Model Management and Formal Methods

James Williams
University of York
Overview

• Model-driven engineering

• Model management: Epsilon
  • Model-to-model transformation
  • Model-to-text transformation

• Integrating formal methods with MDE
  • Model Behaviour Language
Model-Driven Engineering

- Promotes models to first-class artefacts

- More than documentation

- Live entities that are amenable to automated processing
  - Validation, transformation, comparison, merging, refactoring, code generation etc.
MDE: Metamodels and models
MDE: Metamodels and models
MDE: Metamodels and models

Metamodel

Model

conforms to
MDE: Model management

• Transformation
  • Model-to-model
  • Model-to-text

• Evolution
  • Model migration

• Merging, validation, comparison, ...
MDE: Model management

- Transformation
  - Model-to-model
  - Model-to-text

- Evolution
  - Model migration

- Merging, validation, comparison, ...
MDE: Model management

- Transformation
  - Model-to-model
  - Model-to-text

- Evolution
  - Model migration

- Merging, validation, comparison, ...
MDE: Model management

- Transformation
  - Model-to-model

- Model-to-text

- Evolution
  - Model migration

- Merging, validation, comparison, ...
MDE: Model management

- Transformation
  - Model-to-model
  - Model-to-text

- Evolution
  - Model migration

- Merging, validation, comparison, ...
MDE: Petri net example
MDE: Models ≠ UML diagrams

• UML is just one modelling language

• Most domains have different abstractions/semantics

  • Domain Specific Languages (DSLs)

• Models ≠ Pictures

  • Models can be graphical or textual

  • ... or both
Epsilon

www.eclipse.org/gmt/epsilon

• A family of integrated languages and tools for MDE

- Model Migration (Flock)
- Model Refactoring (EWL)
- Model Comparison (ECL)
- M2M Transformation (ETL)
- Epsilon Object Language
- Epsilon Model Connectivity
- EMF driver
- MDR driver
- XML driver
- Z driver
- Unit Testing (EUnit)
- Model Validation (EVL)
- Model Merging (EML)
- M2T Transformation (EGL)
Epsilon: Where is it used?

- INESS: INtegrated European Signalling System (http://www.iness.eu/)
  - Using EGL to transform xUML into PROMELA

- MADES: Model Assisted Development of Embedded Systems (http://mades-project.ning.com/)
  - Using EGL, ETL and EVL. UML MARTE to HDL and CTV

- Industry: IBM, BAE Systems, NASA Jet Laboratory, Telefonica, Siemens, ...
Epsilon: Model-to-text transformation

- The Epsilon Generation Language
Epsilon: Model-to-text transformation

- The Epsilon Generation Language
Epsilon : Model-to-text transformation

• The Epsilon **Generation** Language

```html
<html>
<h1>Overview of our world</h1>

Our world has $\%$-Person.all.size($\%$) people:

<ul>
  $\%$<li>$\%$-p.name$\%$</li>$\%$
</ul>

and we know the following facts:

<ul>
  $\%$<li>$\%$-p.name$\%$ and $\%$-e.name$\%$ are friends</li>$\%$
  $\%$<li>$\%$-p.name$\%$ and $\%$-e.name$\%$ are enemies</li>$\%$
</ul>

</html>
```
Epsilon: Model-to-text transformation

- The Epsilon **Generation** Language

```html
<html>
  <h1>Overview of our world</h1>
  <ul>
    <li>Our world has 4 people:
      - George
      - Ian
      - Bob
      - Alan
    </li>
    and we know the following facts:
    <li>George and Ian are friends</li>
    <li>Ian and Bob are enemies</li>
    <li>Bob and Alan are friends</li>
    <li>Alan and George are enemies</li>
  </ul>
</html>
```
Epsilon: Model-to-model transformation

• The Epsilon Transformation Language
Epsilon: Model-to-model transformation

- The Epsilon Transformation Language
Epsilon : Model-to-model transformation

• The Epsilon Transformation Language
Epsilon : Model-to-model transformation

• The Epsilon Transformation Language
Epsilon : More...

