Relating Meta-modelling and Concrete Textual Syntax

Francisco Pérez Andrés

*(Escuela Politécnica Superior, Ingeniería Informática,
Universidad Autónoma de Madrid)*

MSDL 2006 Summer Presentations
Monday 28 August 2006
McGill University

Motivation

- Visual modeling sometimes is not the better option to represent a system:
  - Mathematical equations:
    - i. e. Physical motion of an electron inside a electromagnetic field (1-Dimension):

$$z(t)=z_0+v_{z_0}+\left(\frac{e\cdot V_0}{m_e\cdot \qquad \cdot d}\cdot\cos(\qquad\cdot t)\right)\cdot(t-t_0)-\frac{e\cdot V_0}{m_e\cdot \qquad {}^2\cdot d}\cdot\sin(\qquad\cdot t)+\frac{e\cdot V_0}{m_e\cdot \qquad {}^2\cdot d}\cdot\sin(\qquad\cdot t_0)$$

- The expressiveness of pure visual modeling languages is not always enough to represent a system:
  - **UML** uses **OCL** for constraints.

Objectives

- Define a bridge between Modelware and Grammarware which allows us:
  - Define instances of models by means of textual representations.
  - Be able to have both visual and textual representation of the same model in different views.
- So, we need a new concrete syntax for the definition of textual representation.

Problems

- Is not possible to get an isomorphism between Meta-Models and EBNF Grammars without additional information.
  - Both Grammars and Meta-models can be represented as a graphs.
  - But properties in Meta-Models contains much more information.
- Meta-Models do not include any information about textual representation of elements.
- Within Meta-Models, do not exist the concept of variable, which is the essential way to keep information on parsers.

Approaches

1. Define a specific parser for each Domain Specific Language (DSL) design by a model.
   - Model.
   - Additional information about textual representation.
2. Assuming a textual language (Modelica), define the transformations between the AST and ASG.
   - Using the –Modelica parser to obtain the Abstract Syntax Tree (AST).
   - Using the Himesis' Abstract Syntax Graph (ASG) for the representation of meta-models.

