TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik Institut Software- und Multimediatechnik, Lehrstuhl Softwaretechnologie

Anticipating Unanticipated Tool Interoperability using Role Models

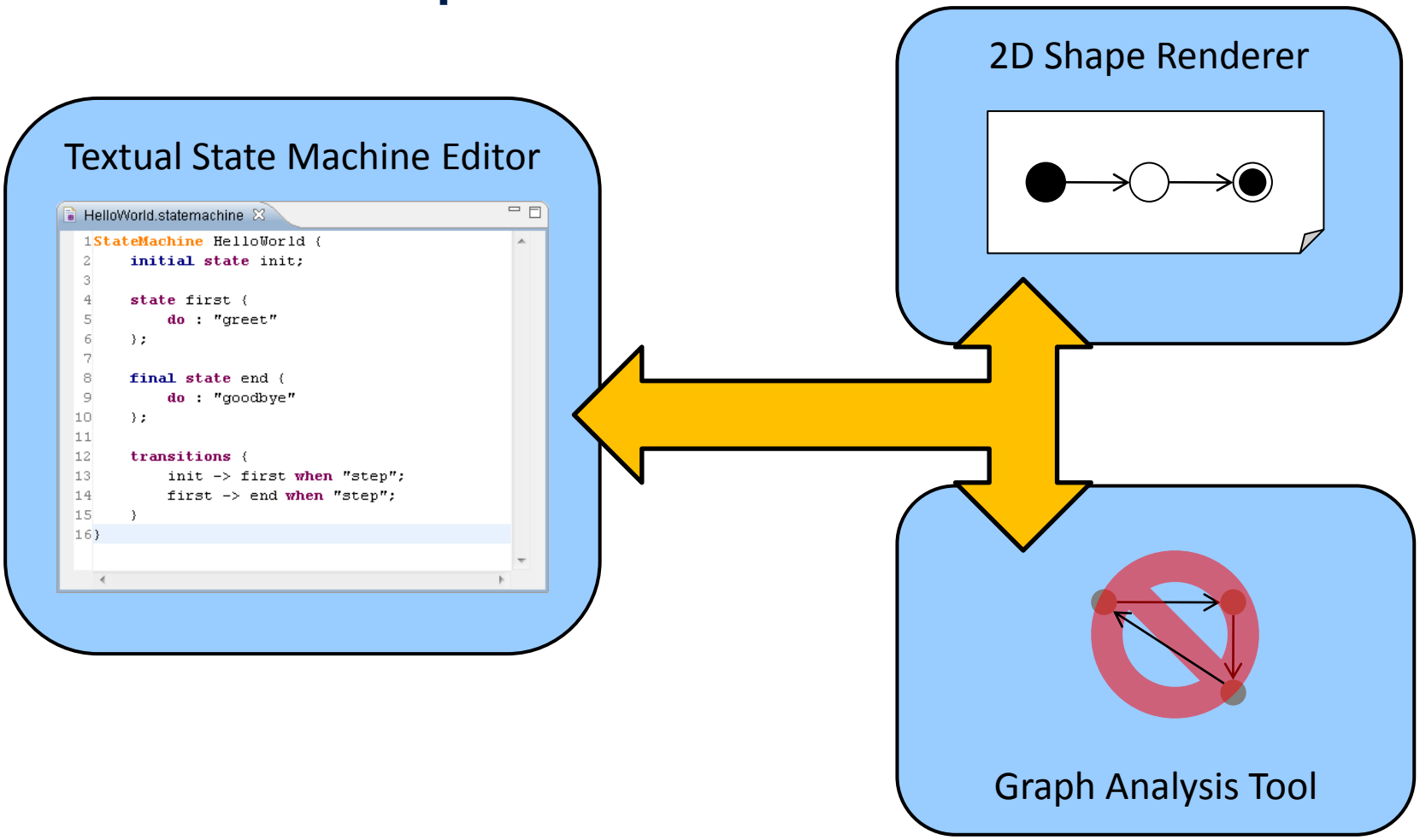
Mirko Seifert, Christian Wende, Uwe Aßmann