• Other languages for:
  • Model refactoring, model merging, model comparison, model validation, model migration

• Epsilon tools:
  • EuGENia : generates graphical editors from simple definitions
  • HUTN : implementation of the OMG’s Human Usable Textual Notation
  • Concordance : indexes and monitors models for reconciliation and reporting
  • ModeLink : managing cross-model references
Integrating formal methods

- MDE has little support for formally analysing models

- Models can be constrained using OCL
  - OCL cannot describe the complete dynamic behaviour of a class diagram

- “Semi-formal” models often transformed into a formal language for analysis
  - e.g. xUML --> PROMELA, UML --> Z [Ama07], UML --> Alloy [Ana07]

- Issues: traceability, transformation validation, inefficient representation
The Model Behaviour Language

- A language for the specification and analysis of the structure and behaviour of models

- Integrated with the state-of-the-art practices in MDE
  - Model transformation
  - Code generation
  - Test-case generation

- In its infancy...!
MBL: Language overview

- State description language
  - Describes the *structure* of the model

- Event language
  - Adds explicit *behavioural* definitions to the model
  - Precondition, postcondition and body definitions

- Expression language
class Aircraft { }

class Airport {
    val Aircraft[*] landed;
    val Aircraft[*] permission;

    inv RestrictNumberLanded: landed.size() <= 20;

    event givePermissionToLand(Aircraft a) [ ] {
        permission.add(a);
    } [ ]

    event landPlane(Aircraft p) [ permission.includes(p) ]{
        landed.add(p);
    } [ ]
}
MBL: Analysis

- Events are the *transitions* between states
- Initial state specified in HUTN
- Model analysed directly (no transformation)
- Custom model checker built atop SpinJa
- Expressions evaluated using EOL
MBL: Future work

- Refinement
  - Data and event
- Event-trace reasoning
  - Temporal expressions
- Test case generation
- Code generation
- Interactive simulator
Conclusions

• Model-driven engineering
  • Models treated as first-class citizens

• Epsilon: state-of-the-art platform for model management

• The Model Behaviour Language
  • Extends structural definitions of models to include behaviour specification
  • Analysis of behavioural models
Searching for Migration Strategies

Applying search techniques to model migration
Searching for migration strategies

- Metamodel evolution can make models inconsistent

- Models need to be *migrated* to conform to the new version

- A *migration strategy* is defined to realign models
  - a sequence of model transformation

- Three categories [Ros09]:
  - Manual specification
  - Operator-based co-evolution
  - Metamodel matching
Searching for migration strategies

• Operator-based approach difficult unless used during **co-evolution**

• Migration strategy = sequence of operations

• *Search* for the strategy
  
  • Calculate the set of *applicable* operations in each state
  
  • Select (one of) the *fittest* operation and apply it to the source model
  
  • Repeat until the source matches the target
Searching for migration strategies

Calculate all possible operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>op 1</td>
<td>0.01</td>
</tr>
<tr>
<td>op 2</td>
<td>0.8</td>
</tr>
<tr>
<td>op 3</td>
<td>0.95</td>
</tr>
<tr>
<td>op 4</td>
<td>0.4</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

Rank wrt fitness

Recalculate applicable operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>op 3</td>
<td>0.95</td>
</tr>
<tr>
<td>op 85</td>
<td>0.8</td>
</tr>
<tr>
<td>op 332</td>
<td>0.8</td>
</tr>
<tr>
<td>op 34</td>
<td>0.8</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

Invalidated operations (remove)

Still applicable operations (no need to calc fitness)

New operations (add + calc fitness)

Repeat
Searching for migration strategies

• Searching for the global optima

• Start at both ends

[Diagram showing a tree structure with nodes and branches, indicating the search for global and local optima.]
Searching for migration strategies

- Optimisations
  - Weight operations based on usage statistics
  - Remove rarely used operators (reintroduce in later generations)
  - Ignore infinitely type attributes
Searching for migration strategies

• Currently experimenting with operation generation and fitness evaluation

• Will evaluate on case studies

  • Small - Petri nets

  • Large - UML1.4 -> UML 2.x
Bibliography