1st approach

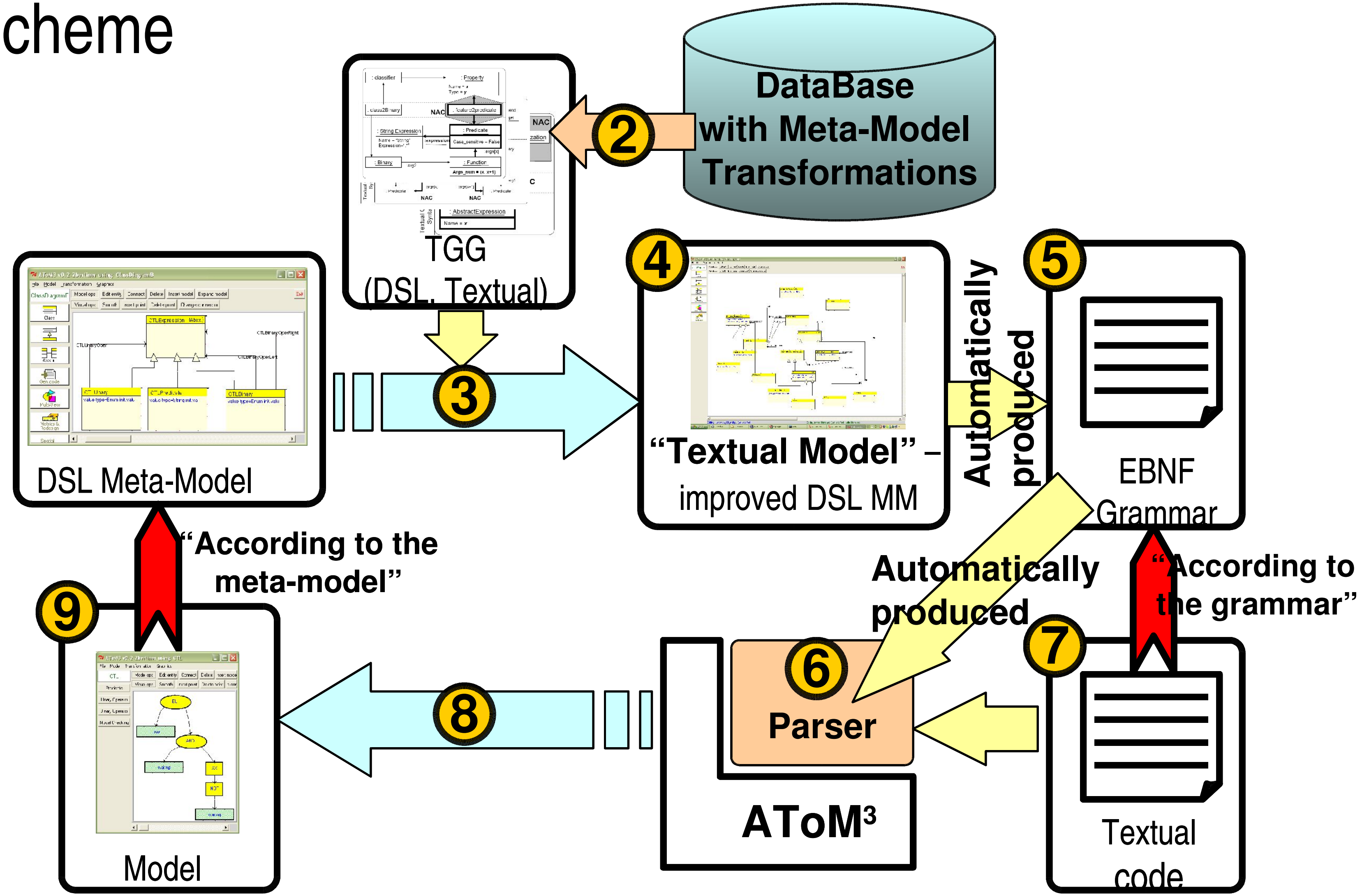
Objective

- Obtain a parser (specific for each model) which admits valid textual representations of instances of the model.

1. Enrich the model with additional textual information.
   1. Define a Meta-Model with concrete textual representation concepts and additional parsing information.
   2. Define rules to transform the DSL model to an improved model with textual information
2. Extract the EBNF Grammar from the improved model.
3. Build (semi)-automatically the parser.

1st approach

Scheme



DataBase with Meta-Model Transformations

2

TGG (DSL, Textual)

3

4

"Textual Model" – improved DSL MM

Automatically produced

5

EBNF Grammar

DSL Meta-Model

"According to the meta-model"

