Project Presentation #3

Synchronization between display objects and representation templates in graphical language construction

François Helg
Fabien Rohrer

Assistant :
Frédéric Fondement

Plan

- Recall of the starting goals
- What goals do we have achieved
- Used technologies
- Technical implementation of the solution
- Example
- Demo
- Conclusion

Overview

- François
  - Recall of the starting goals
  - What goals do we have achieved.
  - Used technologies
  - Technical implementation of the solution
    - Catch the semantically rich events from ProBXS
- Fabien
  - Technical implementation of the solution
    - Interaction with SVG/DOM
    - Model and Interpreter
  - Example
- Both
  - Demo of ProBXS with rising synchronization
- François
  - Conclusion

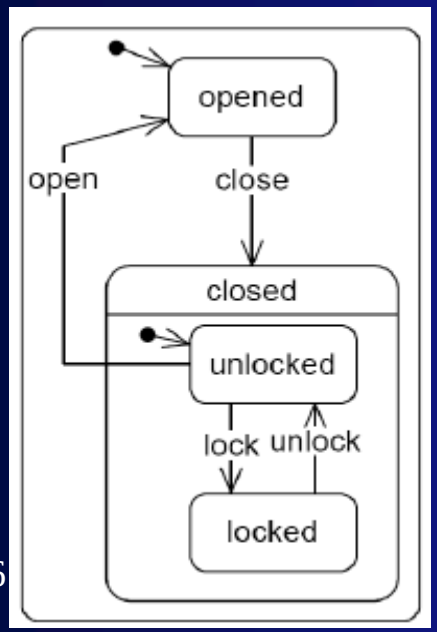
Plan

- Recall of the starting goals
- What goals do we have achieved
- Used technologies
- Technical implementation of the solution
- Example
- Demo
- Conclusion

Starting goals

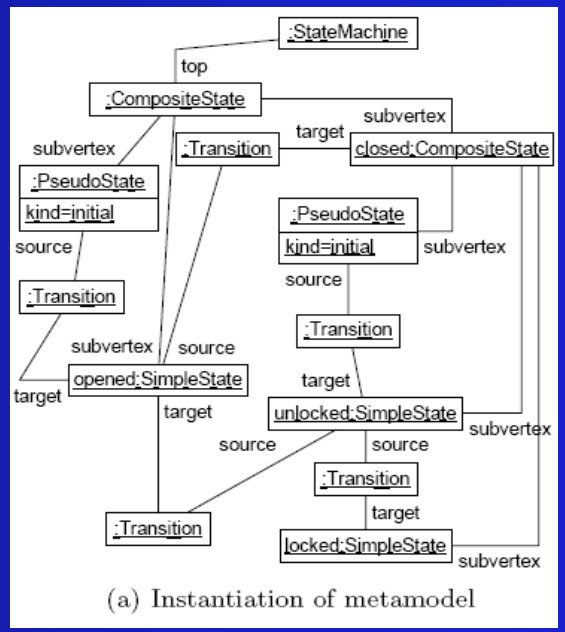
- Implement both rising and descendent synchronization

Graphical Representation



Synchronization

Model



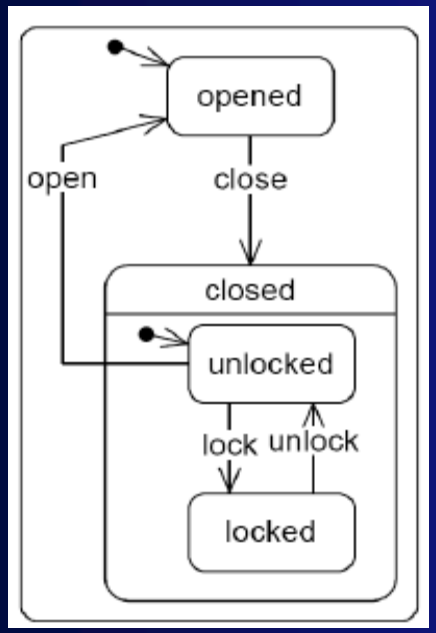
(a) Instantiation of metamodel

Plan

- Recall of the starting goals
- What goals do we have achieved
- Used technologies
- Technical implementation of the solution
- Example
- Demo
- Conclusion

