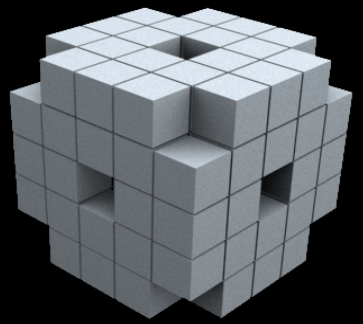


Tihamér Levendovszky

Code generation with the model transformation of Visual Behavior models

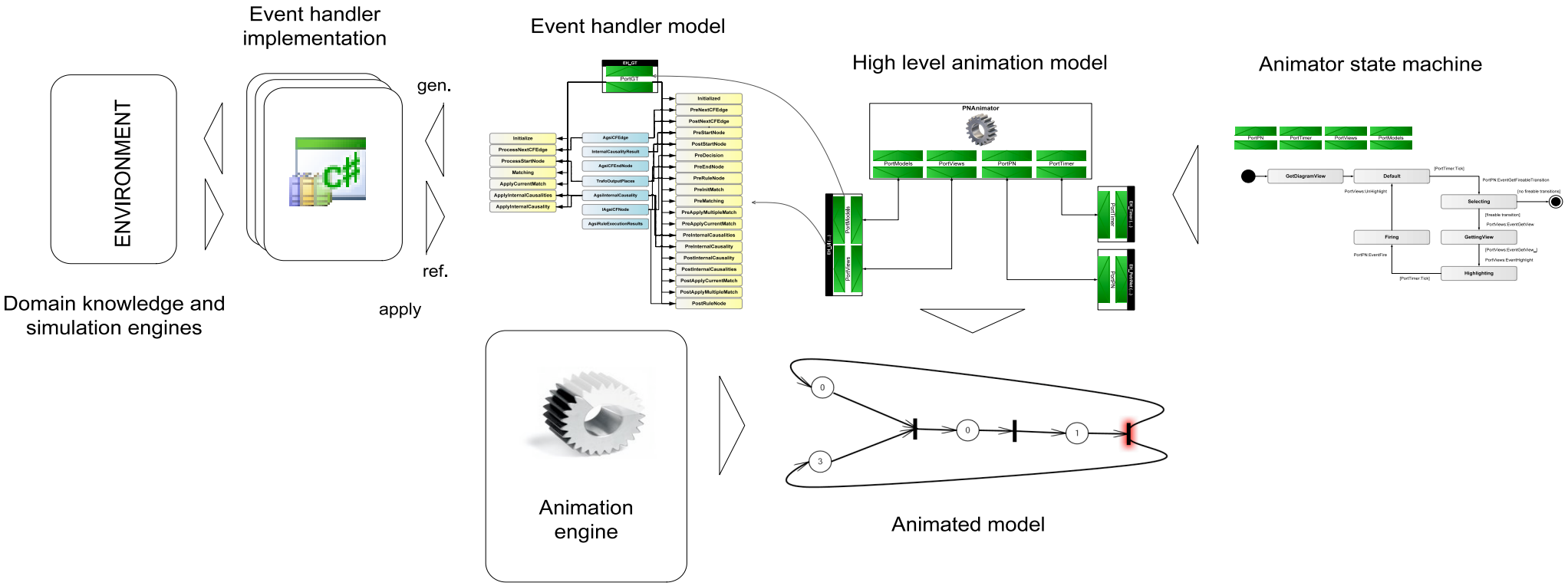
Tamás Mészáros, Gergely Mezei
Budapest University of Technology and Economics
Departement of Automation and Applied Informatics
Visual Modeling and Transformation System (VMTS)

Tihamér Levendovszky
Vanderbilt University
Institute for Software Integrated Systems