9

Model

8

AToM³

6

Parser

Automatically produced

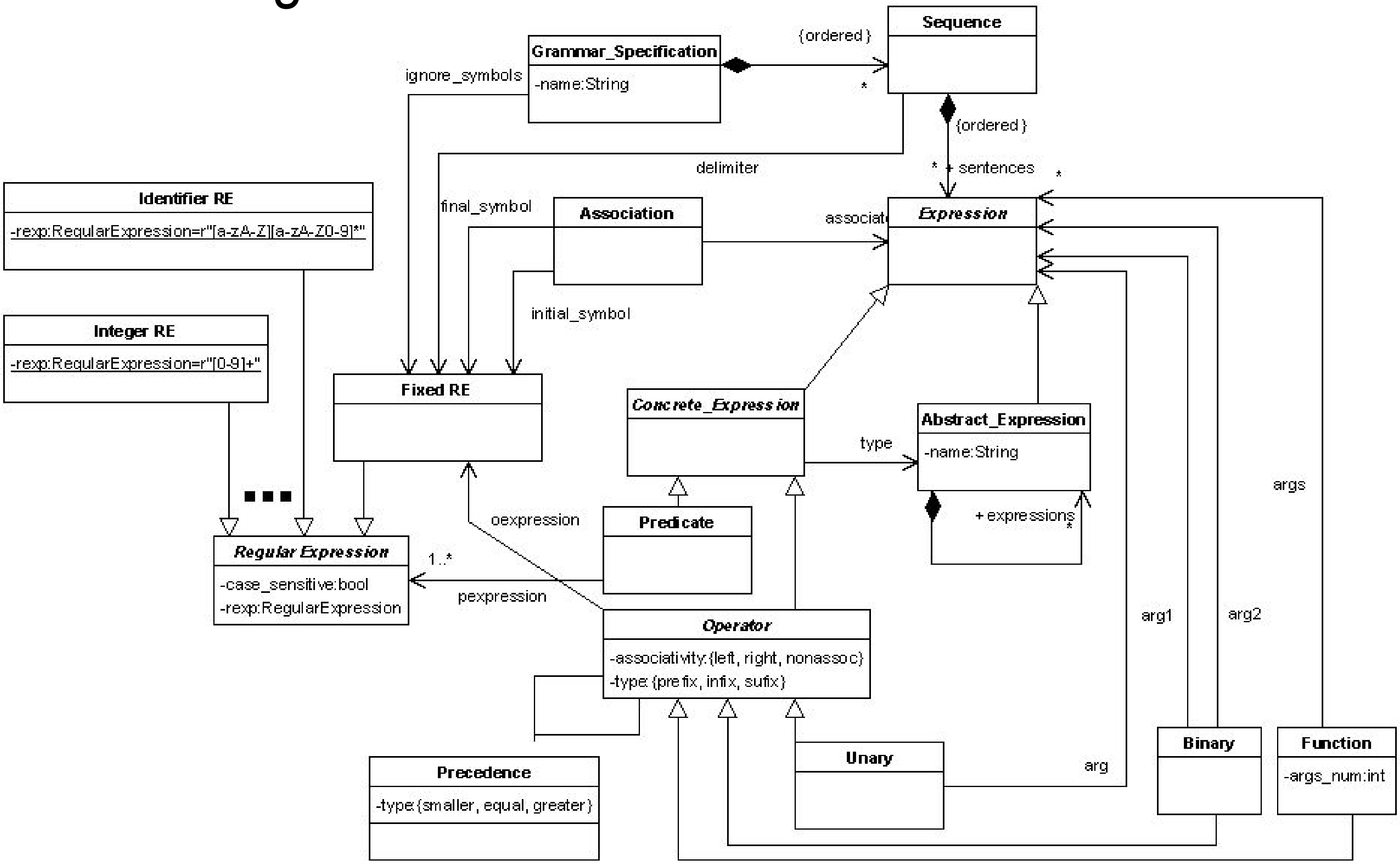
7

Textual code

"According to the grammar"

