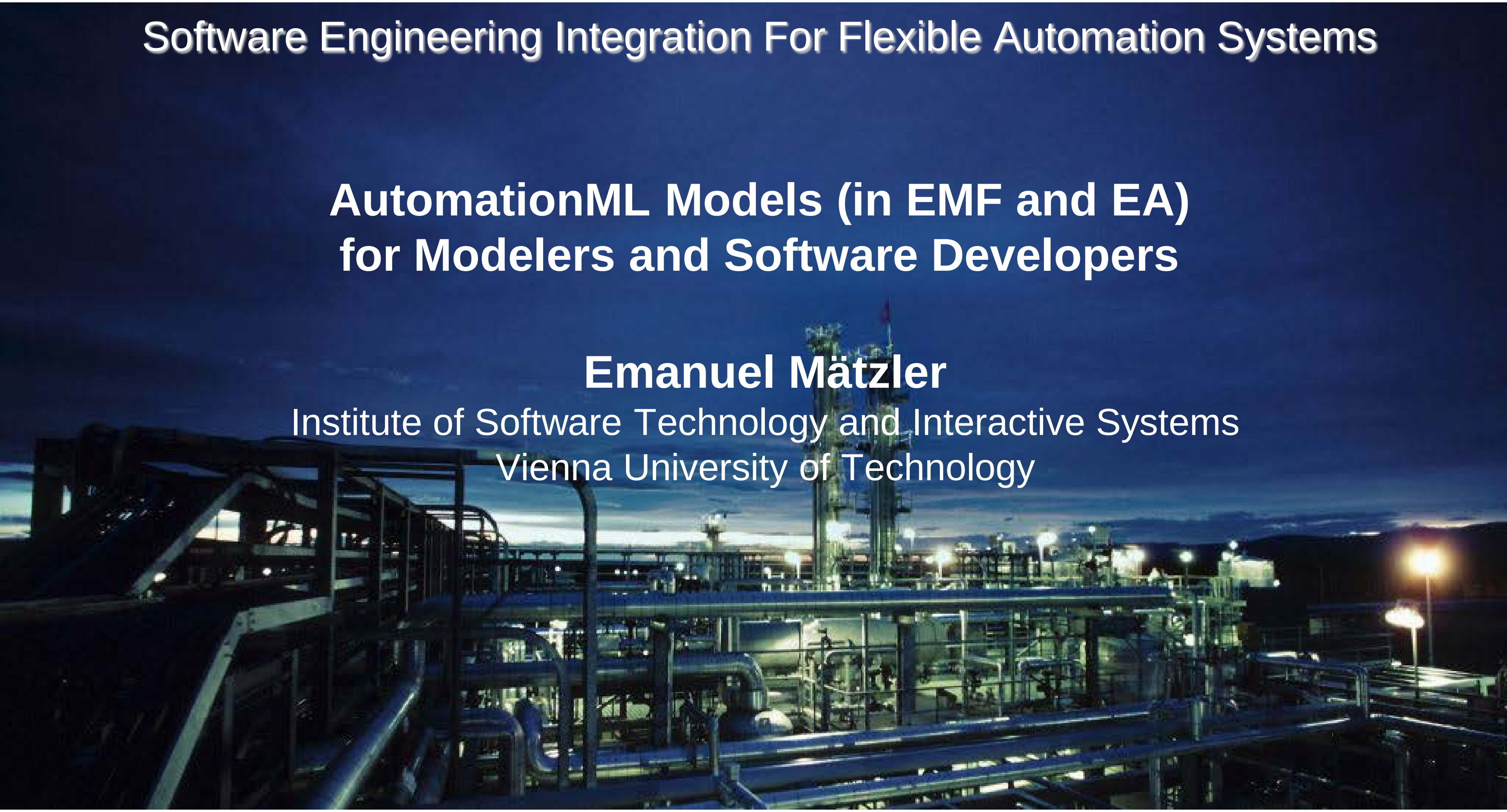
Christian Doppler Laboratory

Software Engineering Integration For Flexible Automation Systems



AutomationML Models (in EMF and EA) for Modelers and Software Developers

Emanuel Mätzler
Institute of Software Technology and Interactive Systems
Vienna University of Technology