MDI Workshop, Oslo, Norway, 05.10.2010

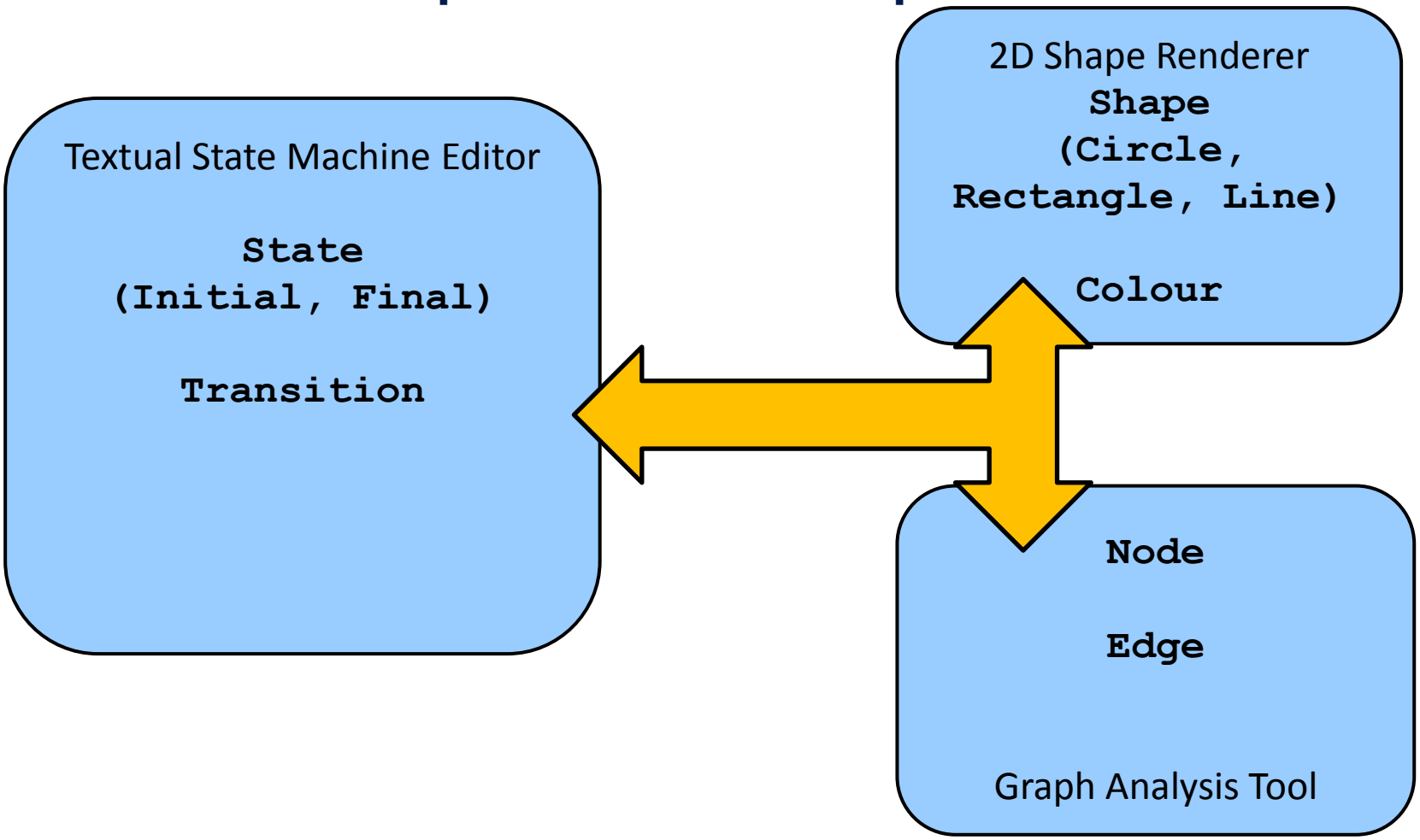
Outline

- Motivational Example
- Proactive vs. Retroactive Tool Integration
- Approach
- Conclusion