Overview

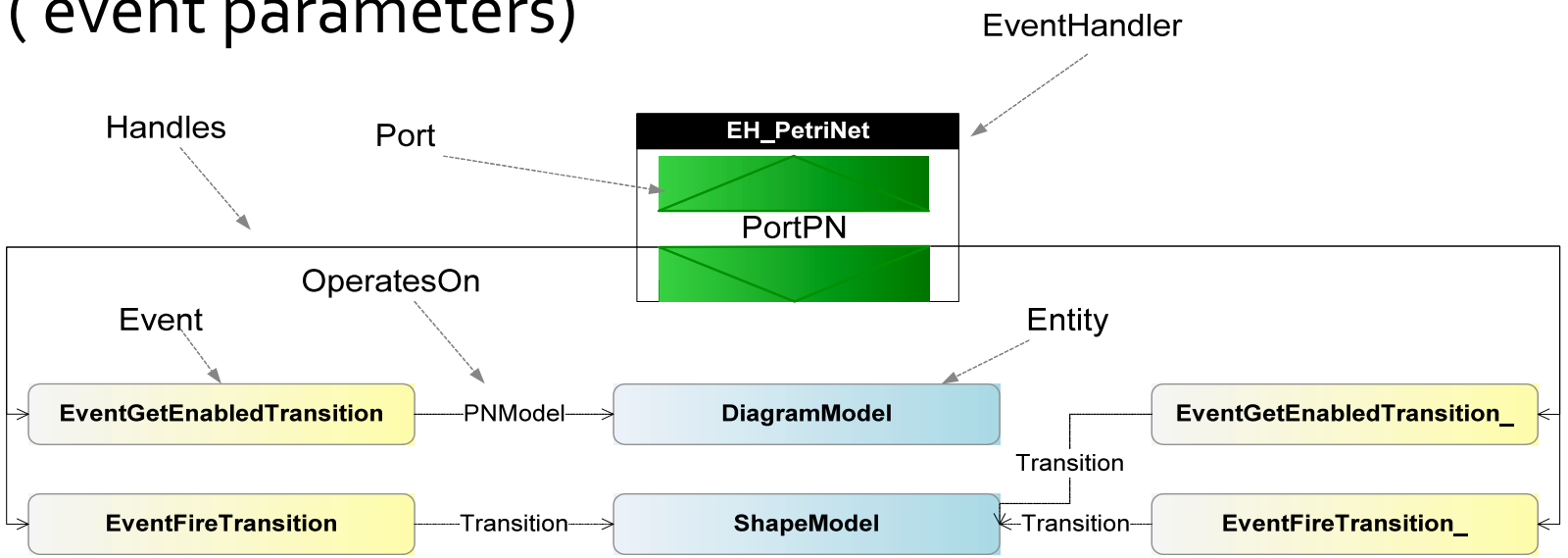
- Defining the Dynamic Behavior of Graphical DSLs
  - VMTS Animation Framework
- Transformation to C# DOM
  - (MS CodeDOM-like)

