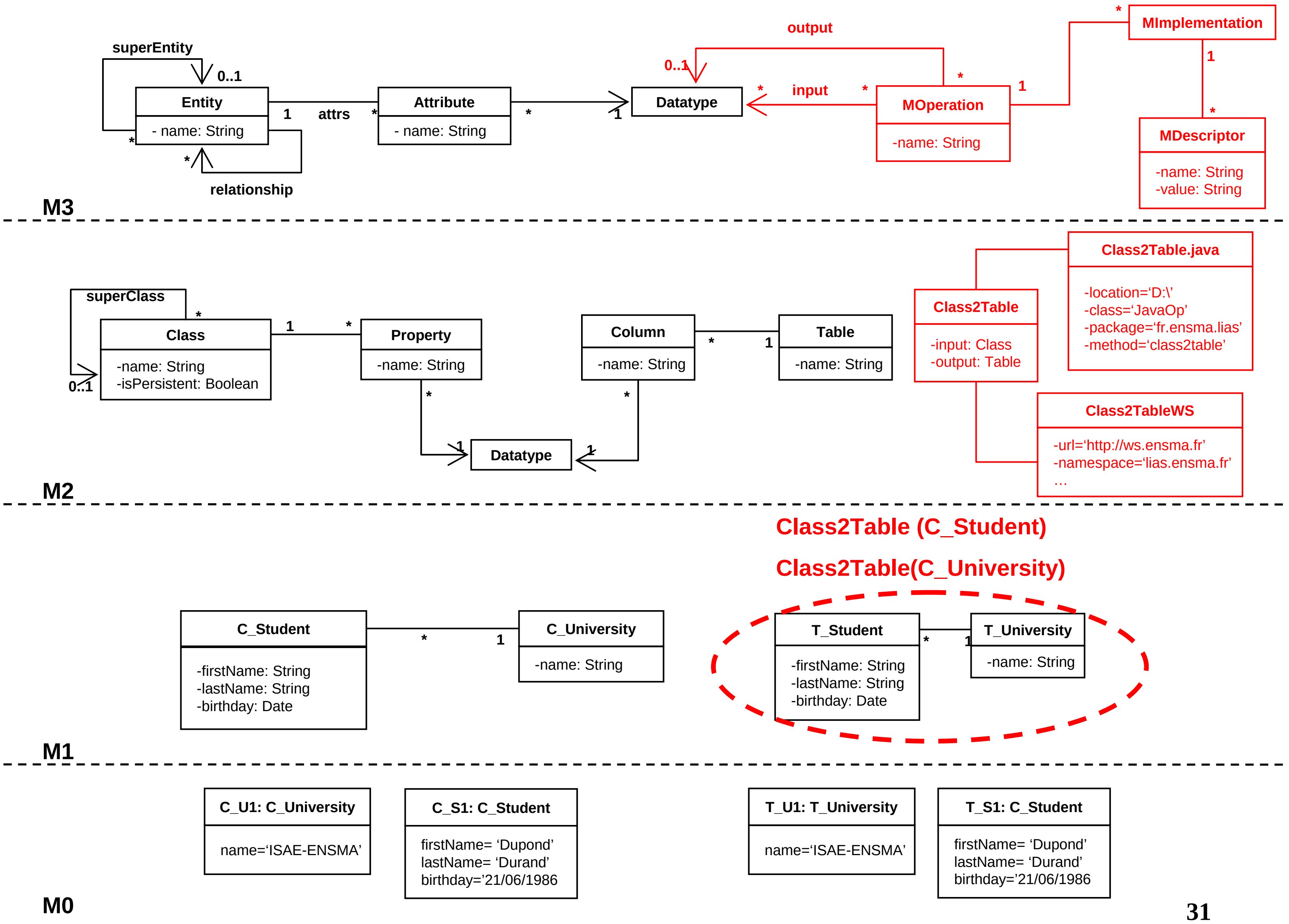
The proposed approach

- A stepwise approach based on the extension of classical PMMS
  - Enrich the M3 model an operation concept made of two parts
    - ❖operation profile
      - providing the operation signature
    - ❖Operation implementation
      - providing the operation implementation meta-data
        - » Operation call
  - Follow the classical meta-modeling techniques

Handling behavioral semantics

Class or model operations

- Introduction
- Persistent MMS
- **Handling behaviors …**
- The case of ontologies
- Conclusion



M3

Entity: - name: String; superEntity 0..1 / *; relationship *; attrs 1 / *

Attribute: - name: String; * → 1 Datatype

Datatype

MOperation: -name: String; input * / *; output 0..1 / *; 1 / * MImplementation

MImplementation; 1 / * MDescriptor

MDescriptor: -name: String, -value: String

M2

Class: -name: String, -isPersistent: Boolean; superClass * / 0..1; 1 / * Property

Property: -name: String; * → 1 Datatype

Column: -name: String; * / 1 Table; * → 1 Datatype

Table: -name: String

Datatype

Class2Table: -input: Class, -output: Table

Class2Table.java: -location='D:\' -class='JavaOp' -package='fr.ensma.lias' -method='class2table'

Class2TableWS: -url='http://ws.ensma.fr' -namespace='lias.ensma.fr' …

Class2Table (C_Student)

Class2Table(C_University)

M1

C_Student: -firstName: String, -lastName: String, -birthday: Date; * / 1 C_University

C_University: -name: String

T_Student: -firstName: String, -lastName: String, -birthday: Date; * / 1 T_University

T_University: -name: String

M0

C_U1: C_University: name='ISAE-ENSMA'

C_S1: C_Student: firstName= 'Dupond' lastName= 'Durand' birthday='21/06/1986

T_U1: T_University: name='ISAE-ENSMA'

