A Framework to Formalise the MDE Foundations

Xavier Thirioux, Benoît Combemale,
Xavier Crégut and Pierre-Loïc Garoche

IRIT Institut de Recherche en Informatique de Toulouse (CNRS UMR 5505),
University of Toulouse, France
first_name.last_name@enseeiht.fr

June 25, 2007
Motivations

Expressing execution semantics
Consistency of multiple semantics
Issues

Our Framework

Intuitive approach
Formal definitions

Application to EMOF and OMG pyramid issues

EMOF Core as a Reference Model
Example of an EMOF model: PetriNet
EMOF Metacircularity
Definition of the MDA Pyramid

Conclusion & Future Works
# Plan

1. Motivations
   - Expressing execution semantics
   - Consistency of multiple semantics
   - Issues

2. Our Framework
   - Intuitive approach
   - Formal definitions

3. Application to EMOF and OMG pyramid issues
   - EMOF Core as a Reference Model
   - Example of an EMOF model: PetriNet
   - EMOF Metacircularity
   - Definition of the MDA Pyramid

4. Conclusion & Future Works
Our general goals

Context: The TOPCASED Project (http://topcased.org)
- An open source CASE environment for critical embedded systems
- Based on Model-Driven Engineering

Goals: execution semantics of models for:
- simulation
- formal verification
- code generation

Requirement
- Definition of DSL’s semantics (structural and behavioral).

Experiments on a Simplified Process Description Language (SimplePDL) and on UML 2 statemachines (work in progress).
**DSL semantics using operational semantics**

May be achieved thanks to:

- meta-programming language (kermeta, action language...)

- endogenous transformations (ATL...)

**Main advantage:** Deals with concepts related to the DSL.
DSL semantics using translational semantics

*Example*: Mapping a SimplePDL model into a time Petri net one to use the TINA toolkit.

**Advantage:** reuse the tools available in the target technical space.
Consistency of multiple semantics

Usefulness of several semantics

1. Define a reference semantics using operational semantics.
2. Define translational semantics to reuse tools in other technical spaces.

Problem

How to assert that all the defined semantics are consistent?

Our solution

Defining a framework based on formal tools like the COQ proof assistant to

1. define operational semantics of the DSL (reference semantics)
2. define operational semantics of the technical space (semantic domain)
3. express the mapping from the DSL to the semantic domain
4. prove the equivalence of translational semantics and reference semantics
Issues

- How to formally express the concepts of models, metamodels, meta-metamodels...?
  - what are their various types?
  - what is the encoding in a formal domain semantics?

- With this encoding, how to express the structural and behavioral semantics?
  - does a model conform to its language?
  - are two languages equivalent from a structural or behavioral point of view?

**Warning:** the OMG vision being one of the possible MDE view... The framework must be more general.
Content of this talk

- describe a formal framework to formalise MDE foundations
- apply it to the formalisation of a subset of EMOF Core

Warning

At the moment, only formalisation of static semantics is addressed
Plan

1. Motivations
   - Expressing execution semantics
   - Consistency of multiple semantics
   - Issues

2. Our Framework
   - Intuitive approach
   - Formal definitions

3. Application to EMOF and OMG pyramid issues
   - EMOF Core as a Reference Model
   - Example of an EMOF model: PetriNet
   - EMOF Metacircularity
   - Definition of the MDA Pyramid

4. Conclusion & Future Works
Notion of \textit{ReferenceModel}

- “A Model conforms to a ReferenceModel”
- “A ReferenceModel is a Model”
  \[ \Rightarrow \text{A ReferenceModel conforms to a ReferenceModel,} \]
  \[ \Rightarrow \text{An initial ReferenceModel has to be reflexive.} \]

Illustrated with the formalisation of KM3

Implemented in Prolog with some limitations
ReferenceModel and Model

Intuitive approach

- **ReferenceModel** (⟨concepts, relations, semantics⟩):
  - modelling language from which one can define a family of models,
  - specifies the semantic properties of its models.

- **Model** (⟨objects, links⟩): the instance level.

- A model MUST conform to a RM.
- A RM may be directly defined.
- A RM may be obtained as the promotion of a model.
Conformity and promotion
Intuitive approach

Conformity

1. Every object \( o \) in \( M \) is the instance of a class \( C \) in \( RM \);
2. Every link between two objects is such that it exists, in \( RM \), a reference between the two classes typing the two elements.
3. Every semantic property defined in \( RM \) is satisfied in \( M \).

Promotion

1. Identify the different concepts among the model elements.
2. Identify relations between the previous concepts.
3. Define the different semantic properties that must hold on the models that conform to the Reference Model.
Formal Approach
General Definitions

Let us consider:

**Definition**
- **Classes** the set of all possible classes,
- **References** the set of reference labels,
- **Objects** the set of instances of such classes.