Animation Framework Overview



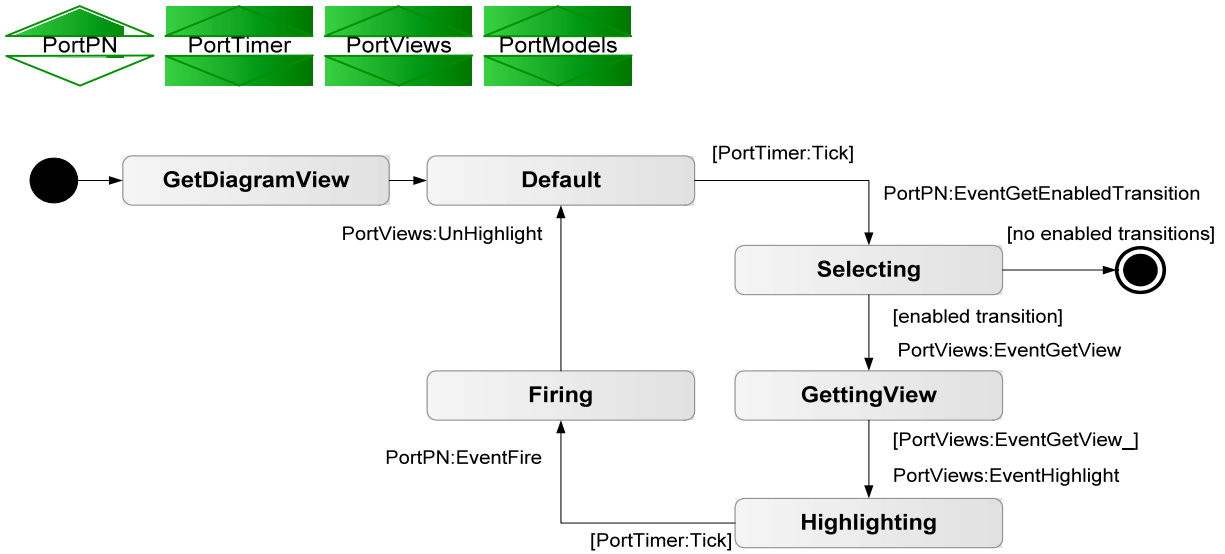
VMTS Animation Framework

- Event handler model
  - Event-driven interface for external components
  - Elements
    - Events
    - Entities (event parameters)
    - Ports