T_S1: C_Student: firstName= 'Dupond' lastName= 'Durand' birthday='21/06/1986

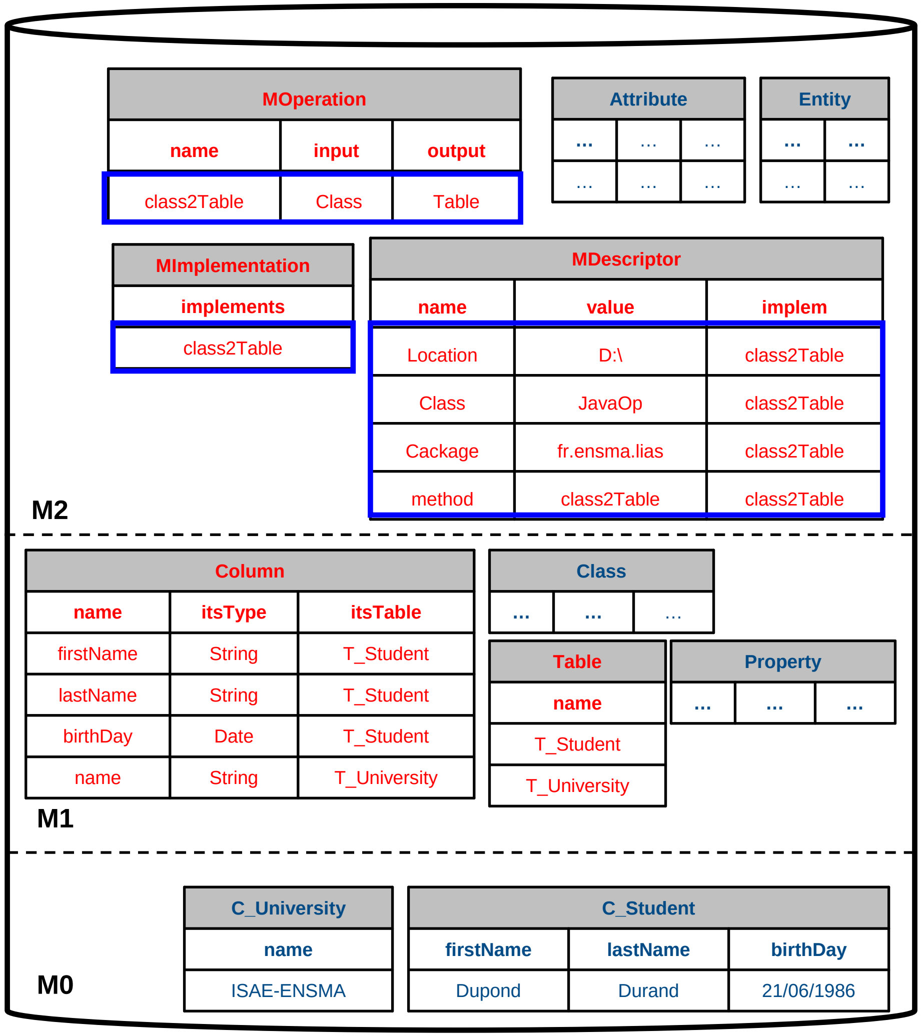
Handling behavioral semantics
Class or model operations

Extending persistent meta-modelling systems to handle behavioural semantics

Yamine AIT AMEUR and Youness BAZHAR – Constantine November 10th, 2012

class2Table

transformation



MOperation		
name	input	output
class2Table	Class	Table

Attribute		
…	…	…
…	…	…

Entity	
…	…
…	…

MImplementation
implements
class2Table

MDescriptor		
name	value	implem
Location	D:\	class2Table
Class	JavaOp	class2Table
Cackage	fr.ensma.lias	class2Table
method	class2Table	class2Table

Column		
name	itsType	itsTable
firstName	String	T_Student
lastName	String	T_Student
birthDay	Date	T_Student
name	String	T_University

