Comparison of Several Meta-modeling Tools 2

Yi Lu
Computer Science Department
McGill University
3.24.2003

Outline

- GME
- DiaGen
- Comparison with AToM3
- Conclusion

GME

- GME (Generic Modelling Environment) is under development of Institute for Software Integrated Systems, Vanderbilt University, US
- It’s written in C++, currently only available in Windows OS
- It’s totally free can be downloaded from http://www.isis.vanderbilt.edu/Projects/gme/Download.html

GME

- The meta-model in GME is UML-based.
- The meta-modelling language is called metaGME2000.
- The constraint language used is MCL (MultiGraph Constraint Language), which is a predicate logic language based on the Object Constraint Language.
- GME uses projects to manage models and meta-models.

GME

Meta-types provided in metaGME2000:

- **Entity: atom and model**
  - **Atom: it can’t contain other objects.**
  - **Model: it can contain other atoms, models and relationships.**
- **Relationship: three kinds of relationship.**
  - **Containment: connect the atom to the model, cardinality attribute can be defined.**
  - **Inheritance: add an Inheritance object and define its super class and subclass.**
  - **Association: add a Connector and Connection object, specify the source and the destination object for the Connector, finally connect the Connector to the Connection to define its association class.**

GME

- Aspect : not a meta-type, a unique feature of GME, to view models in a special point of view.
  - Aspects for a meta-model : Class Diagram, Visualization, Constraints and Attributes.
- Attributes: in the attribute aspect, attributes can be added to the entities and relationships.
- Constraints: in the Constraint aspect, constraints can be defined using the MCL language for specific event.
- Visualization: to specify the aspects for your modelling environment.

GME

- Interfaces to develop users’ own applications, such as simulation:
  - COM interface
  - High-level C++ component interface
  - Plug-ins

A Petri Net example

DiaGen

- DiaGen is a system for easy developing of diagram editors under development of University of Erlangen, Germany.

- It’s written in Java, platform-independent.

- DiaGen is a free software, can be download from:

  http://www2.informatik.uni-erlangen.de/DiaGen/

DiaGen

- DiaGen consists of two parts:
  - A framework of java classes which provides generic functionalities for editing and analyzing diagrams
  - Generator: which can produce Java Source code for most of the functionalities according to the specification of the diagram language.

DiaGen

- The procedures to specify a diagram editor:
  - Define the formal syntax of the diagram using the specification grammar (meta-modelling language) provided by DiaGen.
  - Generate Java Source code according to the meta-model using the generator provided by DiaGen
  - Revise these source code to implement the functionalities that your modelling environment will have
  - Define appropriate constraints for recognized structural relationships that preserve them when the diagram is modified.
  - Generate the final modelling environment of the diagram using the editor functionality provided by DiaGen