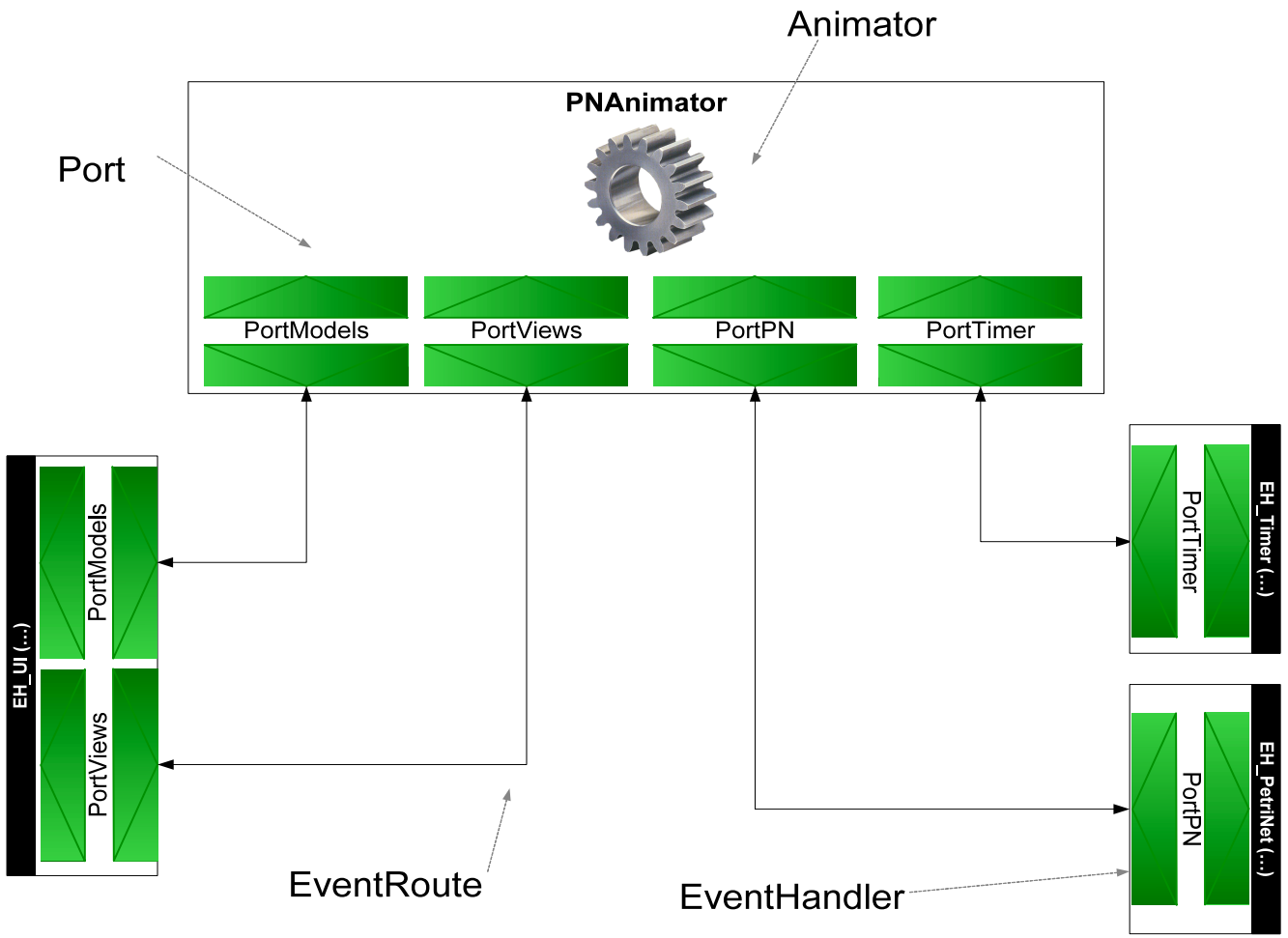
VMTS Animation Framework

- Animator
  - Event-driven state machine
  - Port-based interface
  - States
  - Transitions
    - Guard expression
    - Action expression



VMTS Animation Framework

- High Level Animation Model
  - Event handler
  - Animator
  - Port
  - Event route



VMTS Animation Framework

- Separating animation from model simulation
  - The domain knowledge is considered black-box
  - Animation logic is described with an event-driven state machine
  - Integration with an event-driven approach
- The integration is supported with visual modeling techniques
- Processing animation models
  - Generating skeleton for the event handlers
    - Automated wrapping of third-party components
  - **Generating executable application from animator models**
    - Applies a DEVS-based simulation framework

Generated files – *State machines*

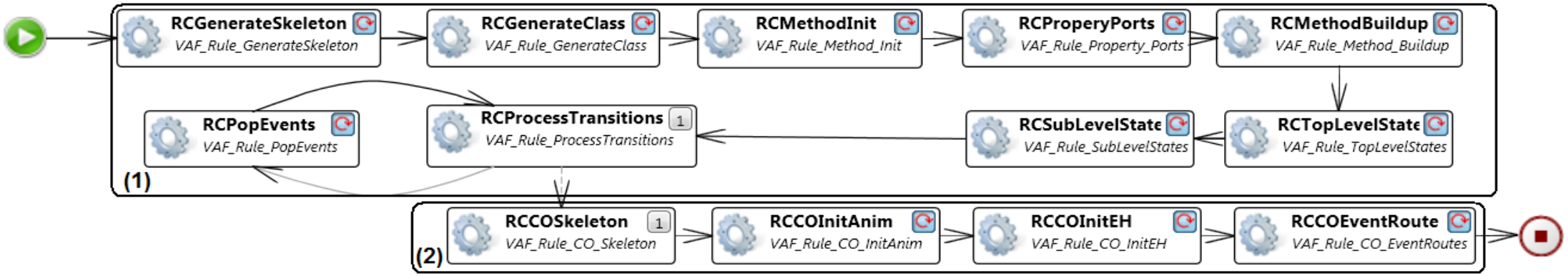
```
namespace VMTS.VAF {
  public class <AnimatorName> : Simulator {
    public Port <PortName> {get; private set;} ...
    public <AnimatorName>(Coordinator c) : base(c) { }
    <Variables of the animator>

    public override void Init() {
      //initialization of the ports
      <portName> = new Port(this);...
      <portName>.Capacity = <capacity>; ...
    }
    public override void BuildUp() {
      startState = new State(this,null,null);
      currentState = startState;
      State <stateName> = new State(this,
        <action>,<container state>); ...
      <fromState>.AddTransition( new Transition(
        this,<toState>, <guard>, <action> ),<isInternal>); ...
}}}
```