Class		
…	…	…

Table
name
T_Student
T_University

Property		
…	…	…

C_University
name
ISAE-ENSMA

C_Student		
firstName	lastName	birthDay
Dupond	Durand	21/06/1986

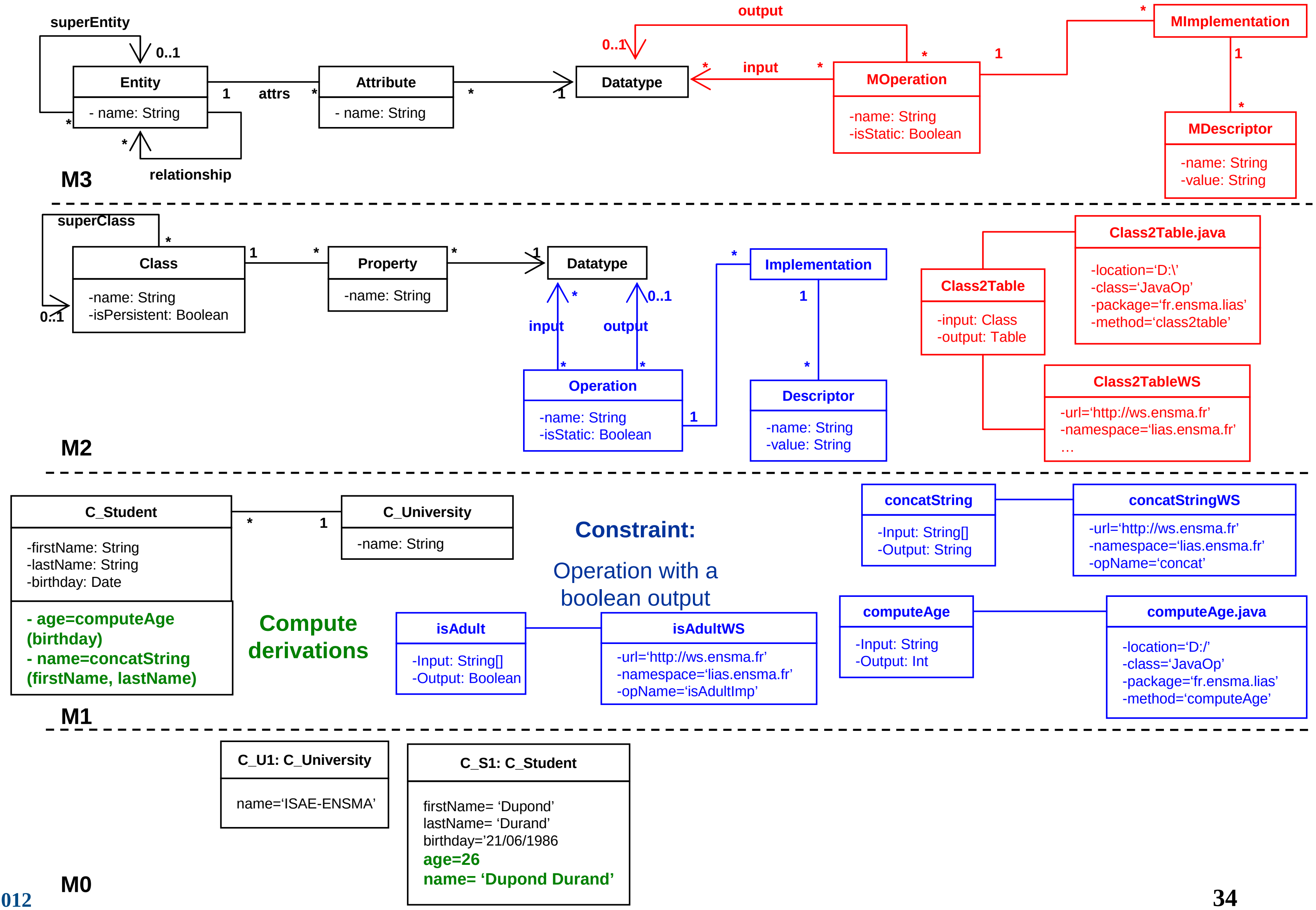
Handling behavioral semantics
Class or model operations

```
CREATE MOperation class2Table
INPUT REF (#Class)
OUTPUT REF (#Table);
```

```
CREATE MImplementation class2TableImp
Location=‘D:\workspace\JavaOp.jar’
Package=‘fr.ensma.lias’
Class=‘JavaOp’
Method=‘class2Table’
IMPLEMENTS class2Table;
```

```
CREATE #Table T_ University
AS class2Table (C_University)
With Implem class2TableImp;

CREATE #Table T_ Student
AS class2Table (C_Student)
With Implem class2TableImp;
```

Handling behavioral semantics
Instance operations



M3

superEntity 0..1

Entity
- name: String

1 attrs *

Attribute
- name: String

relationship

Datatype

output 0..1

input

MOperation
-name: String
-isStatic: Boolean

MImplementation

MDescriptor
-name: String
-value: String

M2

superClass 0..1

Class
-name: String
-isPersistent: Boolean

Property
-name: String

Datatype

input output

Operation
-name: String
-isStatic: Boolean

Implementation

Descriptor
-name: String
-value: String

Class2Table
-input: Class
-output: Table

Class2Table.java
-location='D:\'
-class='JavaOp'
-package='fr.ensma.lias'
-method='class2table'

Class2TableWS
-url='http://ws.ensma.fr'
-namespace='lias.ensma.fr'
…

M1

C_Student
-firstName: String
-lastName: String
-birthday: Date
- age=computeAge (birthday)
- name=concatString (firstName, lastName)

C_University
-name: String

Compute derivations

Constraint:

Operation with a boolean output

isAdult
-Input: String[]
-Output: Boolean

isAdultWS
-url='http://ws.ensma.fr'
-namespace='lias.ensma.fr'
-opName='isAdultImp'

concatString
-Input: String[]
-Output: String

concatStringWS
-url='http://ws.ensma.fr'
-namespace='lias.ensma.fr'
-opName='concat'