DiaGen

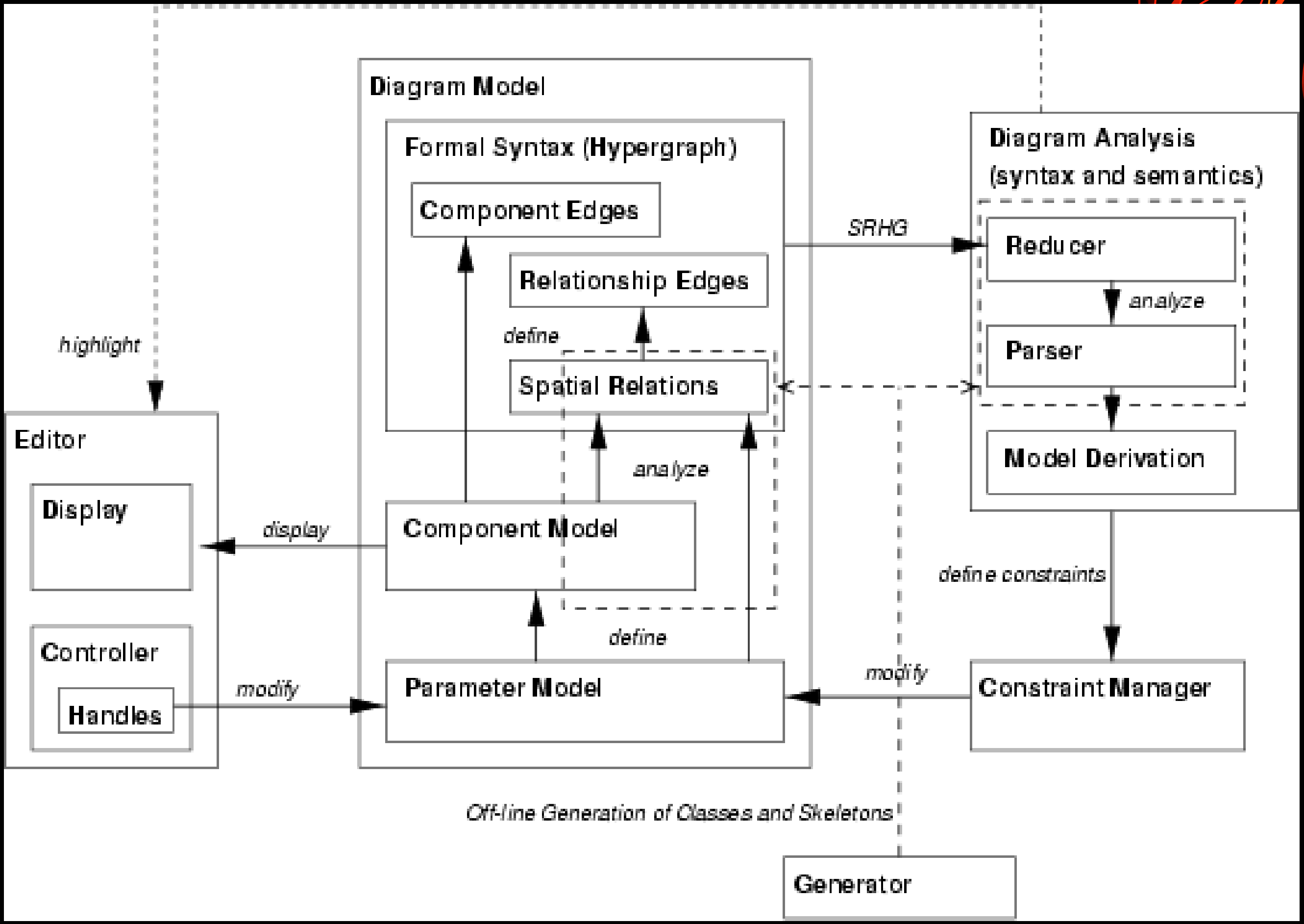
- The diagram model in DiaGen is divided into three layers
  - Parameter model: simple real numbers that determine the properties of the diagram components.
  - Component model: describes how the graphic representation of the diagram is computed from the parameters and updates the graphic representation when the parameters change.
  - Formal syntax level (SRHG): the representation of those components in the formal hypergraph syntax.

DiaGen

- The SRHG structure:

example

DiaGen



highlight
Diagram Model
Formal Syntax (Hypergraph)
Component Edges
Relationship Edges
define
Spatial Relations
analyze
Component Model
define
Parameter Model
Editor
Display
display
Controller
Handles
modify
SRHG
Diagram Analysis
(syntax and semantics)
Reducer
analyze
Parser
Model Derivation
define constraints
modify
Constraint Manager
Off-line Generation of Classes and Skeletons
Generator

DiaGen

- Petri Net example, the specification of the diagram:
  - Build the SRHG, to declare:
    - Components specify what entities will appear: circle, box, arrow, token.
    - Special relations declare the relationships among the components: inside, belongto
  - Transform SRHG to HGM, to declare (reducing) :
    - Terminal edges : place, transition, preArc, postArc
  - Transform HGM to a more simple format (graph parser):
    - Non-terminal edges: Net, places, transitions
  - Operations: complex editing operations

The meta-model of Petri Net

DiaGen

- To build an editor for the diagram (modelling environment)
  - Need to be familiar with the interface of the editor in DiaGen.
  - combine the standard editor provided by DiaGen with the user's customized specifications
  - This part should be coded totally manually.
- Can add customized functionalities, such as simulation, all code by hand.

Petri Net Example

Comparison AToM3 with MetaEdit+ and DOME

Aspects	AToM3	GME	DiaGen
Platforms	Windows, Unix	Windows	Platform-independent
Meta-modeling language	ER	metaGME2000	Specification grammar
Graphical specification?	Yes	Yes	No
Hierarchy	Partly, not implement complete yet	Yes, containment relationship	Yes
Inheritance	No	Yes	Yes
Constraint	Python function or OCL	MCL (subset of OCL)	No specific constraint language
Simulation	Yes	Yes	Yes
Simulation method and implementation workload	Graph Grammar, an intuitive way, less code by hand	COM interface, high level C++ component, plug-ins	Java, all code by hand
Report generation	No	No	No

Conclusion

The best points for these tools:

- **In AToM3, simulation is easy to implement (Graph Grammar).**
- **MetaEdit+ support well the report generation.**
- **DOME and GME support the best customization of the modelling environment.**
- **DOME and GME implement more clear and complete meta-modelling language.**
- **GME implement the constraint language best.**
- **DiaGen uses constraints to automatically adjust and manage the graphical appearance of models.**