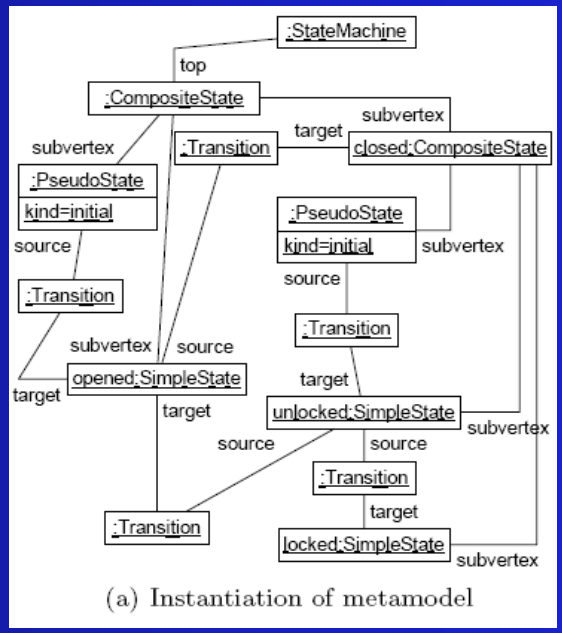
Achieved goals
- Only the rising synchronization -

Graphical Representation



« rising »
Synchronization

Model



(a) Instantiation of metamodel

Achieved goals
- Only the rising synchronization -

- It means that the model can keep synchronized when a change occurs in the graphical representation but not the other way round.
- Reason: Time
  - To solve problems linked with Kermeta
  - To implement the rising synchronization

Achieved goals
- Why did we abandon Kermeta? -

- Some little annoying problems :
  - Bugs in Kermeta2Ecore transformation
  - Problems with saving a model
  - Error detection too imprecise
  - Lack of documentation
- Main problem:
  - Standalone application not implemented yet
  - Unable to dynamically interpret the code
- Unusable!

Plan

- Recall of the starting goals
- What goals do we have achieved
- Used technologies
- Technical implementation of the solution
- Example
- Demo
- Conclusion

Used technologies
- Our solution -

- XMI: format use to exchange model and metamodel
- JMI: Mapping from MOF to Java. It provides an interface to handle model.

- MDR:
  - Can load a metamodel and save an instance (model) thanks to XMI.
  - Can handle model thanks to JMI

- « model-script » encryption and interpretation
  - Koala (Dynamic Java code interpretor)
  - Not Kermeta any more!

- Interaction with ProBXS
  - Java 1.5
    - *DOM for interaction with SVG templates*

Used Technologies
- Dynamic Java: Koala -

- Fortunately we found a Koala!
  - Koala is a Java dynamic interpreter
  - It can interpret a derivated-Java: DynamicJava
  - Free and open source
  - Ideal for instantiating a JMI metamodel
- So we can use it for interpret our queries!

Plan

- Recall of the starting goals
- What goals do we have achieved
- Used technologies
- Technical implementation of the solution
- Example
- Demo
- Conclusion