Generated files - *Configuration*

```
namespace VMTS.VAF {
  class Configuration {
    public Configuration() {
      coordinator = new Coordinator(); Init();
    }
    public Configuration( Coordinator _coordinator) {
      coordinator = _coordinator; Init();
    }
    private void Init() {
      <animator field> = new <animator type>(coordinator);...
      coordinator.Simulators.Add(<animator field>);...
      <event handler field> = new <eh type>(coordinator);
      coordinator.EventHandlers.Add(<event handler field>);...
      <setting event handler parameters>
      coordinator.AddMapping(<from>.<fromPort>, <to>.<toPort>);...
    }
    <Animator field declarations>
    <Event handler field declarations>
}}
```

Transformation

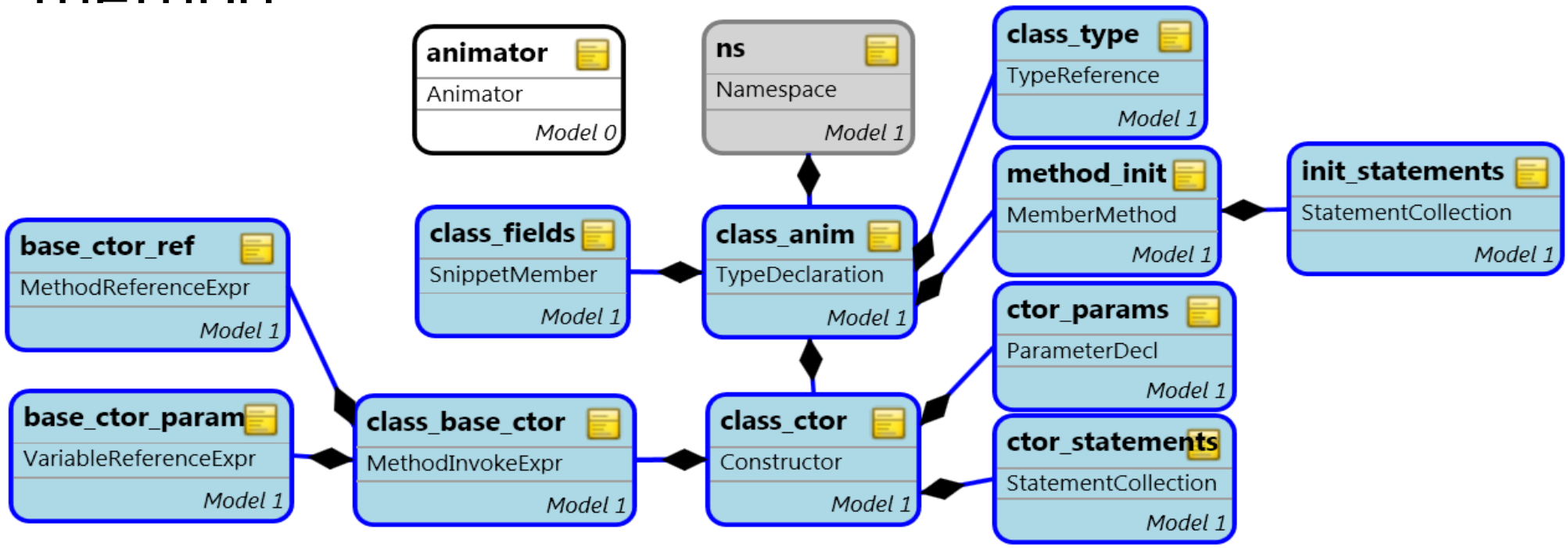
- Transforming animation models to C# DOM
- Transformation control flow



  - (1) Generating Animator classes
  - (2) Generating Configuration class which glues the components together

