MDE and SLE: From Theory to Practice
An experience report on scientific (farming!) models

Benoit Combemale (Inria & Univ. Rennes 1)
http://people.irisa.fr/Benoit.Combemale
benoit.combemale@irisa.fr
@bcombemale

Jean-Michel Bruel (Univ. Toulouse)
http://jmb.c.la
bruel@irit.fr
@jmbruel
Outline

1. A short reminder about MDE, DSL and SLE – BC
2. Use case: Farming Modeling – BC
3. Use case: Demonstration as external DSL – BC
4. Use case: Demonstration as UML Profile – JMB
5. Systems Engineering with Sysml – JMB
"Perhaps surprisingly, the majority of MDE examples in our study followed domain-specific modeling paradigms »

Multiple Concerns

Avionics
Aerodynamics
Mechanical Structure
Propulsion System
Communications
Navigation
Airlines
Human-Machine Interaction
Environmental Impact
Safety Regulations
Authorities
Heterogeneous Modeling
Domain-Specific Languages (DSLs)

- Targeted to a **particular** kind of problem, with dedicated notations (textual or graphical), support (editor, checkers, etc.)
- Promises: more « efficient » languages for resolving a set of specific problems in a domain
Metamodeling

Software Language Engineering (SLE)

• Application of systematic, disciplined, and measurable approaches to the development, use, deployment, and maintenance of software languages

• Supported by various kind of "language workbench"
  - Eclipse EMF, xText, Sirius, GEMOC, Papyrus
  - JetBrains’ MPS
  - MS DSL Tools
  - Etc.

• Various shapes and ways to implement software languages
  - External, internal or embedded DSLs, Profile, etc.

• More and more literature, a dedicated Intl. conference (SLE, cf. http://www.sleconf.org)
Application Domains

• Initially motivated by industry in complex embedded, critical and/or real-time systems

• Now widely used in most domains of software and systems engineering (home automation, internet of things, adaptive systems…)

• And… what about beyond?


See also the Sustainability workshop at Modularity 2015
Farming Modeling
Description and requirements

Benoit Combemale (Inria & Univ. Rennes 1)
http://people.irisa.fr/Benoit.Combemale
benoit.combemale@irisa.fr
@bcombemale

Jean-Michel Bruel (Univ. Toulouse)
http://jmb.c.la
bruel@irit.fr
@jmbruel
Farming Modeling: Description of the Use Case

1. 10-page document introducing the wide spectrum of the scientific fields (incl., 8 application domains: crop, beef/lamb, farming exploitation, water, city, biodiversity, economics)

2. 2-page document detailing the farming exploitation use case

3. 3h video conference with INRA (H. Raynal)

See all materials at: http://github.com/jmbruel/idm2014
Farming Modeling: Description of the Use Case

- Structural description of an exploitation
  - 3 workshops (crop, ovine and bovine)
  - Resources (human and equipment)
  - Surface area
- Functional (/behavioral) description of an exploitation
  - Activities and (some examples of) business rules for each workshop
- Expected outcomes:
  - Domain-specific modeling
  - Domain-specific analysis (constraint satisfaction, simulation… )
Farming Modeling: Experimentation Achieved

- Modeling and analysis thanks to a set of external DSLs
  - Tooling: EMF, xText, Sirius and GEMOC
  - Collaboration INRIA / Obeo

- Modeling and analysis thanks to a UML profile
  - Tooling: EMF and Papyrus
  - Collaboration IRIT / CEA
Farming Modeling
An Experience Report With EMF, Sirius, xText and GEMOC

Benoit Combemale (Inria & Univ. Rennes 1)
http://people.irisa.fr/Benoit.Combemale
benoit.combemale@irisa.fr
@bcombemale

With the help of Cédric Brun (CTO, Obeo)
@bruncedric
Farming Modeling: metamodeling approach

<table>
<thead>
<tr>
<th>Language Engineers</th>
<th>Language Users</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Domain</strong></td>
<td><strong>Data</strong></td>
</tr>
<tr>
<td><strong>Viewpoint</strong></td>
<td><strong>Views and static checking</strong></td>
</tr>
<tr>
<td>(graphical editor)</td>
<td>(textual editor)</td>
</tr>
<tr>
<td><strong>Grammar</strong></td>
<td><strong>Textual editing and static checking</strong></td>
</tr>
<tr>
<td>(textual editor)</td>
<td></td>
</tr>
<tr>
<td><strong>Behavioral</strong></td>
<td><strong>Globalization, execution, simulation and animation</strong></td>
</tr>
<tr>
<td>semantics</td>
<td></td>
</tr>
<tr>
<td>( animator)</td>
<td></td>
</tr>
</tbody>
</table>
Farming Modeling: metamodeling approach

<table>
<thead>
<tr>
<th>Language Engineers</th>
<th>Language Users</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>emf</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Domain</strong></td>
<td><strong>Data</strong></td>
</tr>
</tbody>
</table>
What is it?