- Switch -

- Fabien will continue the presentation…

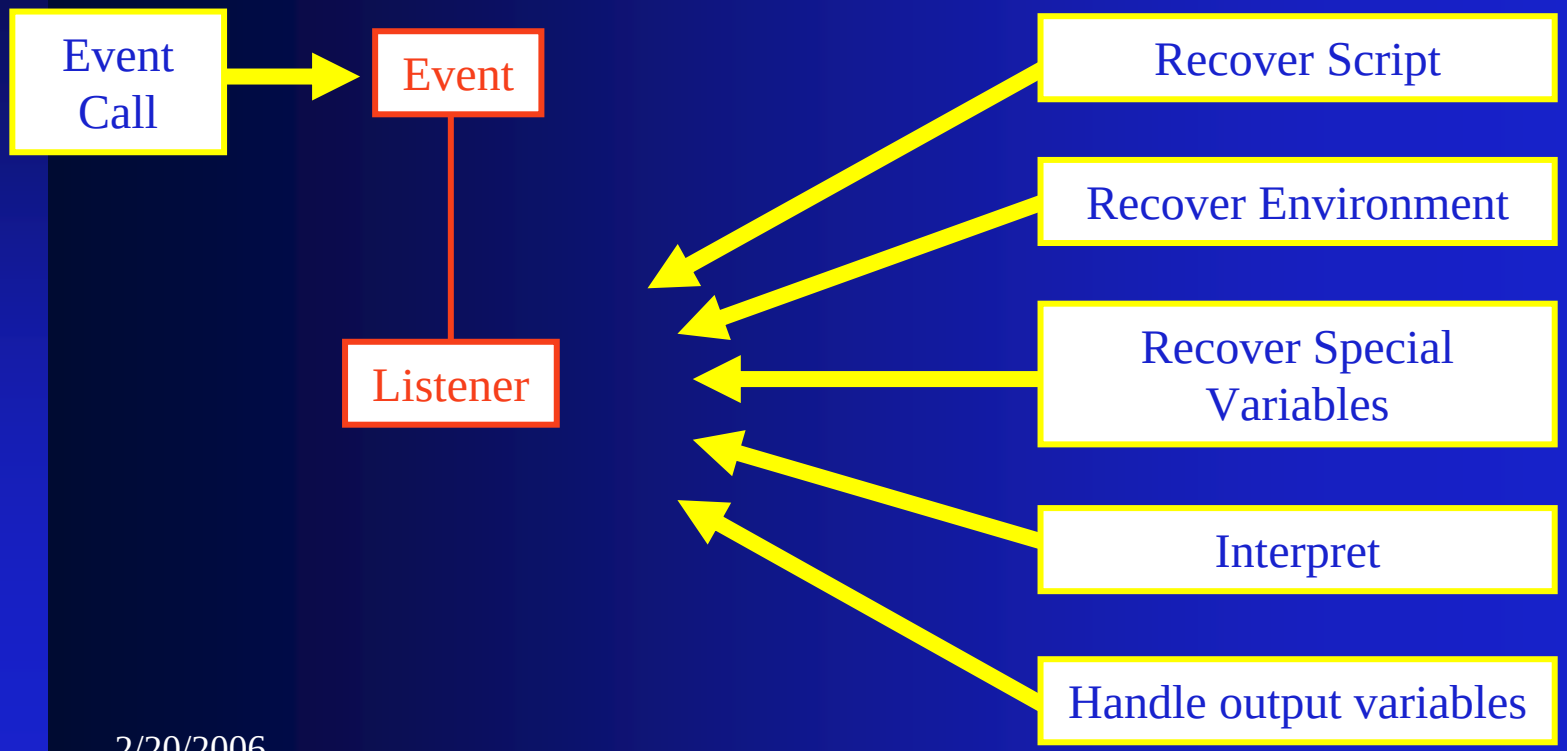
Technical implementation
- Generalities -

- The program has to « understand » a SVG template like:

```
<svg onCreation="{Koala| s =
    model.getSimpleState().createSimpleState();}" <!-- … --> >
    <!-- … -->
    <text onChange="{Java| self.setName(content);}" var_self="$s"
          dpi:component="test.EditableString">newState</text>
    <!-- … -->
</svg>
```

Technical implementation
- Generalities -

- The rising synchronization can be subdivided into a chain of simpler tasks



Technical implementation
- Plan -

- The rising synchronization can be subdivided into a chain of simpler tasks
  - Listeners on sementically rich events in graphical representation (ProBXS)
  - Retrieve script, environment from DOM-tree and parameters of the events.
  - Set the environment in the right language
  - Interpret the script
  - Get back the environment from the interpreter

Technical implementation
- Listeners -

- Listeners on sementically rich events
  - CharacterDeletedEvent, CharacterInsertedEvent: Appear when a character is changed in a textbox
  - ComponentCreatedEvent: Appear when a new component is added into the scene
  - DirectionAdjustEvent: Appear when a component changes its direction
  - LocateEvent, PositionEvent, TranslateEvent: Appear when a component move on the scene
  - ContainedEvent: Appear when a component is slided into another component
  - StickEvent: Appear when a component is sticked to another one.

Technical implementation
- Plan -

- The rising synchronization can be subdivided into a chain of simpler tasks
  - Listeners on sementically rich events in graphical representation (ProBXS)
  - Retrieve script, environment from DOM-tree and parameters of the events.
  - Set the environment in the right language
  - Interpret the script
  - Get back the environment from the interpreter

Technical implementation
- Recover Information -

- Retrieve script, environment from DOM-tree and parameters of the events.
  - SVG template → DOM tree
  - Link DOM – ProBXS => Component class (each component wrapps a dom element)
  - Recover the node corresponding to the event is easy because we have the wrapped element (DOM node) of the component which raises the event.

