Extending Persistent Meta-Modeling Systems to Handle Behavioral Semantics

Yamine AIT AMEUR\textsuperscript{1} and Youness BAZHAR\textsuperscript{2}

\textsuperscript{1}IRIT/ENSEEIHT, \textsuperscript{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,
  - ...
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
Introduction

The MOF
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
  - ...

---

Extending persistent meta-modelling systems to handle behavioural semantics

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

Yamine AIT AMEUR and Youness BAZHAR

– Constantine  November 10th, 2012

Persistent MMS

An illustrative example

Meta Meta-Model layer (M3)

Meta-Model layer (M2)

Model layer (M1)

Instance layer (M0)

Introduction

Persistent MMS

Handling behaviors …

The case of ontologies

Conclusion

Entity

- name: String

Attribute

- name: String

Datatype

superEntity

0..1

relationship

1

attrs

1

Datatype

superClass

0..1

-class

-name: String

-isPersistent: Boolean

Datatype

C_University

C_Student

firstName='Dupond'
lastName='Durant'
birthday='21/06/1986'

C_U1: C_University

name='ISAE-ENSMA'

C_S1: C_Student

firstName='Dupond'
lastName='Durant'
birthday='21/06/1986'
An illustrative example

**Persistent MMS**

- Introduction
- Persistent MMS
- Handling behaviors ...
- The case of ontologies
- Conclusion
Persistent MMS
An illustrative example
The OntoQL exploitation language

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

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

CREATE #Class C_University Properties (name STRING);

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

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

INSERT INTO C_Student
(firstName, lastName, birthDay)
VALUES ('Dupond', 'Durand', '21/06/1986');
Extending persistent meta-modelling systems to handle behavioural semantics
Yamine AIT AMEUR and Youness BAZHAR – Constantine November 10th, 2012

Persistent MMS
Instantiation for the PLIB B model

M3
superEntity

0..1

1

attr

relationship

M2
superClass

0..1

PLIBClass

,name: String

PLIBProperty

,name: String

M1
<<PLIBClass>>

PC_Student

,*

1

<<PLIBClass>>

PC_University

,-firstName: String

,-lastName: String

,-birthday: Date

M0
C_U1: PC_University

name='ISAE-ENSMA'

C_S1: PC_Student

firstName='Dupond'

lastName='Durant'

birthday='21/06/1986'

Introduction
Persistent MMS
Handling behaviors ...
The case of ontologies
Conclusion
Persistent MMS
Instantiation for the OWL model

- Introduction
- Persistent MMS
- Handling behaviors ...
- The case of ontologies
- Conclusion
Perspective 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, #properties
FROM #OWLClass

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

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.
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)

...
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

- 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
  ◇ Constraints
  ◇ …

◆ 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

Class2Table (C_Student)
Class2Table(C_University)

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

C_U1: C_University
name='ISAE-ENSMA'

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

T_U1: T_University
name='ISAE-ENSMA'

M1

M2

M3

Class2Table.java
-url=http://ws.ensma.fr/
-namespace=lia.ensma.fr'

Class2TableWS
-located='D:1/
-class=JavaOp'
-package=fr.ensma.ilia'
-method=class2table'

- Entity
  - name: String

- Attribute
  - name: String

- Datatype

- MOperation
  - name: String

- MDescriptor
  - name: String
  - value: String

- MImplementation
  - name: String

- Entity
  - name: String

- Attribute
  - name: String

- Operation

- Descriptor

- Implementation

- Entity
  - name: String

- Attribute
  - name: String

- Datatype

- MOperation
  - name: String

- MDescriptor
  - name: String
  - value: String

- MImplementation
  - name: String

- Entity
  - name: String

- Attribute
  - name: String

- Class
  - name: String
  - isPersistent: Boolean

- Property
  - name: String

- Column
  - name: String

- Table
  - name: String

- MOperation
  - name: String

- MDescriptor
  - name: String
  - value: String

- MImplementation
  - name: String
Handling behavioral semantics
Class or model operations

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

---

**M0**

<table>
<thead>
<tr>
<th>Class</th>
<th>name</th>
<th>firstName</th>
<th>lastName</th>
<th>birthDay</th>
</tr>
</thead>
<tbody>
<tr>
<td>C_Student</td>
<td>ISAE-ENSMA</td>
<td>Durand</td>
<td>Dupond</td>
<td>21/06/1986</td>
</tr>
</tbody>
</table>

**M1**

- **Column**
  - name: String
  - itsType: T_University
  - itsTable: T_Student

- **Class**
  - Table: T_Student
  - Property: T_University

- **M2**

- **MOperation**
  - name: class2Table
  - input: Class
  - output: Table

- **MDescriptor**
  - name: Location
  - value: D:\
  - implem: class2Table

- **MImplementation**
  - implements: MOperation

- **class2Table transformation**
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;
Extending persistent meta-modelling systems to handle behavioural semantics

Yamine AIT AMEUR and Youness BAZHAR

IWAISE 2012

Handling behavioral semantics
Instance operations

Constraint:
Operation with a boolean output

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'