**Definition**
- $\mathcal{C} \subseteq \text{Classes}$ be a set of classes,
- $\mathcal{R} \subseteq \{\langle c_1, r, c_2 \rangle \mid c_1, c_2 \in \mathcal{C}, r \in \text{References} \}$ be the set of references among classes where
  \[ \forall c_1 \in \mathcal{C}, r \in \text{References}, \text{card}\{\langle c_1, r, c_2 \rangle \in \mathcal{R} \} \leq 1. \]
Formal Approach
Model and Reference Model

Definition (Model)
A model $\langle MV, ME \rangle \in \text{Model}(C, R)$ is a multigraph built over a finite set $MV$ of typed objects and a finite set $ME$ of typed edges such that:

$$MV \subseteq \{\langle o, c \rangle | o \in \text{Objects}, c \in C\}$$
$$ME \subseteq \{\langle\langle o_1, c_1\rangle, r, \langle o_2, c_2\rangle\rangle | \langle o_1, c_1\rangle, \langle o_2, c_2\rangle \in MV, \langle c_1, r, c_2\rangle \in R\}$$

Definition (ReferenceModel)
A reference model $\langle (RV, RE), \text{conformsTo} \rangle$ is a multigraph built over a finite set $RV$ of classes and a finite set $RE$ of references, with semantic properties over the instances of both classes and references.

$$RV \subseteq \text{Classes}$$
$$RE \subseteq \{\langle c_1, r, c_2\rangle | c_1, c_2 \in RV, r \in \text{References}\}$$

$\text{conformsTo} : \text{Model}(RV, RE) \rightarrow \text{Bool}$
Motivations

Expressing execution semantics
Consistency of multiple semantics
Issues

Our Framework

Intuitive approach
Formal definitions

Application to EMOF and OMG pyramid issues

EMOF Core as a Reference Model
Example of an EMOF model: PetriNet
EMOF Metacircularity
Definition of the MDA Pyramid

Conclusion & Future Works
EMOF Core as a Reference Model

Traditional notation (class diagram notation)
EMOF Core as a Reference Model

Formal notation

Definition (EMOF Core)

The EMOF Core Reference Model is $\langle (RV, RE), conformsTo \rangle$ where:

$$RV \triangleq \{ \text{NamedElement}, \text{Type}, \text{TypedElement}, \text{DataType}, \text{Boolean}, \text{String}, \text{Natural}^+, \text{Class}, \text{Property} \}$$

$$RE \triangleq \{ \langle \text{Class}, \text{ownedAttribute}, \text{Property} \rangle, \langle \text{Class}, \text{isAbstract}, \text{Boolean} \rangle, \langle \text{Class}, \text{inh}, \text{Type} \rangle, \ldots \}$$

$\text{conformsTo}(\langle MV, ME \rangle) \triangleq \langle MV, ME \rangle \in \text{Model}(RV, RE) \land lower(\text{TypedElement}, \text{type}, 1) \land upper(\text{TypedElement}, \text{type}, 1) \land \text{and other semantic properties (next slide)} \ldots$
EMOF Core as a Reference Model: semantics

**Definition (Lower Property)**

\[
lower(c_1 \in RV, r_1 \in RE, n \in \text{Natural}^+) \triangleq \langle MV, ME \rangle \mapsto \forall \langle o, c \rangle \in MV, c = c_1 \Rightarrow \text{card}(\{m_2 \in MV \mid \langle \langle o, c_1 \rangle, r_1, m_2 \rangle \in ME\}) \geq n
\]

**Definition (Opposite Property)**

\[
\text{isOpposite}(r_1, r_2 \in RE) \triangleq \langle MV, ME \rangle \mapsto \forall m_1, m_2 \in MV, \langle m_1, r_1, m_2 \rangle \in ME \iff \langle m_2, r_2, m_1 \rangle \in ME
\]

**Definition (Abstract Classes)**

\[
\text{isAbstract}(r \in RE, c_1 \in RV) \triangleq \langle MV, ME \rangle \mapsto \forall \langle o, c \rangle \in MV, c = c_1 \Rightarrow \exists c_2 \in RV, \langle \langle o, c_2 \rangle, r, \langle o, c_1 \rangle \rangle \in ME
\]

And also: upper, inheritance, composite, ordered...
Example of an EMOF model: PetriNet

Class diagram graphical notation

```
PetriNet
  nodes : 0..*
  arcs : 0..*

Node
  name : String

Transition

Place
  tokenNb : Int

Arc
  source
  target
  tokenNb : Int

Arc
  source
  target
  tokenNb : Int
```

Example of an EMOF model: PetriNet

Class diagram graphical notation

```
PetriNet
  nodes : 0..*
  arcs : 0..*

Node
  name : String

Transition

Place
  tokenNb : Int

Arc
  source
  target
  tokenNb : Int
```