computeAge
-Input: String
-Output: Int

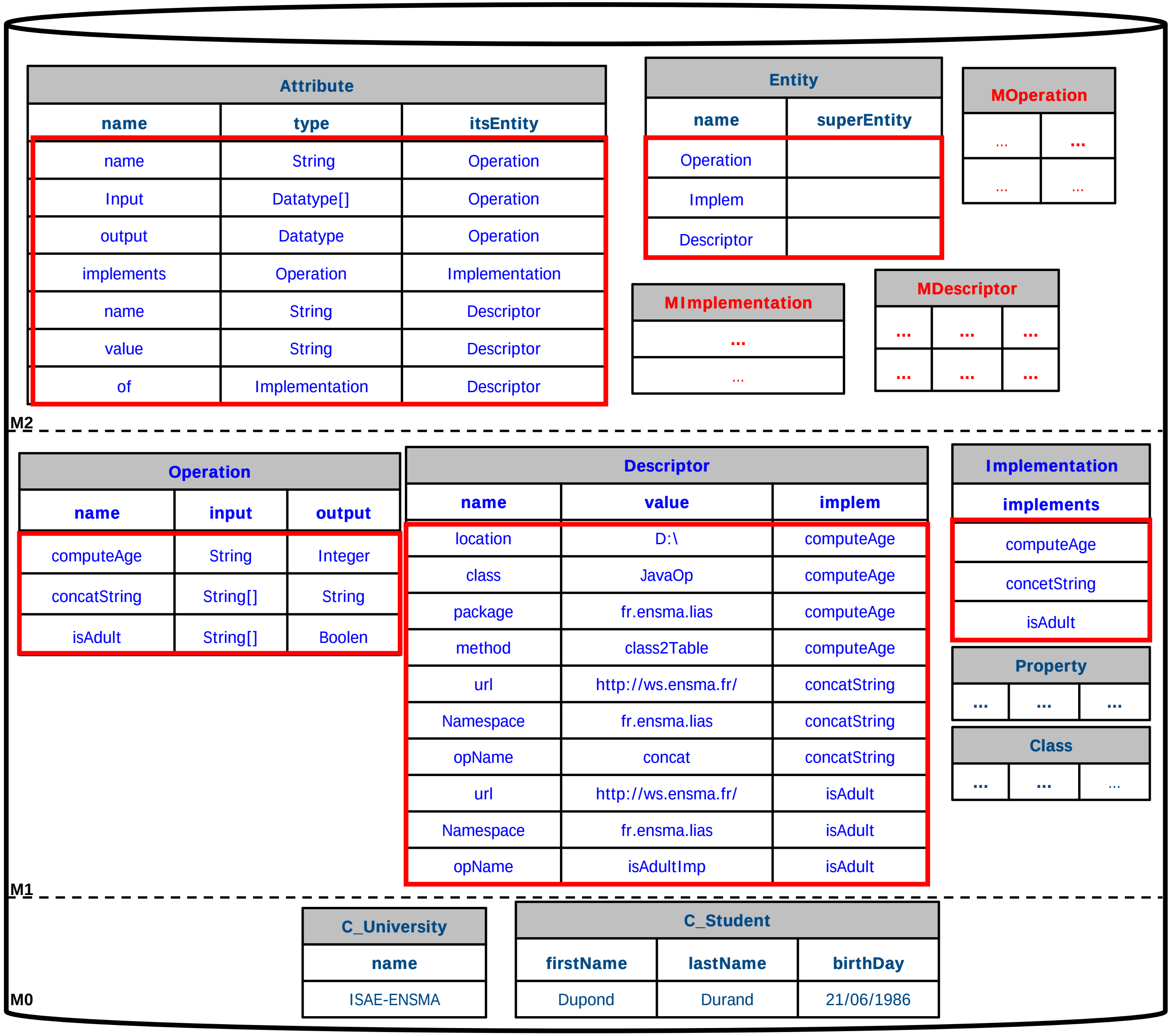
computeAge.java
-location='D:/'
-class='JavaOp'
-package='fr.ensma.lias'
-method='computeAge'

M0

C_U1: C_University
name='ISAE-ENSMA'

C_S1: C_Student
firstName= 'Dupond'
lastName= 'Durand'
birthday='21/06/1986
age=26
name= 'Dupond Durand'

Handling behavioral semantics
Instance operations



M2

Attribute		
name	type	itsEntity
name	String	Operation
Input	Datatype[]	Operation
output	Datatype	Operation
implements	Operation	Implementation
name	String	Descriptor
value	String	Descriptor
of	Implementation	Descriptor

Entity	
name	superEntity
Operation	
Implem	
Descriptor	

MOperation	
…	…
…	…

MImplementation
…
…

MDescriptor		
…	…	…
…	…	…

M1

Operation		
name	input	output
computeAge	String	Integer
concatString	String[]	String
isAdult	String[]	Boolen

Descriptor		
name	value	implem
location	D:\	computeAge
class	JavaOp	computeAge
package	fr.ensma.lias	computeAge
method	class2Table	computeAge
url	http://ws.ensma.fr/	concatString
Namespace	fr.ensma.lias	concatString
opName	concat	concatString
url	http://ws.ensma.fr/	isAdult
Namespace	fr.ensma.lias	isAdult
opName	isAdultImp	isAdult

Implementation
implements
computeAge
concetString
isAdult

Property		
…	…	…

Class		
…	…	…

M0

C_University
name
ISAE-ENSMA

C_Student		
firstName	lastName	birthDay
Dupond	Durand	21/06/1986

Handling behavioral semantics
Instance operations

- Introduction
- Persistent MMS
- **Handling behaviors …**
- The case of ontologies
- Conclusion