M1

Compute derivations

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

C_University
- name: String

M2

superClass

Class
- name: String
- isPersistent: Boolean

Property
- name: String

Datatype
- name: String

Implementation
- name: String
- value: String

Descriptor
- name: String
- value: String

Constraint:
Operation with a boolean output

Operation
- name: String
- isStatic: Boolean

M3

superEntity

Entity
- name: String

Attribute
- name: String

Datatype
- name: String

MImplementation
- name: String
- value: String

MDescriptor
- name: String
- value: String

M0

M1

M2

M3

Introduction
Persistent MMS
Handling behaviors ...
The case of ontologies
Conclusion
Extending persistent meta-modelling systems to handle behavioural semantics

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

IWAISE  2012

Handling behavioral semantics
Instance operations

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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Entity</th>
<th>MOperation</th>
<th>MDescriptor</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>name</td>
<td>Operation</td>
<td>Descriptor</td>
</tr>
<tr>
<td>type</td>
<td>superEntity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>itsEntity</td>
<td>Operation</td>
<td>Implement</td>
<td>Descriptor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operation</th>
<th>Descriptor</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>value</td>
<td>implem</td>
</tr>
<tr>
<td>input</td>
<td></td>
<td></td>
</tr>
<tr>
<td>output</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C_University</th>
<th>C_Student</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>firstName</td>
</tr>
<tr>
<td>ISAE-ENSMA</td>
<td>Dupond</td>
</tr>
</tbody>
</table>
Handling behavioral semantics

Instance operations

CREATE #Operation concatString
INPUT STRING ARRAY
OUTPUT STRING;

CREATE #Operation computeAge
INPUT STRING,
OUTPUT STRING;

CREATE #Operation isAdult
INPUT STRING,
OUTPUT BOOLEAN;

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 isAdultImp(s.birthday) = TRUE;

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;
Extending persistent meta-modelling systems to handle behavioural semantics

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

Handling behavioural semantics
Prototype

Support of structural and descriptive semantics

Model repository

Exploitation language

Behavior API

Java

WS

…

Introduction
Persistent MMS
Handling behaviors ...
The case of ontologies
Conclusion

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

Exploitation language extension:
- Operations definition
- Embedded in the exploitation language
- Data mappings (between datatypes of PMMS and implementations)

Behavior API:
- Embedded in the exploitation 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, …

- Reasoners are set up
  - Racer, Pellet, …

- Non canonical concepts can be materialized in a PMMS
  - Structures
  - Instances
Extending persistent meta-modelling systems to handle behavioural semantics

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

IWAISE  2012

40

The case of ontology concepts

Model transformation PLIB2OWL

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

PLIB2OWL

Transformation

superEntity

Entity

- name: String

1

relationship

Attribute

- name: String

0..1

Datatype

0..1

output

input

MOperation

- name: String

MDescriptor

- name: String

OWLClass

- name: String

ObjectProperty

UnionClass

IntersectionClass

OWLProperty

- name: String

PLIB2OWL

- url='http://ws.ensma.fr'
- namespace='lias.ensma.fr'
- method='Plib2OwlImp'

<<PLIBClass>>

PC_University

- name: String

<<PLIBClass>>

PC_Student

- firstName: String
- lastName: String
- birthday: Date

<<PLIBClass>>

C_S1: PC_Student

firstName='Dupond'
lastName='Durant'
birthday='21/06/1986'

<<PLIBClass>>

C_U1: OC_University

name='ISAE-ENSMA'

<<DatatypeProperty>>

lastName

<<DatatypeProperty>>

firstName

<<DatatypeProperty>>

itsUniversity

<<DatatypeProperty>>

birthday

<<DatatypeProperty>>

itsUnivInst

IntersectionClass

UnionClass

ref

PLIB2OWLWS

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

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

M3

unionOf
Operator

M2

OC_Institute=
unionOf(OC_University, OC_College)

M1

Instances of
OC_Institute

M0

Introduction
Persistent MMS
Handling behaviors ...
The case of ontologies
Conclusion
The case of ontology concepts
Non canonical concepts

**Introduction**

- Persistent MMS
- Handling behaviors ...
- The case of ontologies
- Conclusion

---

**M0**

- **OC_University**
  - name
  - ISAE-ENSMA

- **OC_College**
  - name
  - INPT-ENSEEIHT

- **OC_Institute**
  - name
  - unionOf
    - unionOf
      - OC_University
      - OC_College

---

**M1**

- **OWLClass**
  - name
  - OC_University
  - OC_College
  - OC_Institute

- **UnionClass**
  - name
  - unionOf
  - OC_Institute
  - (OC_University, OC_College)

---

**M2**

- **MOperation**
  - name
  - input
  - output
  - unionOf
    - OWLClass,
    - OWLClass

- **MImplementation**
  - implements
  - unionOf

---

**The case of ontology concepts**

**Non canonical concepts**

- **OC_Institute** = unionOf(OC_University, OC_College)
The case of ontology concepts
Non canonical concepts

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

M3

UnionOf
Operator

unionOf
Operator

unionOf
-Input: OWLClass
-Output: OWLClass

OWLProperty
-Name: String

OWLClass
-Name: String

IntersectionClass

UnionClass

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

M2

M1

M0

Introduction
Persistent MMS
Handling behaviors...
The case of ontologies
Conclusion
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
Extending persistent meta-modelling systems to handle behavioural semantics
Yamine AIT AMEUR and Youness BAZHAR – Constantine November 10th, 2012

Video Demonstration
Future directions

◆ Definition of new interpretations for other programming languages

◆ Model evolutions

◆ Model analyses

◆ Formal validation of the proposed approach
Thank you for your attention

yamine@n7.fr