Technical implementation
- Recover Information -

- Important values in DOM tree :
  - onEvent=”{ Language | codeToInterpret }”
  - var_name = “value”

- Example :

onEvent="{Koala | System.out.println(val);}" var_val="String(Hello world)"

Technical implementation
- Plan -

- The rising synchronization can be subdivided into a chain of simpler tasks
  - Listeners on sementically rich events in graphical representation (ProBXS)
  - Retrieve script, environment from DOM-tree and parameters of the events.
  - Set the environment in the right language
  - Interpret the script
  - Get back the environment from the interpreter

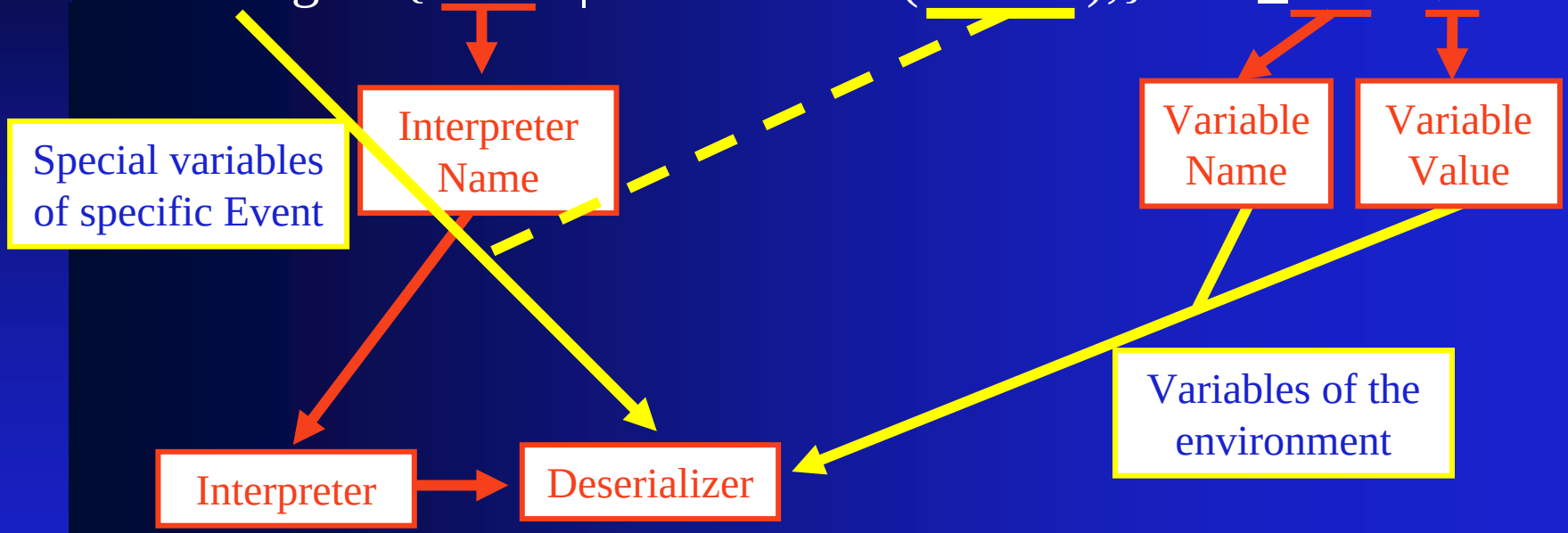
Technical implementation
- Environment -

- Recover the environment
  - Recover the right interpreter
  - Recover the environment
  - Recover the special variables of the specific environment

Technical implementation
- Environment -

- Example :

onChange="{Koala | self.setName(content);}" var_self="$s"



Technical implementation
- Special Variables -

- For each event:
  - Some specific variables
    - Ex: variable *content* for CharacterInsertedEvent
  - Variable *model* to handle the model

- → Compete list in the report

Technical implementation
- Plan -

- The rising synchronization can be subdivided into a chain of simpler tasks
  - Listeners on sementically rich events in graphical representation (ProBXS)
  - Retrieve script, environment from DOM-tree and parameters of the events.
  - Set the environment in the right language
  - Interpret the script
  - Get back the environment from the interpreter

Technical implementation
- Interpretation -

- At this point we have all what we need:
  - The right interpreter
  - The right variables
  - The code to interpret
- So we can interpret!