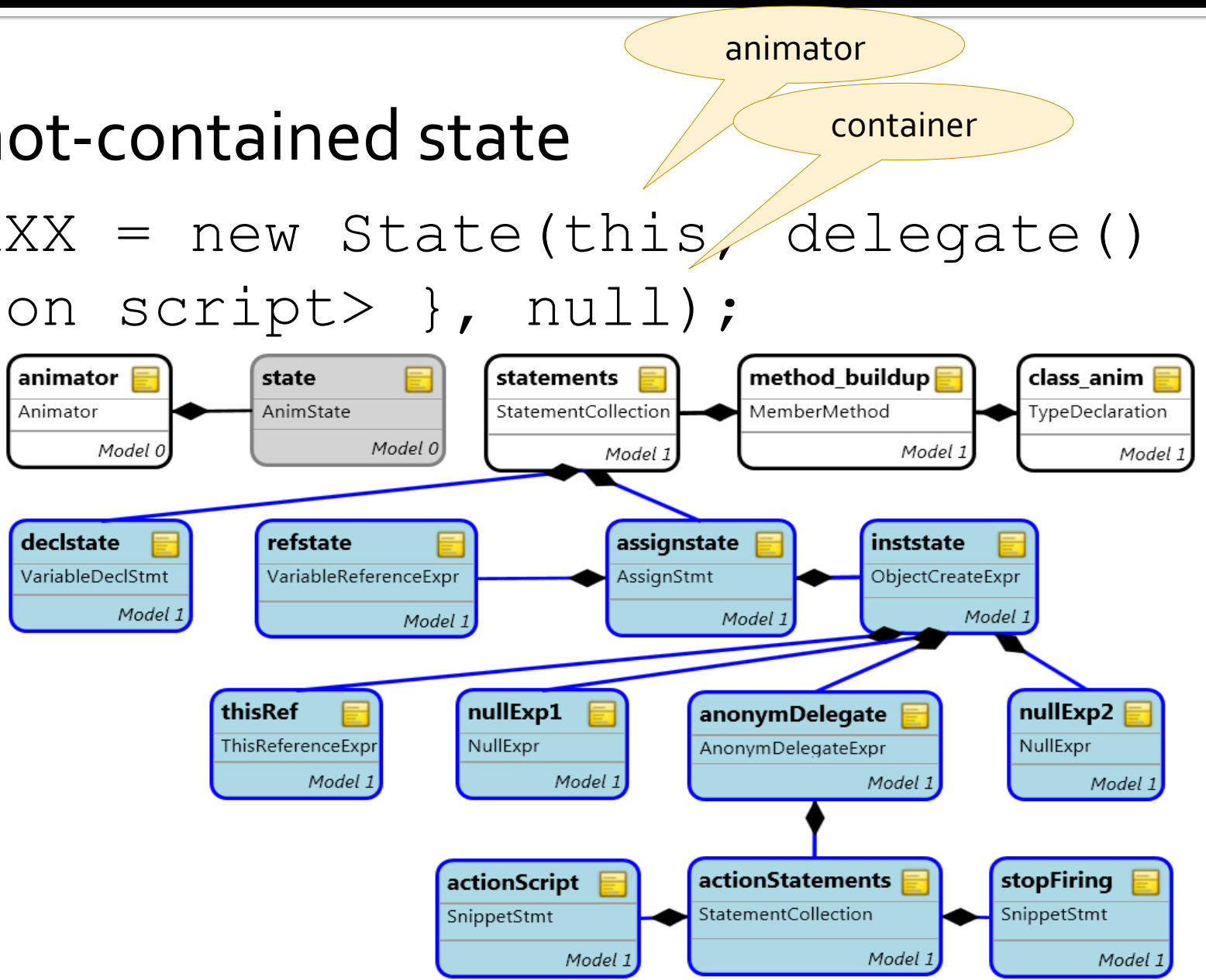
Animator classes - *GenerateClass*

- Executed for each *Animator* node
  - Exhaustive – flagging already processed ones
- Creates class declaration, constructor, *Init* method