- **Meta** Modeling (think of UML/OCL)
- Interoperability (think of XMI)
- Editing tool support (think Eclipse)
- Code generation (think of MDA)

EMF serves as the foundation: It provides the Ecore meta-metamodel, and frameworks and tools around it for tasks such as:

- Editing
- Transactions
- Validation
- Query
- Distribution/Persistence (CDO, Net4j, Teneo)

See [http://www.eclipse.org/modeling/emf](http://www.eclipse.org/modeling/emf)
• Ecore is an implementation proposed by EMF, and aligned to EMOF

• Provides a language to build languages

• A metamodel is a model; and its metamodel is Ecore.
  • So a metamodel is an Ecore model!

• Ecore has concepts like:
  • Class – inheritance, have properties
  • Property – name, multiplicity, type

• Essentially this is a simplified version of class modeling in UML
Supported by a lot of (meta) tools (e.g., graphical editor, code generator...)

### Farming Modeling: metamodeling approach

<table>
<thead>
<tr>
<th>Language Engineers</th>
<th>Domain</th>
<th>Viewpoint (graphical editor)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language Users</td>
<td>Data</td>
<td>Views and static checking</td>
</tr>
</tbody>
</table>

- **EMF**: Eclipse Modeling Framework
- **Sirius**: Graphical editor for modeling
Concevoir simplement et rapidement des ateliers de modélisation sur-mesure.
Sirius: Principles

Envirionnement de spécification

(Outiller)

Runtime

(Utisateur final)
Sirius: Principles

Define the Domain Model

Business Vocabulary
- Concepts
- Relations
- Properties

Describe the Graphical Designer

Representations
- Displayed elements
- Shapes
- Colors
- Fonts

Palette
- Buttons
- Icons

Model-Driven Tools
- Generation
- Validation
- Comparison
- Transformation

Leverage the Models

 előző kötet

új kötet
Sirius: Examples of Viewpoints
## Farming Modeling: metamodeling approach

<table>
<thead>
<tr>
<th>Language Engineers</th>
<th>Language Users</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>emf</strong></td>
<td><strong>Data</strong></td>
</tr>
<tr>
<td><strong>Sirius</strong></td>
<td><strong>Views and static checking</strong></td>
</tr>
<tr>
<td><strong>Xtext</strong></td>
<td><strong>Textual editing and static checking</strong></td>
</tr>
</tbody>
</table>

- **Domain** (graphical editor)
- **Grammar** (textual editor)
Farming Modeling with EMF, xText, Sirius and GEMOC
Give me a grammar,

I’ll give you (for free)

• a comprehensive editor (auto-completion, syntax highlighting, etc.) in Eclipse
• an Ecore metamodel and facilities to load/serialize/visit conformant models (Java ecosystem)
• extension to override/extend « default » facilities (e.g., checker)
Farming Modeling with EMF, xText, Sirius and GEMOC
Give me a **metamodel**,

I’ll give you (for free)

- a comprehensive editor (auto-completion, syntax highlitening, etc.) in Eclipse
- a grammar and facilities to load/serialize/visit conformant models (Java ecosystem)
- extension to override/extend « default » facilities (e.g., checker)
Farming Modeling with EMF, xText, Sirius and GEMOC

- Grammar
  - conforms To
  - Source Code A
  - conforms To
  - Model A

- Metamodel
  - conforms To
## Farming Modeling: metamodeling approach

<table>
<thead>
<tr>
<th>Language Engineers</th>
<th>Domain</th>
<th>Viewpoint (graphical editor)</th>
<th>Grammar (textual editor)</th>
<th>Behavioral semantics (animator)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language Users</td>
<td>Data</td>
<td>Views and static checking</td>
<td>Textual editing and static checking</td>
<td>Globalization, execution, simulation and animation</td>
</tr>
</tbody>
</table>

**Tools:**
- **EMF**: Eclipse Modeling Framework
- **Sirius**: A graphical modeling tool
- **Xtext**: A textual modeling tool
- **Gemoc**: A framework for model animation
• Breathe life into your DSLs
  • Operational and translational semantics
  • Modular, explicit and formal model of computation (e.g. DEVS)
  • Explicit behavioral language interface

• Coordinate your multiple DSLs
  • Edition, execution, simulation and animation of, possibly heterogeneous, models
Globalization of Modeling Languages

Challenge:

• DSMLs are developed in an independent manner to meet the specific needs of domain experts,

• DSMLs should also have an associated framework that regulates interactions needed to support collaboration and work coordination across different system domains.

Supporting coordinated use of modeling languages leads to what we call the globalization of modeling languages, that is, the use of multiple modeling languages to support coordinated development of diverse aspects of a system.