Motivational Example

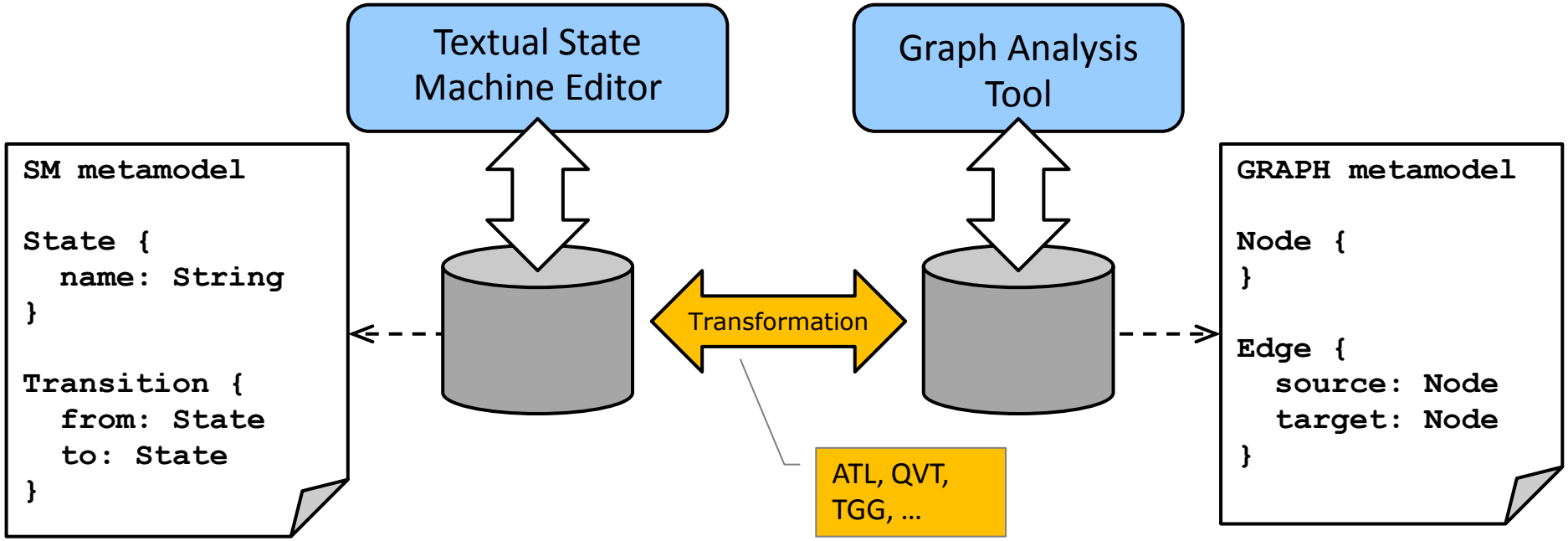


Motivational Example – Domain Concepts



Retroactive Tool Integration

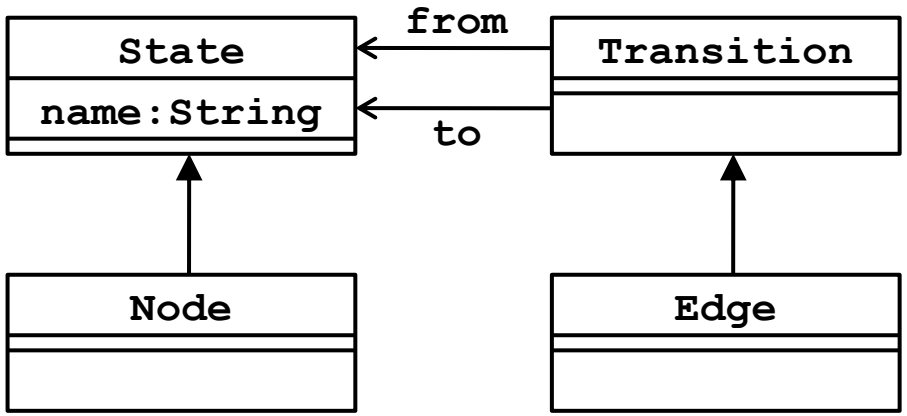
- Tools and metamodels already exist
- Use transformations to convert data from one tool to another



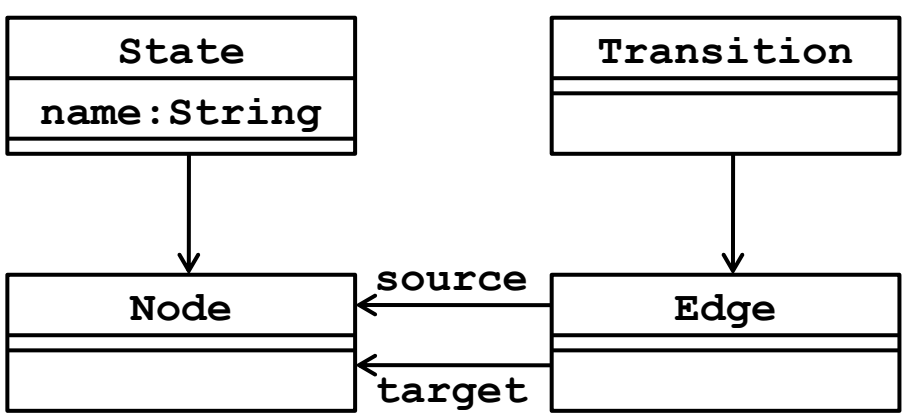
Proactive Tool Integration

- Tool and metamodels not fixed yet
- Use metamodel integration to make data from one tool accessible to another