Technical implementation
- Plan -

- The rising synchronization can be subdivided into a chain of simpler tasks
  - Listeners on sementically rich events in graphical representation (ProBXS)
  - Retrieve script, environment from DOM-tree and parameters of the events.
  - Set the environment in the right language
  - Interpret the script
  - Get back the environment from the interpreter

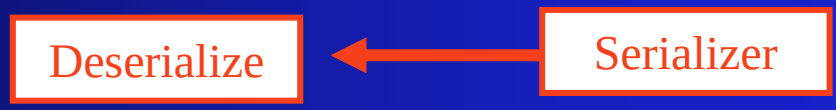
Technical implementation
- Output variables -

- The interpretation generates some output variables
- We need to handle them!

Technical implementation
- Output variables -

- Example :

onCreation="{Koala | s = model.getState().createState();}"

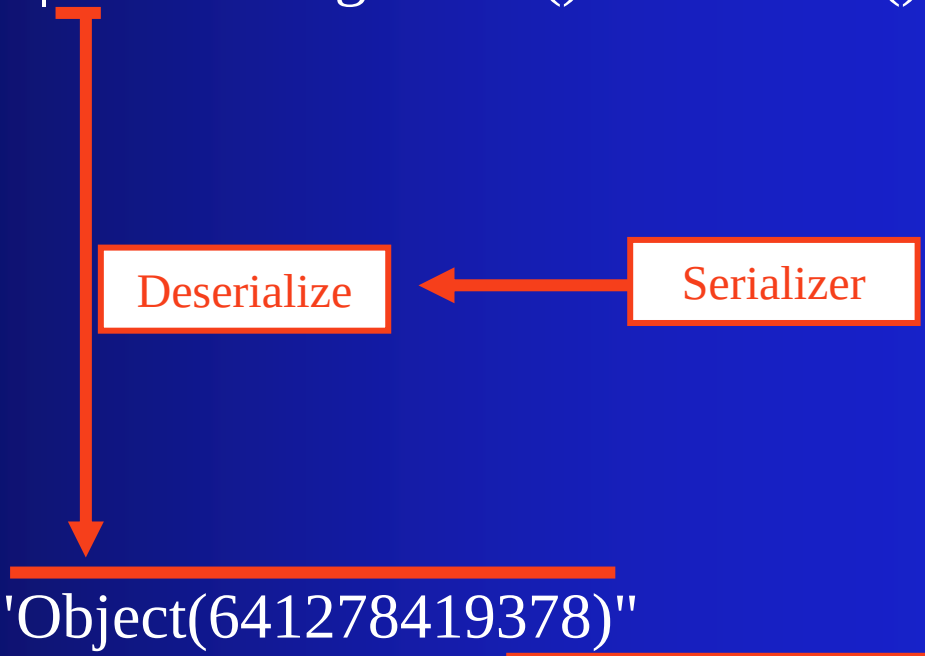


var_self="$s"

Technical implementation
- Output variables -

- Example :

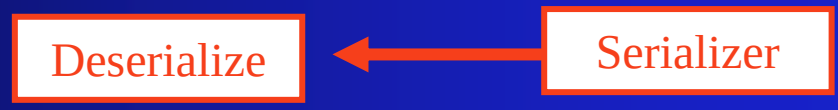
onCreation="{Koala | s = model.getState().createState();}"



Technical implementation
- Output variables -

- Example :

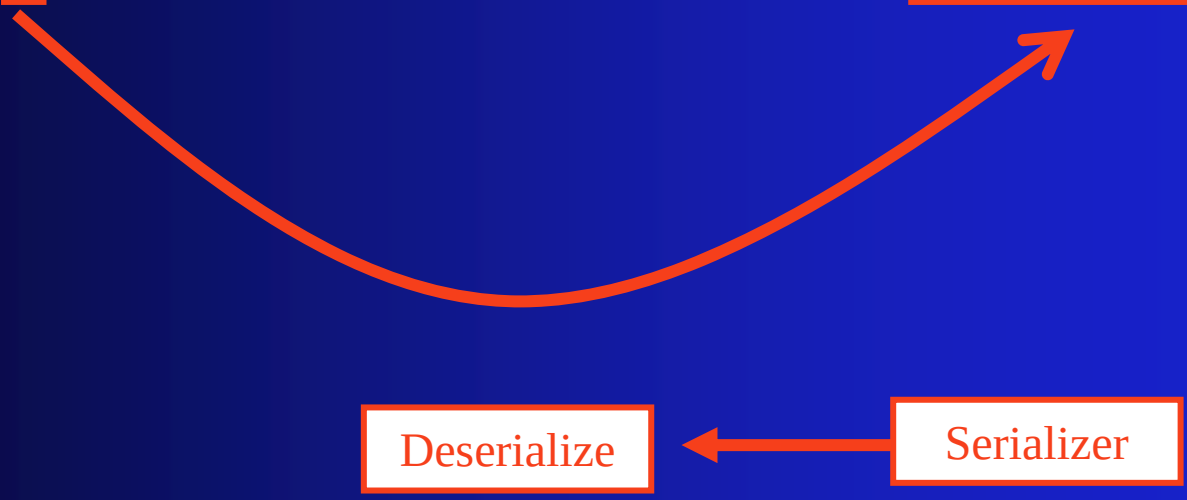
"{Koala | b = new Boolean(false);}" var_b="Boolean(true)"