1st approach

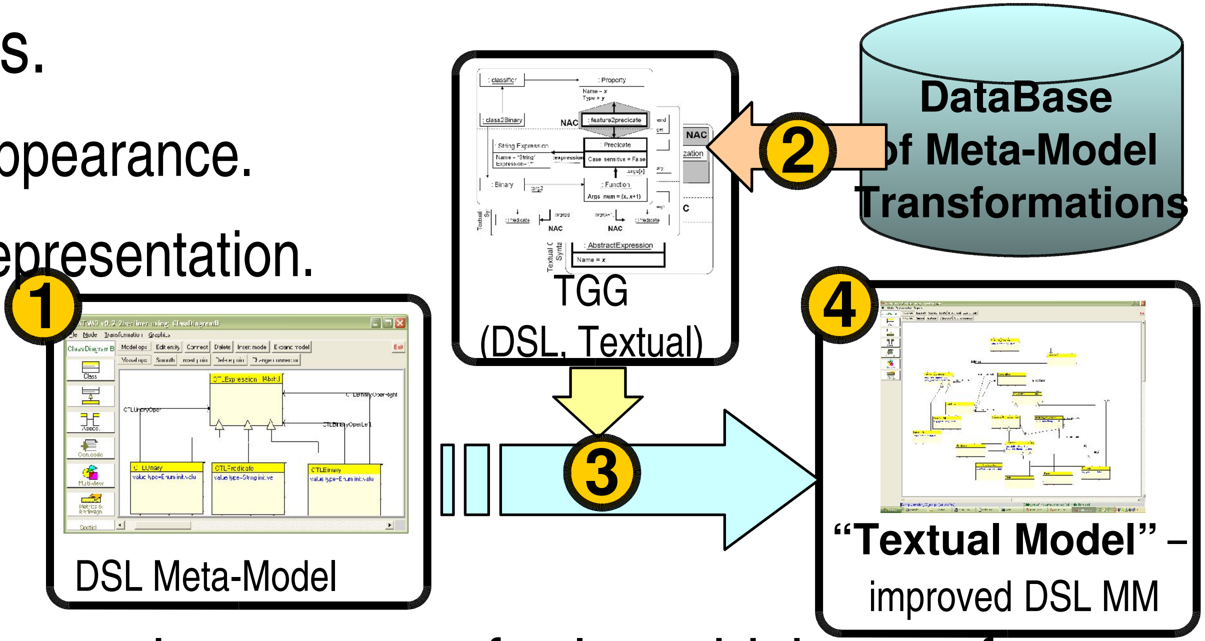
1.1.Defining the “Textual” Meta-Model



1st approach

1.2.Defining rules…

- There are several patters identified for different textual representations.
  - Expression appearance.
  - Declarative representation.



- Each of them, must have a set of rules which transform the original meta-model (1) to the "Textual Model" (4).

1st approach

1.2.Defining rules…

- Each rule is represented as a compact Triple Graph Grammar Rule:

Above: The DSL meta-model.

Among: Mappings between both meta-models

Below: The Textual Model (the improved DSL meta-model).

1st approach

1.2.Defining rules…

- After applying rules:
  - The resulting “Textual Model” can be modified.
  - The expected model shall only be a valid instance of the Textual Meta-Model.
- In fact, there are no guaranties that the model could produce a correct parser.

1st approach

2.Extracting grammar… & 3.Building parser...

- From a correct textual model, the EBNF Grammar comes out easily.
- The Parser also arises straight forward:
  - The syntactic analyzer comes from the grammar.
  - The morphologic actions to generate the ASG are Parameterized Graph Grammar Rules, which are produced by the mapping between in the Correspondence Graph.
    - This approach is based on Pair Grammars.
  - The parser is implemented in Python Lex Yacc (PLY).

1st approach

Problems

- The generated Textual Model is not an optimized model for the Textual Representation.
- Despite a good Textual Model, could be problems that hider generating the parser.
  - Problems with the basic data types.
  - Problems with identifiers.
- The applicability is restricted to simple models.

2nd approach

Assumptions

- A fixed syntax given by Modelica.
- We have a powerful parser (–Modelica) which produce Abstract Syntax Tree (AST).