a) Inheritance



b) Delegation

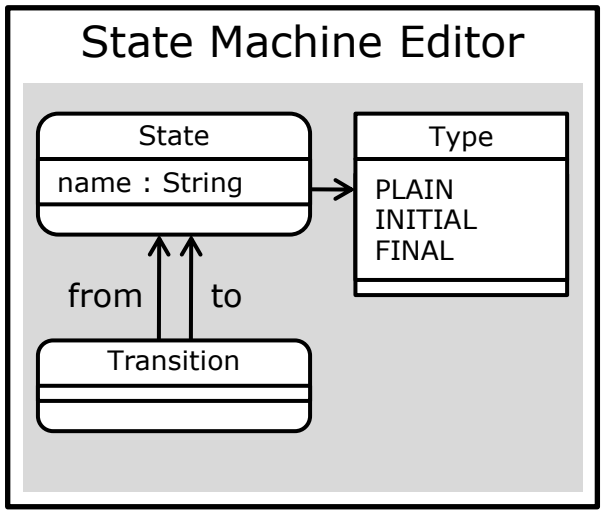


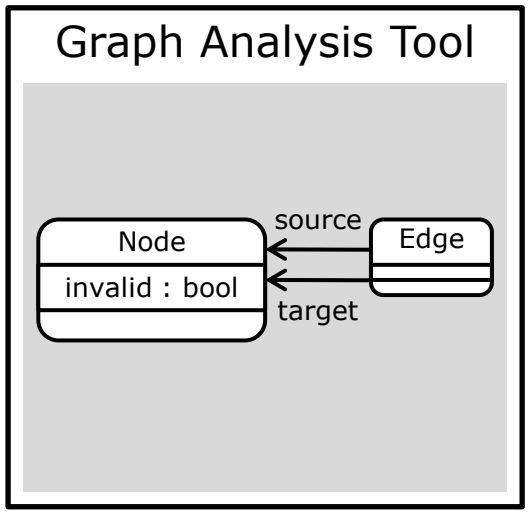
Petroactive vs. Retroactive Tool Integration

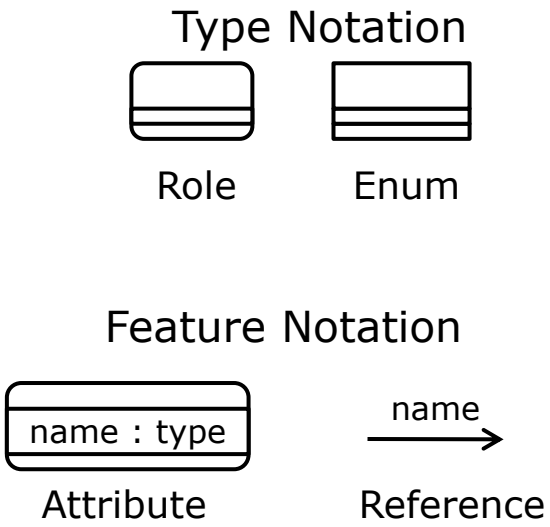
	Proactive	Retroactive
Technique	Inheritance, delegation	Transformation
Appropriate Abstraction	Metamodels need to be adapted ⊖	Metamodels unaffected ⊕
Tool Independence	Strong coupling ⊖	No coupling ⊕
Shared Data	Sharing among all integrated tools ⊕	Replicated Data, Synchronization needed ⊖
Tool Interaction	Support for anticipated interaction only ⊕/⊖	Transformations hinder interaction ⊖