Technical implementation
- Output variables -

- Example :

"{Koala | b = new Boolean(false);}" var_b="Boolean(false)"



Plan

- Recall of the starting goals
- What goals do we have achieved
- Used technologies
- Technical implementation of the solution
- Example
- Demo
- Conclusion

Example
- stick -

- Point of view of the stuck object t (ex: Transition):

```
<!-- … -->
<rect  var_self="$t"  var_isSource="Boolean(true)“
  dpi:component="PBXSComponents.Arrow“ … />
<polygon  var_self="$t"  var_isSource="Boolean(false)“
  dpi:component="PBXSComponents.Arrow" … />
<!-- … -->
```

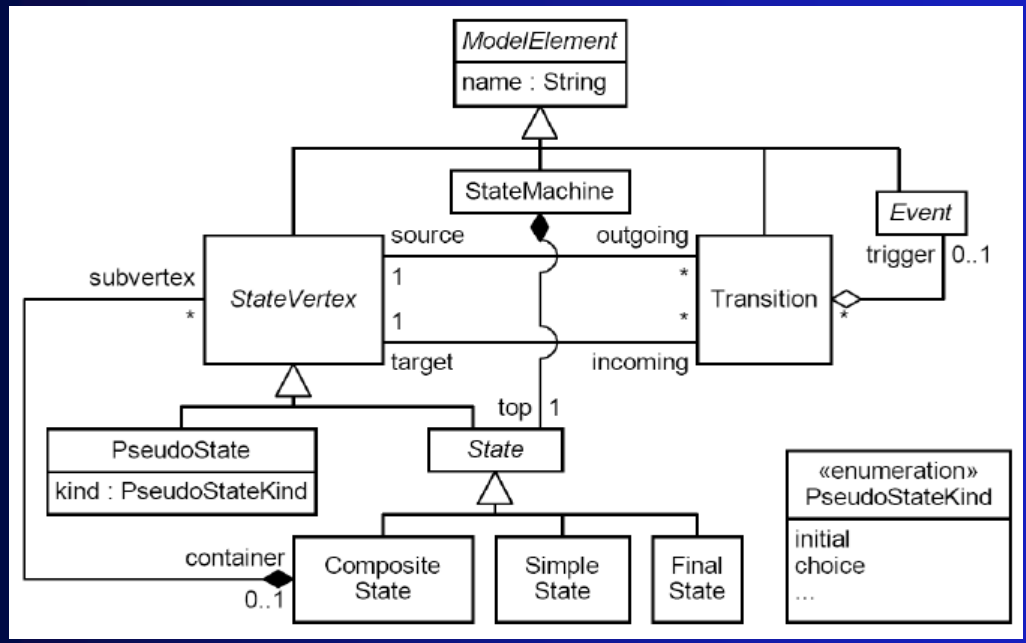
- Point of view of the stickable object s (ex: State):

```
<!-- … -->
<rect  onStick="{Java|
     if (((Boolean) stickedComponent_isSource).booleanValue()) {
         self.getOutgoing().add(stickedComponent_self);
      } else {
         self.getIncoming().add(stickedComponent_self);
      };}"
   var_self="$s"  dpi:component="PBXSComponents.AnchorPoint“ … />
<!-- … -->
```

Plan

- Recall of the starting goals
- What goals do we have achieved
- Used technologies
- Technical implementation of the solution
- Example
- Demo
- Conclusion