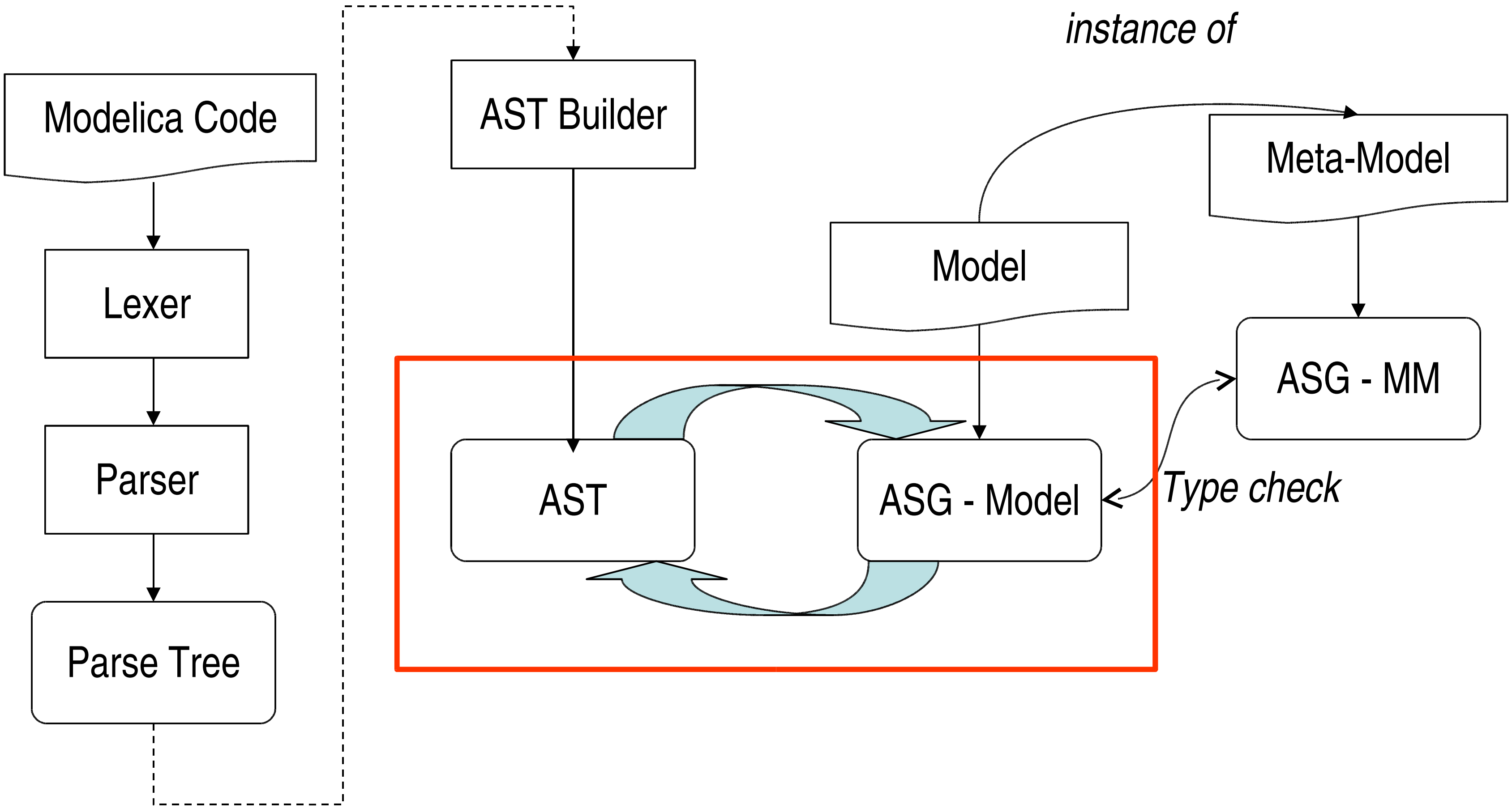
- We can depict models easily by the Abstract Syntax Graph (ASG) provided by Himesis.

2nd approach
Objectives

- Use Modelica as language to define instances of models, as the models themselves.
- Provide a mechanism to transform AST to ASG and vice-versa.
- Achieve a type-checking mechanism.
  - It is needed to check that the model is a valid instance of its meta-model.

2nd approach

Objectives



2nd approach

Improvements

- This approach focuses on the main problem: The transformation between the two abstract representations.
- Modelica syntax is a more powerful textual language that the ones we can produce in the former approach.

2nd approach
Problems

- Modelica-like textual representation loses flexibility and adaptability for simple DSL.
- Producing a valid AST, and its corresponding ASG, does not guaranties which the model was correct.
  - It is needed a extra type checking.
  - In concrete visual modelling, models are built correctly because of the meta-modeling environment.
- It could be needed additional information to accomplish the transformation.

2nd approach
Future work

- Gain flexibility on the textual representation:
  - developing specific parsers which will be able to construct the AST,
  - and could transform the AST to the specific textual representation.

Conclusions

- Formalism must combine visual and textual representation.
- Both approaches for concrete textual representation are complementary.
- The latest stress on the transformation problem, while the former gives a more whole view.
- Besides, the former goes from the meta-model outlook to the textual representation ,while the latter crosses in the opposite direction.