Processing states - *TopLevelStates*

- For each not-contained state
  - `stateXXXX = new State(this, delegate() { <action script> }, null);`

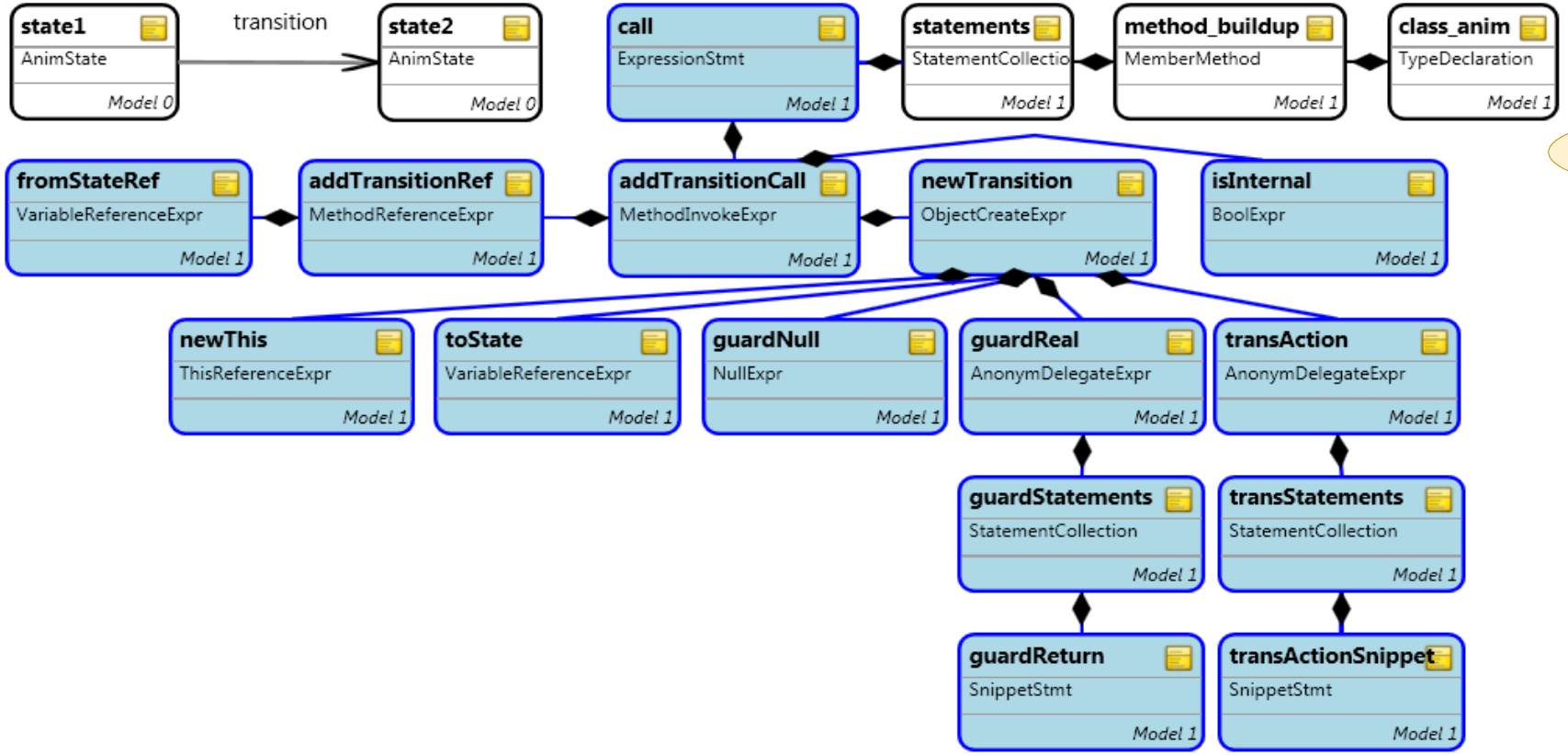


Processing transitions

- For each not-contained state
  - `<from>.AddTransition(new Transition(this, <to>, delegate(){ return <guard condition>;}, delegate(){ <action script> }), false);`

