SysML to UML transformation for test generation purpose

November the 16th 2010

Jonathan Lasalle
Fabrice Bouquet – Bruno Legeard – Fabien Peureux
VETESS project

- **VETESS**: verification of vehicle embedded system by automatic test generation from specifications.


- **Project members:**
  - Smartesting: editor of tooled MBT solution (Test Designer).
  - Clemessy: testing bench provider (Test In View).
  - PSA Peugeot Citroën: car manufacturer.
  - LIFC: Model-Based Testing expertise (MBT).
  - MIPS: Model Driven Engineering expertise (MDE).
Outline

- UML4MBT
- SysML4MBT
- Transformation from SysML4MBT to UML4MBT
- Experimentations
- Conclusion & future works
Class Diagram
- Static view of the system.
- Classes, associations, enumerations, class attributes and operations.

Object Diagram
- Concrete objects used to compute test cases.
- Define the initial state of the system.
- Must be an instanciation of the Class Diagram.
Dynamic view:

- OCL expressions on pre/post condition of operations.
- One Statemachine Diagram (annotated with OCL constraint).

- parallel states
- historic states
- fork and join states

Several restrictions on OCL.
Block Definition Diagram (BDD)
- Static view of the system and its environment.
- Blocks, associations, compositions enumerations, blocks attributes and operations, PORTS and SIGNALS.

Internal Block Diagram (IBD)
- Interconnection between blocks.
- Represent electrical or mechanical communications.
Dynamic view:

- OCL expressions on pre/post conditions of operations.
- One or more Statemachime Diagram(s) (annotated with OCL constraints).
  - parallel states
  - historic states
  - fork and join states

- Triggers: signal reception.
OCL
- Same restrictions than in UML4MBT.
- Addition of OCL ^ operator (signal sending).

Requirements Diagram
- Represents system requirements.
- Links requirements with model elements that satisfy them.
Algorithm outline:

1. Transformation of BDD & IBD to Class Diagram.
2. Rewriting of Requirement Diagram.
3. Translation of signal sends/receives.
4. Transformation of fork/join states to parallel states.
5. Rewriting of each composite, historic and parallel states by hierarchical stage.
SysML BDD to UML Class Diagram

- No changes: blocks (classes), associations, operations, attributes and enumerations on both.

![Diagram of SysML BDD to UML Class Diagram]

UML4MBT to UML4MBT transformation – BDD
SysML4MBT to UML4MBT transformation – Signals

SysML IBD to UML Class Diagram

Signals:
- Used on IBD.
- Defined on BDD => UML classes.
- Add a new attribute isUsed.

![UML Class Diagram with Signals](image1)

![SysML IBD Diagram with Signals](image2)
SysML IBD to UML Class Diagram

- **Ports:**
  - Used on IBD.
  - Defined on BDD.
  - Which signal is pending on which port.

Block B
Port p which can receive Sig1 & Sig2
SysML4MBT to UML4MBT transformation – Requirements

 SysML Requirement Diagram to OCL

- OCL for MBT: expression of requirements:
  /**@REQ: description of the requirement*/

- For each requirement of diagram
  - If satisfied by a transition: OCL added to effect.
  - If satisfied by an operation: OCL added to post condition.
  - If satisfied by an onEntry/onExit expression: OCL added to onEntry/onExit effect.
SysML4MBT to UML4MBT transformation – Dynamic view

Statemachine Diagrams

- Shared concepts:
  - initial states,
  - final states,
  - standard states,
  - composite states,
  - transitions without signal reception,
  - OCL expressions without the circumflex operator.
Signal sending: ^ OCL operator
- Block.Port ^ Signal(parameters).
- Useful information: a new signal is pending in the corresponding port.

Instantiation of associations:

\[
\text{let } s = \text{Sig.allInstances() -> any(isUsed = false) in } \\
\text{s:Param1 = Val1 and s:Param2 = Val2 and } \\
\text{s:isUsed = true and } \\
\text{Block.Port_Sig -> includes(s)}
\]
Signal receiving (trigger on transitions)

- The corresponding signal is pending.
- After crossing the transition, the signal was read.
- Check of link and deletion.

A trigger defining the reception of “Sig” on “Port” hosted by “Block”.

Add of guard: 
\[ Block.Port_Sig -> notEmpty() \]

Add of effect: 
\[ let s = Block.Port_Sig.allInstances()->any(true) in s:isUsed = false and Block.Port_Sig->excludes(s) \]
Fork & join states

Rewriting to parallel states.

SysML4MBT to UML4MBT transformation – Fork & join states
Composite states

Must not contain parallel or historic states.
SysML4MBT to UML4MBT transformation – Historic states
Parallel states

- Same steps to merge parallel states and parallel Statemachines.

Parallelism of Statemachine Diagram

- Multiple Statemachines in SysML4MBT.
- Single Statemachine in UML4MBT.

=> Merging of Statemachines
Following steps:

1. Translation of all complex states (fork, join, composite, parallel and historic states).

2. Cartesian product of all Statemachines:
   1. Draw transition only if a path to reach start state exists.
   2. Informations of pending signals are stored on states.
   3. Transitions triggered by signal receiving: drawn only if the signal is pending on the root state.
UML Object Diagram

- Each class => one instance.
- Associations => instantiated using the minimum number of links (lower multiplicity).
- Classes representing signals: instantiated according how many times it can be pending at the same time.
Case studies

- **Lighting**
  - Front lighting system of a car.
  - Light on and light off independently headlights and highlights with a control lever.

- **Steering**
  - Representation of the steering column of a car.
  - Reaction of the steering column in regard of road.

- **Wiper**
  - Specification of the front wiper system of a car.
  - The modeled functionalities are slow speed drying up, high speed drying up, intermittently speed drying up and cleaning with drying up.
## Results

<table>
<thead>
<tr>
<th></th>
<th>Lightings</th>
<th>Steering</th>
<th>Wiper</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blocks</td>
<td>6</td>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td>Connectors</td>
<td>4</td>
<td>10</td>
<td>18</td>
</tr>
<tr>
<td>Statemachines</td>
<td>5</td>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>States</td>
<td>(2,2,2,2,4)</td>
<td>(2,2,2,2,2)</td>
<td>(1,1,1,1,1,2,17,10,2,2,2,2)</td>
</tr>
<tr>
<td>Transitions</td>
<td>(3,3,3,3,9)</td>
<td>(3,3,3,3,2,8)</td>
<td>(3,4,3,5,2,4,53,17,3,3,3,3)</td>
</tr>
<tr>
<td>Classes</td>
<td>10</td>
<td>16</td>
<td>29</td>
</tr>
<tr>
<td>Objects</td>
<td>15</td>
<td>20</td>
<td>57</td>
</tr>
<tr>
<td>States</td>
<td>64</td>
<td>18</td>
<td>2526</td>
</tr>
<tr>
<td>Transitions</td>
<td>256</td>
<td>123</td>
<td>31873</td>
</tr>
</tbody>
</table>
Conclusion & future works

- Rewriting rules to translate SysML4MBT models into UML4MBT models.
- Made it possible to generate test cases from SysML4MBT models with Test Designer.

Problem about Scalability.
- Improving rewriting rules.
- Increasing UML4MBT expressiveness (native support of parallelism).

About testing: Increasing model coverage with new test generation strategies.
Any questions?