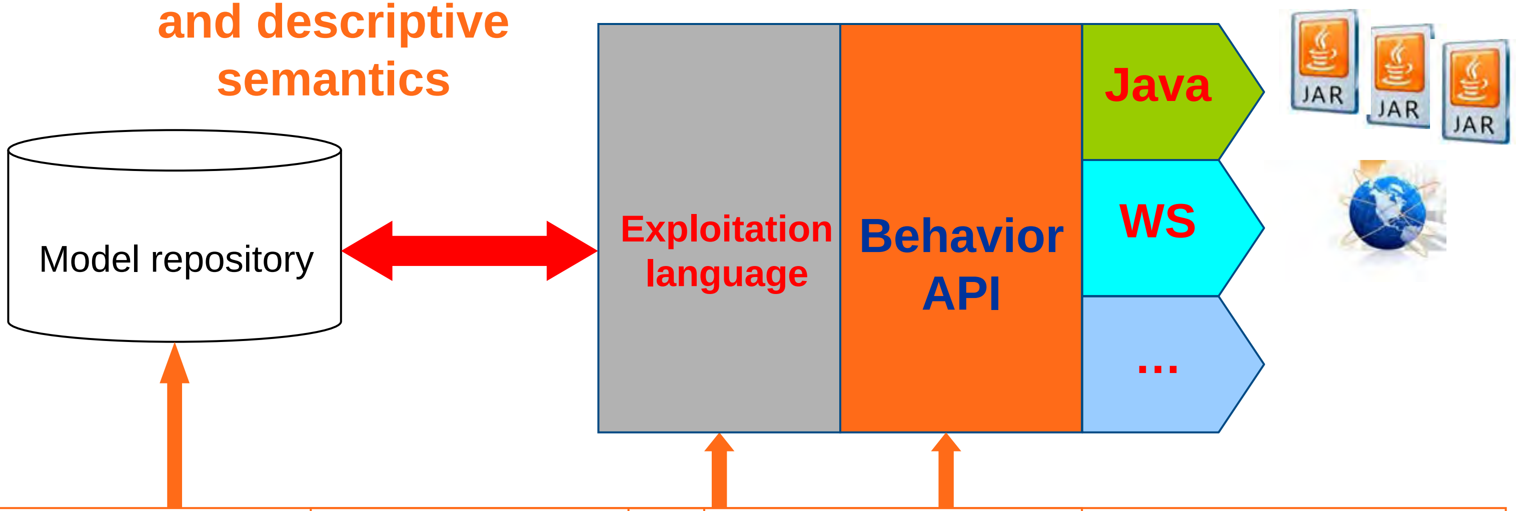
```
CREATE #Operation concatString
INPUT STRING ARRAY
OUTPUT STRING;

CREATE #Operation computeAge
INPUT STRING,
OUTPUT STRING;

CREATE #Operation isAdult
INPUT STRING,
OUTPUT BOOLEAN;
```

```
CREATE Implementation computeAgeImp
Location='D:\workspace\JavaOp.jar'
Package='fr.ensma.lias'
Class='JavaOp'
Method='computeAge'
IMPLEMENTS computeAge;

CREATE Implementation concatStringImp
url='http://ws.ensma.fr/services'
namespace='fr.ensma.lias'
opName='concat'
IMPLEMENTS concatString;

CREATE Implementation isAdultImp
url='http://ws.ensma.fr/services'
namespace='fr.ensma.lias'
opName='concat'
IMPLEMENTS isAdult;
```

```
SELECT computeAge (s.birthday)
FROM C_Student AS s
With computeAgeImp
WHERE isAdult(s.birthday) = TRUE;

SELECT concatString (s.firstName, s.lastName)
FROM C_Student AS s
With concatStringImp
WHERE isAdult(s.birthday) = TRUE;
```

Handling behavioural semantics Prototype

Support of structural and descriptive semantics



Model repository extension:
- Structures
- Store (Operations, Implementations and Descriptions)

Exploitation language:
- Operations definition
- Implementation definition
- Operations invocation

Behavior API:
- Embedded in the exploit. language
- Data mappings (between datatypes of PMMS and implementations)
- Implementations invocation

Plan

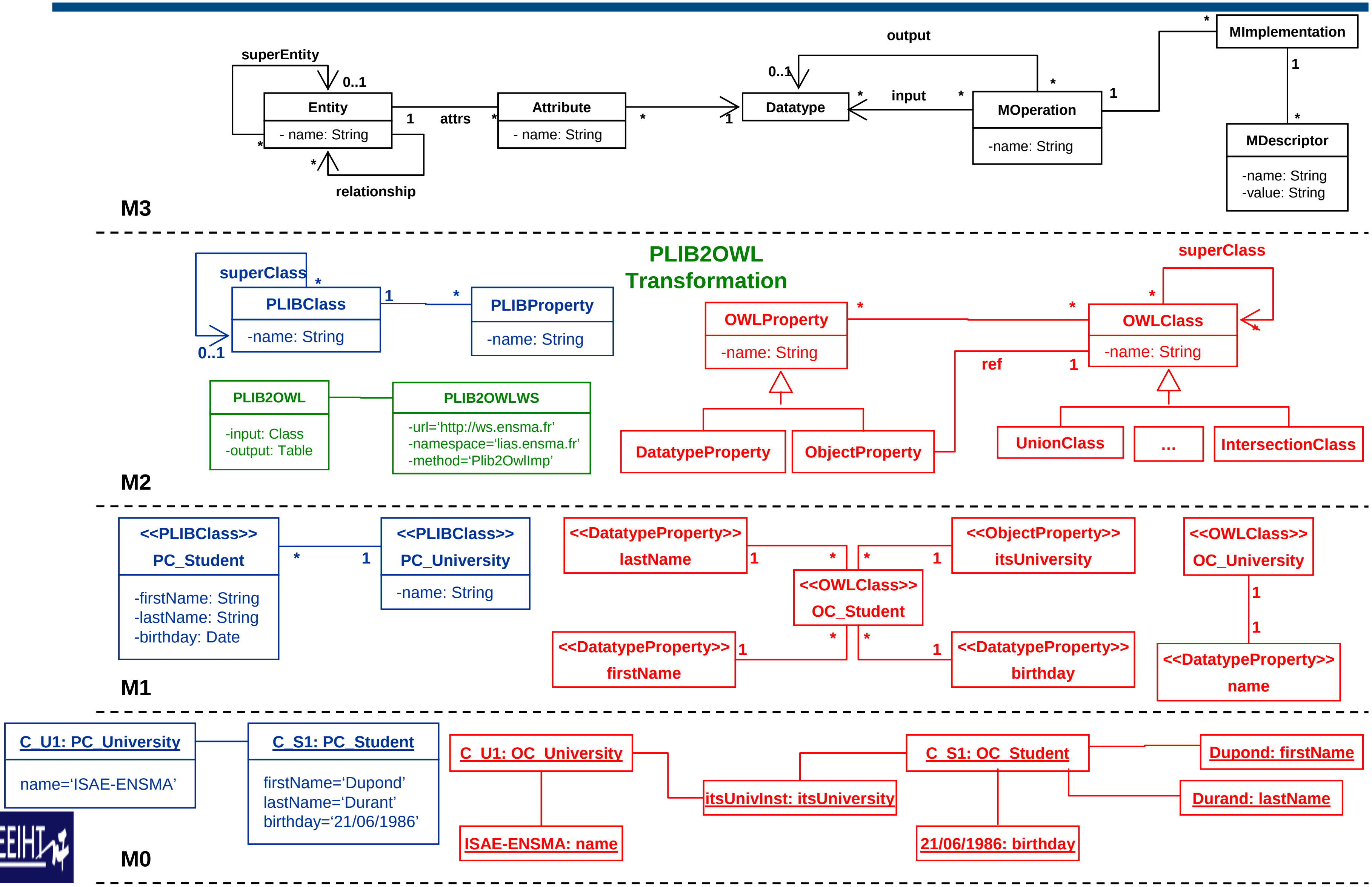
- Introduction
- Persistent Meta-Modelling systems
- Handling behavioural semantics
- **The case of ontology concepts**
- Conclusion