Demo
- Statechart Language -



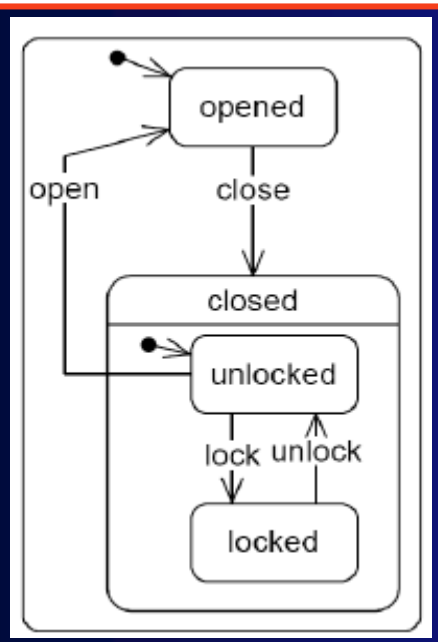
Transition	SimpleState	Composite State	FinalState	PseudoState (initial)	PseudoState (choice)
–event→	name	name / contents	◉	●	○

Plan

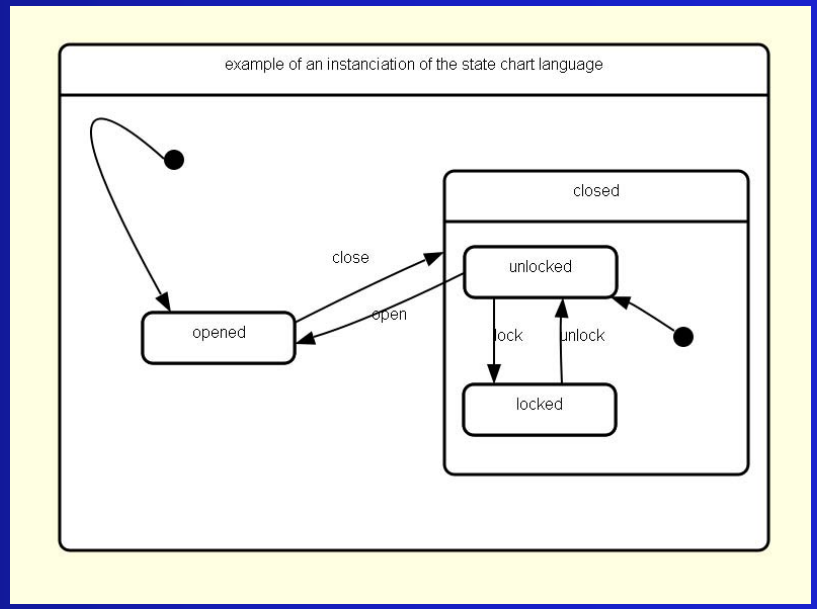
- Recall of the starting goals
- What goals do we have achieved
- Used technologies
- Technical implementation of the solution
- Example
- Demo
- Conclusion