Example of an EMOF model: PetriNet

Object diagram notation

```
<<ownedAttribute>> <<ownedAttribute>>

: Class
name = "PetriNet"

: Property
name = "nodes"
lower = 0
upper = *
isComposite = true

: Class
name = "Node"
isAbstract = true

: Property
name = "source"

: Class
name = "Arc"

: Property
name = "target"

: Class
name = "Place"

: Property
name = "tokenNb"

: Class
name = "String"

: Class
name = "Transition"

: Property
name = "nodes"
lower = 0
upper = *
isComposite = true

: Property
name = "arcs"
lower = 0
upper = *
isComposite = true

<<type>>
<<type>>
<<type>>
<<superClass>>
<<superClass>>

Thirioux et al. (IRIT)

Formalisation of the MDE Foundations

June 25, 2007

21 / 31
Example of an EMOF model: PetriNet

Formal notation

\[ MV \triangleq \{ \langle \text{PetriNet, Class} \rangle, \]
\[ \langle \text{arcs, Property} \rangle, \]
\[ \langle \text{nodes, Property} \rangle, \]
\[ \langle \text{Node, Class} \rangle, \ldots \} \]

\[ ME \triangleq \{ \langle \langle \text{PetriNet, Class} \rangle, \text{ownedAttribute}, \langle \text{arcs, Property} \rangle \rangle, \]
\[ \langle \langle \text{arcs, Property} \rangle, \text{type}, \langle \text{Node, Class} \rangle \rangle, \]
\[ \langle \langle \text{PetriNet, Class} \rangle, \text{ownedAttribute}, \langle \text{nodes, Property} \rangle \rangle, \]
\[ \ldots \} \]
Example of an EMOF model: PetriNet

Other graphical notation

- : ownedAttribute
- : superClass
- : type
- : opposite

: Property
  name = "name"
  isAbstract = true

: Class
  name = "Node"

: Class
  name = "Arc"

: Property
  name = "source"

: Property
  name = "target"

: Class
  name = "Place"

: Class
  name = "String"

: Class
  name = "Transition"

: Class
  name = "PetriNet"

: Property
  name = "tokenNb"

: Property
  name = "transition"
  lower = 0
  upper = *
  isComposite = true

: Property
  name = "nodes"
  lower = 0
  upper = *
  isComposite = true

: Property
  name = "arcs"
  lower = 0
  upper = *
  isComposite = true

: Property
  name = "tokenNb"

Thirioux et al. (IRIT)
Formalisation of the MDE Foundations
June 25, 2007
Example of an EMOF model: PetriNet

Conformity and promotion
Validation of the EMOF Metacircularity

Structural Promotion, the EMOF Model
Validation of the EMOF Metacircularity
Structural Promotion, the EMOF (RM) graph definition

let \((MV, ME)\) be the graph of the model

**Definition (Nodes)**

The set \(RV\) of reference model vertices is defined such that,

\[
RV = \{ o.name | \langle o, Class \rangle \in MV \} 
\]

**Definition (Edges)**

The set \(RE\) is the set of edges in the reference model such that,

\[
RE \triangleq \{ \langle o_1.name, o_2.name, o_3.name \rangle \mid Rule(o_1, o_2, o_3) \} 
\]

Where,

\[
Rule(o_1, o_2, o_3) \triangleq \langle \langle o_1, Class \rangle, ownedAttribute, \langle o_2, Property \rangle \rangle \in ME \\
\land \langle \langle o_2, Property \rangle, type, \langle o_3, Class \rangle \rangle \in ME
\]
Validation of the EMOF Metacircularity

Definition of the semantics

**Definition (Lower (resp. Upper) Property)**

\[ \land \{ \text{lower}(o_1.name, o_2.name, o_2.lower) \mid \text{Rule}(o_1, o_2, o_3) \} \]

**Definition (Opposite Property)**

\[ \land \{ \text{isOpposite}(o_1.name, o_2.name) \mid \langle \langle o_1, \text{Property} \rangle, \text{opposite}, \langle o_2, \text{Property} \rangle \rangle \in ME \} \]

**Definition (Abstract Classes)**

\[ \land \{ \text{isAbstract}(o.name) \mid \langle \langle o, \text{Class} \rangle, \text{isAbstract}, \langle \text{true}, \text{Boolean} \rangle \rangle \in ME \} \]

And also: superclass, composite, ordered....
Definition of the MDA Pyramid

MOF:M
MOF:RM
<<promotionOf>>
<<conformsTo>>
UML:M UML:RM
<<promotionOf>>
<<conformsTo>>
System:M
<<conformsTo>>
Real World
M0
M1
M2
M3
metametamodel
metamodel
model
Plan

1. Motivations
   - Expressing execution semantics
   - Consistency of multiple semantics
   - Issues

2. Our Framework
   - Intuitive approach
   - Formal definitions

3. Application to EMOF and OMG pyramid issues
   - EMOF Core as a Reference Model
   - Example of an EMOF model: PetriNet
   - EMOF Metacircularity
   - Definition of the MDA Pyramid

4. Conclusion & Future Works
Conclusion & Future Works

Conclusion:

- A mathematical formalisation of the MDE framework
- Being implemented using the COQ proof assistant
- Formal definition of the conformsTo predicate and promotion operation
- Formal definition of the EMOF Core language
- Framework validation through the expression of traditional problems

Future Works:

- Expressing operational semantics (including operations)
- Proof of behavioral equivalence (bissimulation)
- Reflecting the OCL logic into our formalism
- Formalizing common operations on models such as merge, import...
Thank you for your attention...

Questions?