The case of ontology concepts

- Ontology models are handled by classical PMMS
  - PLIB
  - OWL
- How about migration from a model to another ?
  - Model transformations can be expressed in a PMMS
- How about non canonical concepts ?
  - Union, restriction, …
- Reasonners are set up
  - Racer, Pellet, …
- Non canonical concepts can be materialized in a PMMS
  - ❖ Structures
  - ❖ Instances

The case of ontology concepts
Model transformation PLIB2OWL

- Introduction
- Persistent MMS
- Handling behaviors …
- **The case of ontologies**
- Conclusion



The case of ontology concepts
Model transformation PLIB2OWL

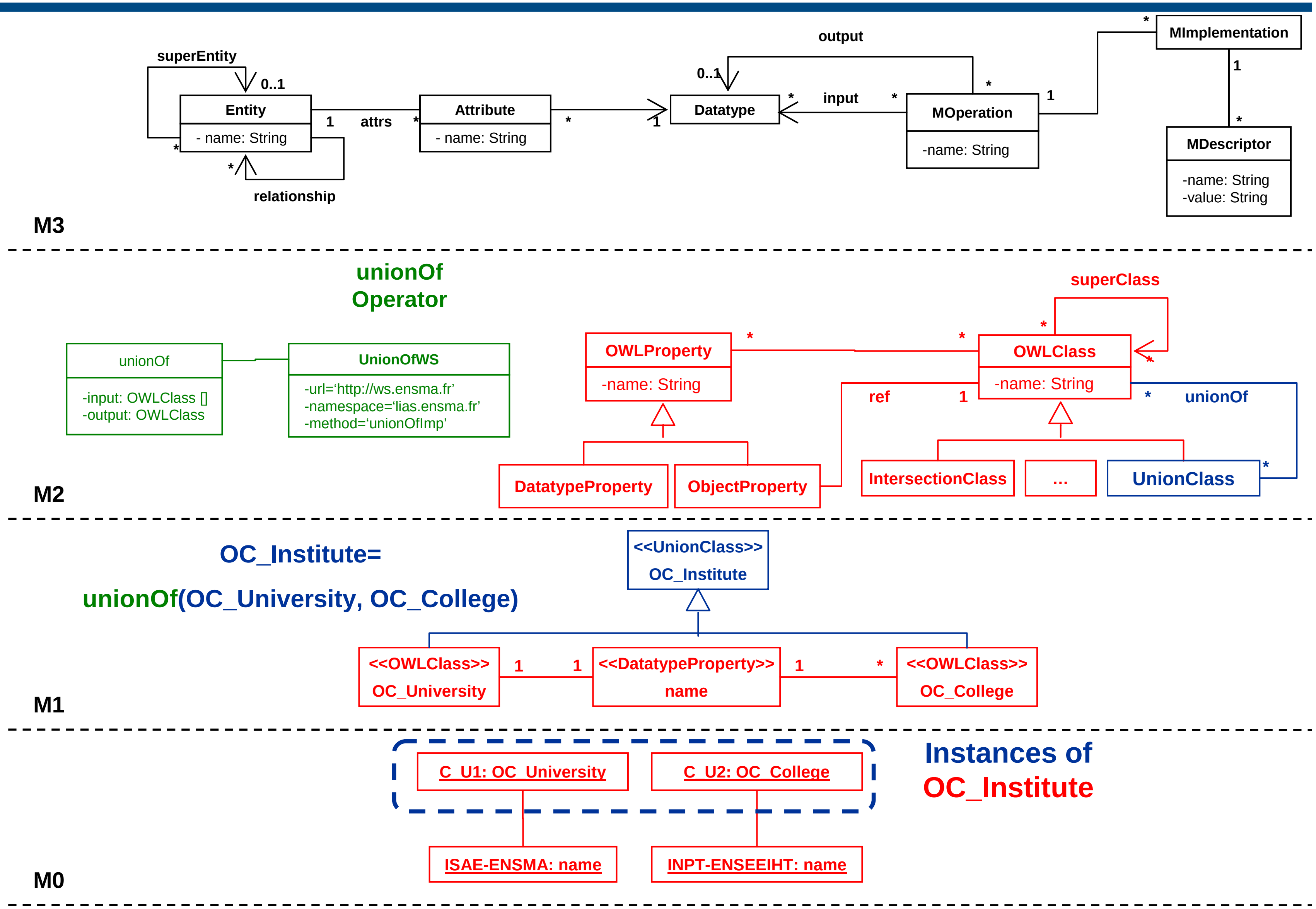
```
CREATE MOperation PLIB2OWL
INPUT REF (#PLIBClass)
OUTPUT REF (#OWLClass);
```

```
CREATE MImplementation PLIB2OWLWS
url='http://ws.ensma.fr/services'
namespace='fr.ensma.lias'
opName='Plib2OwlImp'
IMPLEMENTS PLIB2OWL;
```

```
CREATE #OWLClass OC_ University
AS PLIB2OWL (PC_University)
With Implem PLIB2OWLWS;

CREATE #Table OC_ Student
AS PLIB2OWL (PC_Student)
With Implem PLIB2OWLWS;
```