Tool Integration using Role Models

- Use roles instead of classes







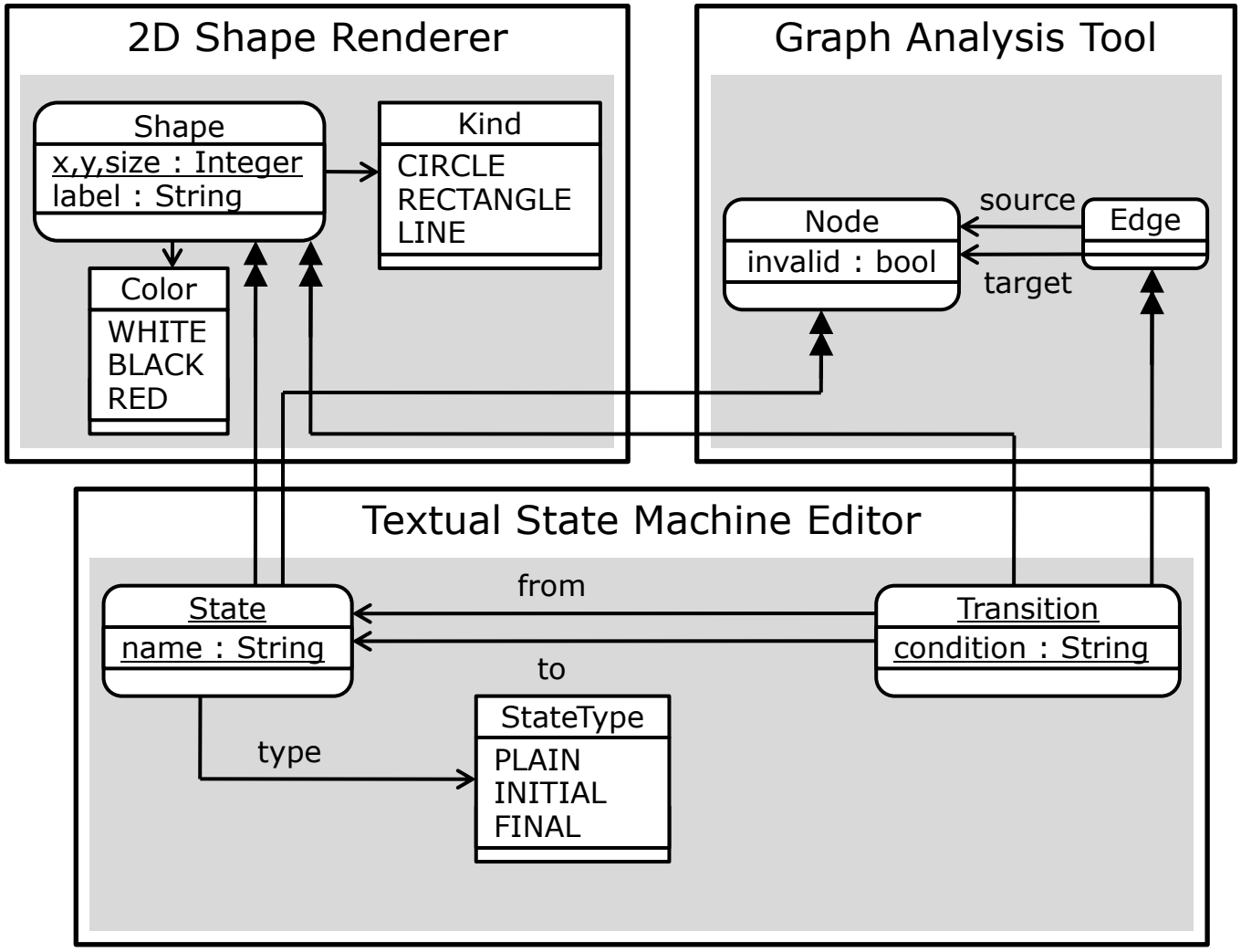
- At first sight not much different from object-oriented metamodels

Tool Integration using Role Models

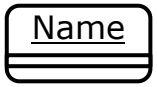
- Use role bindings/grounding for integration

- Role Bindings
  - Connect roles and role players
  - Define how to obtain value of attribute or reference
  - Allow to create views on other classes

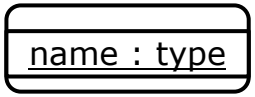
- Grounding
  - Defines which attributes/classes are represented physically
  - This is the trick! Decision about which data is derived and which is not is done at integration time!