Conclusion
- Results -

The beginning possible instantiation of the model

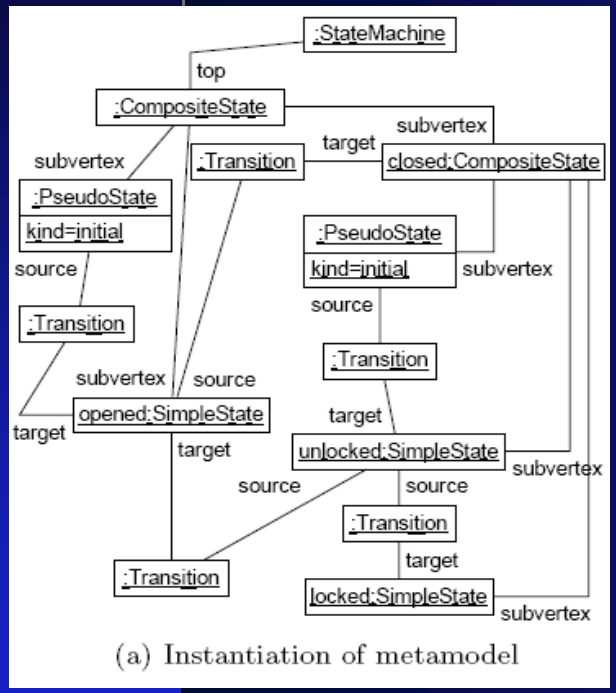


The same instantiation in ProBXS



Conclusion
- Results -

The corresponding model



(a) Instantiation of metamodel

The generated model in the XMI file

```
<?xml version = '1.0' encoding = 'windows-1252' ?>
<XMI version = '2.0' xmlns = 'http://www.omg.org/XMI'>
 <sc.StateMachine id = 'a1'>
  <sc.StateMachine.top>
   <sc.CompositeState id = 'a2' name = 'example of an
instanciation of the statechart language'>
    <sc.CompositeState.subvertex>
     <sc.CompositeState id = 'a3' name = 'closed'>
      <sc.StateVertex.incoming>
       <sc.Transition idref = 'a4'/>
      </sc.StateVertex.incoming>
      <sc.CompositeState.subvertex>
       <sc.SimpleState id = 'a5' name = 'unlocked'>
        <sc.StateVertex.outgoing>
         <sc.Transition idref = 'a6'/>
         <sc.Transition idref = 'a7'/>
        </sc.StateVertex.outgoing>
        <sc.StateVertex.incoming>
         <sc.Transition idref = 'a8'/>
         <sc.Transition idref = 'a9'/>
        </sc.StateVertex.incoming>
       </sc.SimpleState>
       <sc.SimpleState id = 'a10' name = 'locked'>
        <sc.StateVertex.outgoing>
         <sc.Transition idref = 'a9'/>
        </sc.StateVertex.outgoing>
        <sc.StateVertex.incoming>
         <sc.Transition idref = 'a6'/>
        </sc.StateVertex.incoming>
       </sc.SimpleState>
       <sc.PseudoState id = 'a11' kind = 'initial'>
        <sc.StateVertex.outgoing>
         <sc.Transition idref = 'a8'/>
        </sc.StateVertex.outgoing>
       </sc.PseudoState>
      </sc.CompositeState.subvertex>
     </sc.CompositeState>
     <sc.SimpleState id = 'a12' name = 'opened'>
      <sc.StateVertex.outgoing>
       <sc.Transition idref = 'a4'/>
      </sc.StateVertex.outgoing>
      <sc.StateVertex.incoming>
       <sc.Transition idref = 'a13'/>
       <sc.Transition idref = 'a7'/>
      </sc.StateVertex.incoming>
     </sc.SimpleState>
     <sc.PseudoState id = 'a14' kind = 'initial'>
      <sc.StateVertex.outgoing>
       <sc.Transition idref = 'a13'/>
      </sc.StateVertex.outgoing>
     </sc.PseudoState>
    </sc.CompositeState.subvertex>
   </sc.CompositeState>
  </sc.StateMachine.top>
 </sc.StateMachine>
 <sc.Transition id = 'a13'>
  <sc.Transition.source>
   <sc.PseudoState idref = 'a14'/>
  </sc.Transition.source>
  <sc.Transition.target>
   <sc.SimpleState idref = 'a12'/>
  </sc.Transition.target>
 </sc.Transition>
 <sc.Transition id = 'a8'>
  <sc.Transition.source>
   <sc.PseudoState idref = 'a11'/>
  </sc.Transition.source>
  <sc.Transition.target>
   <sc.SimpleState idref = 'a5'/>
  </sc.Transition.target>
 </sc.Transition>
 <sc.Transition id = 'a6' name = 'lock'>
  <sc.Transition.source>
   <sc.SimpleState idref = 'a5'/>
  </sc.Transition.source>
  <sc.Transition.target>
   <sc.SimpleState idref = 'a10'/>
  </sc.Transition.target>
 </sc.Transition>
 <sc.Transition id = 'a9' name = 'unlock'>
  <sc.Transition.source>
   <sc.SimpleState idref = 'a10'/>
  </sc.Transition.source>
  <sc.Transition.target>
   <sc.SimpleState idref = 'a5'/>
  </sc.Transition.target>
 </sc.Transition>
 <sc.Transition id = 'a4' name = 'close'>
  <sc.Transition.source>
   <sc.SimpleState idref = 'a12'/>
  </sc.Transition.source>
  <sc.Transition.target>
   <sc.CompositeState idref = 'a3'/>
  </sc.Transition.target>
 </sc.Transition>
 <sc.Transition id = 'a7' name = 'open'>
  <sc.Transition.source>
   <sc.SimpleState idref = 'a5'/>
  </sc.Transition.source>
  <sc.Transition.target>
   <sc.SimpleState idref = 'a12'/>
  </sc.Transition.target>
 </sc.Transition>
</XMI>
```

Conclusion
- Further work -

- Descendent synchronization
- Load an existing model
- Correction of bugs

Time for questions!