The case of ontology concepts
Non canonical concepts

- Introduction
- Persistent MMS
- Handling behaviors …
- The case of ontologies
- Conclusion



M3

superEntity
0..1
Entity
- name: String
*
*
relationship
1
attrs
*
Attribute
- name: String
*
1
Datatype
0..1
output
*
input
*
MOperation
-name: String
*
1
*
MImplementation
1
*
MDescriptor
-name: String
-value: String

M2

unionOf Operator

unionOf
-input: OWLClass []
-output: OWLClass

UnionOfWS
-url='http://ws.ensma.fr'
-namespace='lias.ensma.fr'
-method='unionOfImp'

OWLProperty
-name: String
*
*
DatatypeProperty
ObjectProperty
ref
1
superClass
*
OWLClass
-name: String
*
*
unionOf
IntersectionClass
...
UnionClass
*

M1

OC_Institute=

unionOf(OC_University, OC_College)

<<UnionClass>>
OC_Institute
<<OWLClass>>
OC_University
1
1
<<DatatypeProperty>>
name
1
*
<<OWLClass>>
OC_College

M0

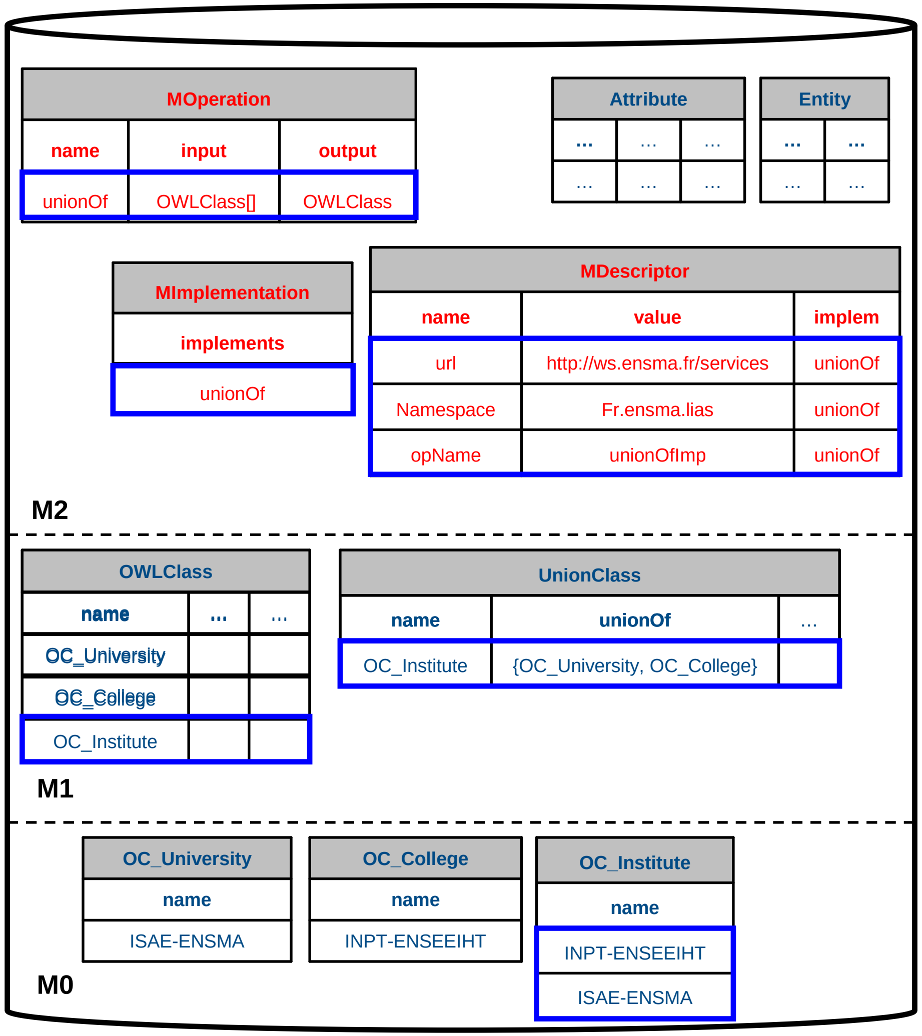
C_U1: OC_University
C_U2: OC_College
ISAE-ENSMA: name
INPT-ENSEEIHT: name

Instances of OC_Institute

The case of ontology concepts
Non canonical concepts

OC_Institute= unionOf(

OC_University, OC_College)



M2

MOperation		
name	input	output
unionOf	OWLClass[]	OWLClass

Attribute		
…	…	…
…	…	…

Entity	
…	…
…	…

MImplementation
implements
unionOf

MDescriptor		
name	value	implem
url	http://ws.ensma.fr/services	unionOf
Namespace	Fr.ensma.lias	unionOf
opName	unionOfImp	unionOf