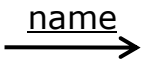
Grounding Notation



Grounded Role



Grounded Attribute



Grounded Reference

Binding Notation

Integration DSL Example

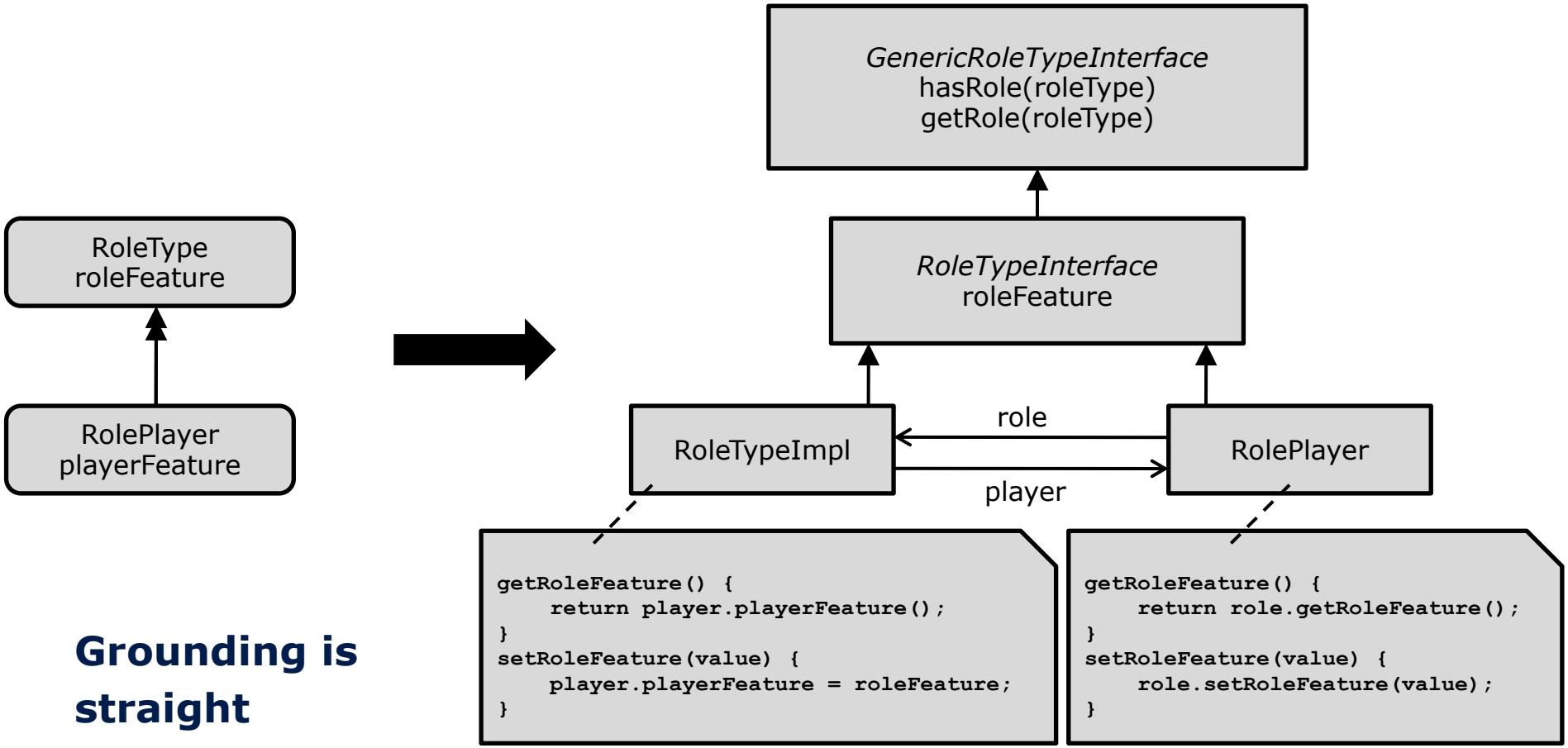
```
integrate statemachine, 2dShapes, graph {
    State plays Shape {
     label: name
     kind: if (player.type == PLAIN) return RECTANGLE
           else return CIRCLE
     colour: if (player.type == INITIAL) return WHITE
           else return BLACK
    }
    Transition plays Shape {
     label: condition
     kind: return LINE
     colour: return BLACK
    }
    State plays Node {}
    Transition plays Edge {
     source: from
     target: to
    }

    ground State { name, type }
    ground Transition { condition, from, to }
}
```

Role Binding

Grounding

Role Binding Realisation



Grounding is straight forward

Open Issues

- Data migration (if grounding evolves)
- Practical validation required

Conclusion

- Role modelling allows for unanticipated tool integration, but needs to be applied at tool design time
- Clean separation of required interface (to access tool-specific data) and realisation of this interface (to obtain data)
- Physical representation define at integration time

Thank you! Questions?

http://www.hyperadapt.net
http://www.most-project.eu

http://www.langems.org

emftext

http://www.emftext.org/language/rolecore