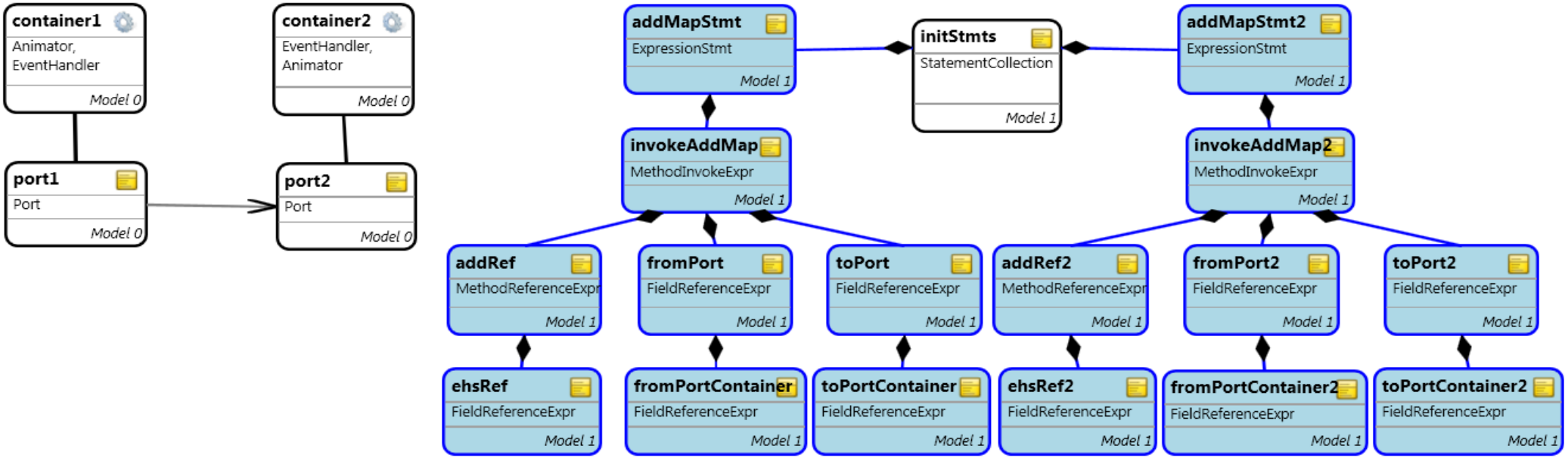
animator

not internal



Generating the Configuration class

- Connecting components (*EventRoute* rule)
  - `coordinator.AddMapping(<from>.<fromPort>, <to>.<toPort>)`
- Matching each port-pair => extending the Init method
- Conditional creation of a second *AddMapping* call
  - Direction of the event route (or bidirectional)

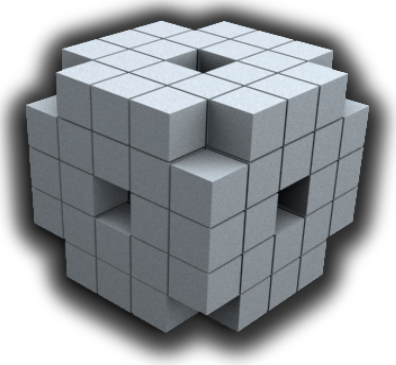


Conclusion

- Generating executable source code from animation models
  - Graph rewriting-based processing
- Easy to modify, formal verification possible
  - using graph and category theorem methods
  - Termination and complexity analysis, topological verification of the input
- Future work
  - Improvement of the modeling language
    - Using modeling instead of manual coding (firing of events, conditions on ports) : less error-prone, faster developement

Thank you for your attention!

Thank you for your attention!



http://vmts.aut.bme.hu