M1

OWLClass		
name	…	…
OC_University		
OC_College		
OC_Institute		

UnionClass		
name	unionOf	…
OC_Institute	{OC_University, OC_College}	

M0

OC_University
name
ISAE-ENSMA

OC_College
name
INPT-ENSEEIHT

OC_Institute
name
INPT-ENSEEIHT
ISAE-ENSMA

The case of ontology concepts
Non canonical concepts

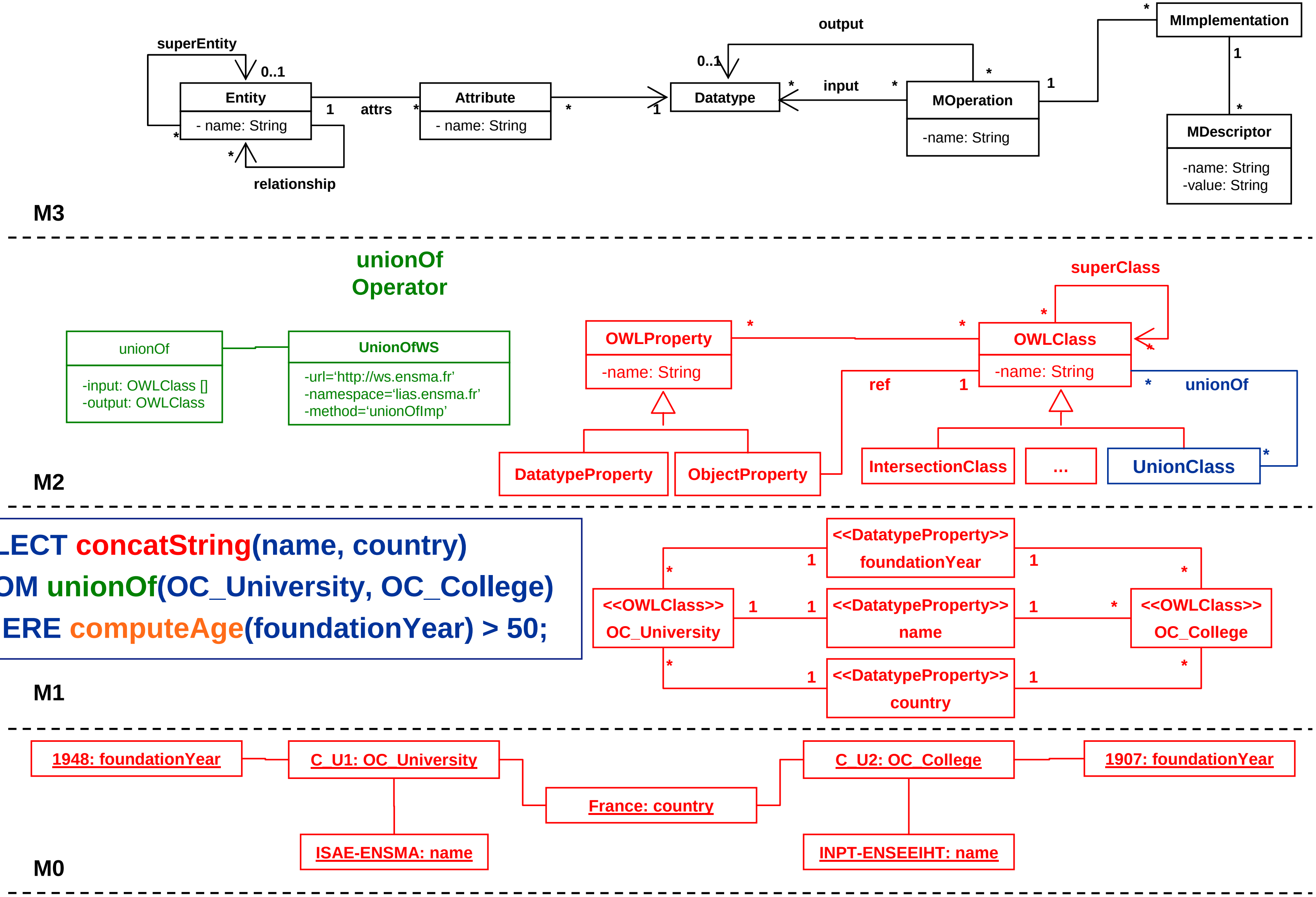
- Introduction
- Persistent MMS
- Handling behaviors …
- The case of ontologies
- Conclusion

```
CREATE MOperation unionOf
INPUT REF (#OWLClass) ARRAY
OUTPUT REF (#OWLClass);
```

```
CREATE MImplementation unionOfWS
url='http://ws.ensma.fr/services'
namespace='fr.ensma.lias'
opName='unionOfImp'
IMPLEMENTS unionOf;
```

```
CREATE #OWLClass OC_ Institute
AS unionOf (PC_University, PC_College)
With Implem unionOfWS;
```

The case of ontology concepts
Non canonical concepts



```
SELECT concatString(name, country)
FROM unionOf(OC_University, OC_College)
WHERE computeAge(foundationYear) > 50;
```

Conclusion

- The PMMS can be seen as a Model Store
  - Other examples
    - AADL to MARTE
    - BPEL models
    - DB view computations
- Extension of OntoQL to handle behavioral semantics
  - Implementation of the Algebra of operators for behaviors
- Power of programming languages
- Heterogeneous languages
- Towards a notion of MOTS

  Models on the shelf

Demonstration

- Video Demonstration

Future directions

- Definition of new interpretations for other programming languages
- Model evolutions
- Model analyses
- Formal validation of the proposed approach

Thank you !

Thank you
for
your attention

yamine@n7.fr