Globalization of Modeling Language

- Context: new emerging DSML in open world
  ⇒ impossible \textit{a priori} unification
  ⇒ require \textit{a posteriori} globalization

- Objective: socio-technical coordination to support interactions across different system aspects
  ⇒ Language-based support \textit{for technical integration} of multiples domains
  ⇒ Language-based support \textit{for social translucence}

GEMOC: The Studio

Design and compose your executable DSMLs

Edit, simulate and animate your heterogeneous models

http://gemoc.org/studio

Language Workbench

Modeling Workbench
GEMOC: The French ANR Project
Grant #ANR-12-INSE-0011 (01.12.12 – 30.03.16)

Focus: concurrent execution of behavioral heterogeneous models of complex software-intensive systems (=> systems engineering)

Breakthroughs:
- modular and explicit definition of the behavioral semantics of modeling languages, incl. concurrency [APSEC’12, SLE’12, SLE’13]
- explicit behavioral interface of modeling languages [GEMOC’13]
- integration of modeling languages for heterogeneous model coordination [Computer’14]

Visit http://gemoc.org/ins
Capella

Operational Analysis Model
What the users of the system need to accomplish

System Functional and Non-Functional Need Model
What the system has to accomplish for the users

Logical Architecture Model
How the system will work in order to fulfil expectations

Physical Architecture Model & Product Breakdown
How the system be developed and built

Capella

Capella

GEMOC Use Case: xCapella

Capella DSL

- Reuse Capella Abstract Syntax
- Reuse Capella Concrete Syntax

Operational semantic

- Define Domain Specific Action
- Define Domain Specific Event
- Define and reuse MoCC
- Define Animation
GEMOC Use Case: xCapella

```
package capellacommon

import capellacommon.StateMachineAspect

metamodel Capella {
  ecore "platform:/resource/org.polarsys.capella.core.data.gen/model/CapellaModeller.ecore"
  exactType CapellaMT
}

metamodel xCapella inherits Capella {
  resource EMF uri "http://com.thalesgroup.mde.xCapella"
  exactType xCapellaMT
  aspect StateMachineAspect
  aspect AbstractStateAspect
  aspect RegionAspect

  aspect FunctionalExchangeAspect
  aspect PhysicalComponentAspect
  aspect PhysicalFunctionAspect
  aspect FunctionPortAspect
  aspect FunctionInputPortAspect
  aspect FunctionOutputPortAspect
```
Farming Modeling???
Experiments

- DSLs for farming modeling
  - Focus: edition and animation
  - Collaboration INRIA (B. Combemale) and Obeo (C. Brun)
  - Large leeway!

- Organization:
  - 3h video-conference INRIA/IRIT/INRA (H. Raynal)
    - + 2-page description of the domain + examples
  - 3h meeting INRIA/Obeo
  - 10h distributed work INRIA/Obeo through the github repository
    - including the POC, and the preparation of the demo and slides!
  - 2h video-conference INRIA/Obeo

⇒ 26 hours of work!
Demonstration

• Farming modeling with EMF, Sirius, xText and GEMOC

• All materials (source, documentation) available at https://github.com/jmbruel/idm2014/tree/master/contri/gemoc
  • Source:
    • Language workbench (Farming DSL):
    • Modeling workbench (Examples):
  • Documentation:
## Farming Modeling: metamodeling approach

<table>
<thead>
<tr>
<th></th>
<th>Language Engineers</th>
<th></th>
<th>Language Users</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>emf</td>
<td>Sirius</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Domain</td>
<td></td>
<td></td>
<td>Data</td>
<td></td>
</tr>
<tr>
<td>Viewpoint (graphical editor)</td>
<td></td>
<td></td>
<td>Views and static checking</td>
<td></td>
</tr>
<tr>
<td>Grammar (textual editor)</td>
<td></td>
<td></td>
<td>Textual editing and static checking</td>
<td></td>
</tr>
<tr>
<td>Behavioral semantics ( animator)</td>
<td></td>
<td></td>
<td>Globalization, execution, simulation and animation</td>
<td></td>
</tr>
</tbody>
</table>
Demonstration: conclusion

- Explicit domain models (metamodels)
- (Structural) Integration of metamodels
- Combination of graphical and textual editors
- Model transformation (POC)
  - Operation semantics (~VM)
  - Translational semantics (~compiler)
Demonstration: perspectives

• Relevant model transformations
  • static and dynamic analysis
  • import / export

• (domain-specific) Animation with GEMOC (incl. concurrent heterogeneous models)

• Domain-specific property languages
Farming Modeling
An Experience Report With Papyrus

Systems engineering
Practice with SysML

Jean-Michel Bruel (Univ. Toulouse)
http://jmb.c.la
bruel@irit.fr
@jmbruel