Outline

- Introduction
- Modeling Language Engineering
  *From **Data Exchange Format to Modeling Language***
  - Background on Modeling Language Engineering
  - Interactive session with Eclipse Modeling Framework
- Model Transformation Engineering
  *From **AutomationML to Enterprise Architect (EA)***
  - Background on Model Transformations
  - Interactive session with the EA AML Engineer Plugin
- Conclusions

Introduction

- Identified needs
  a. A large number of heterogeneous models is involved in systems engineering

     → Need for process support

  b. A wide variety of models in systems engineering come from different disciplines

     → Need for management

     → Need for customized integration for concrete application scenarios

     → Vision: encompassing various viewpoints to get a better understanding

→ At TU Wien we have the expertise

a. To **develop software tools the model driven way** (AutomationML Hub example)

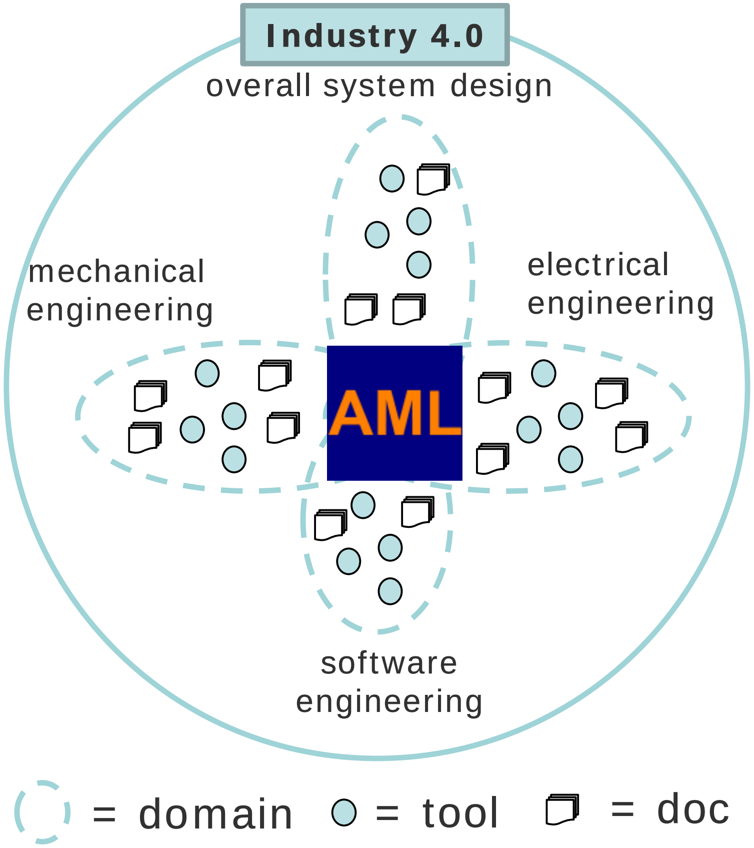
b. To **deal with a large number of heterogeneous** software and systems models

Model Exchange: AML as Common Format

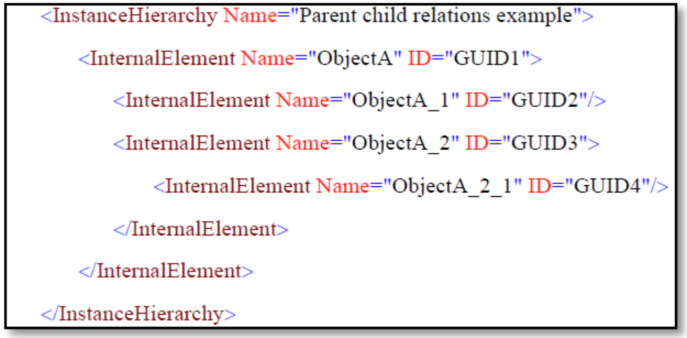
- **AutomationML (AML)**

- Emerging **standard** for **tool data exchange**

- Foundation for harmonizing engineering data coming from a heterogeneous tool network by means of a **unified format** and **data model**



Data Exchange vs. Modeling Languages



```xml
<InstanceHierarchy Name="Parent child relations example">
    <InternalElement Name="ObjectA" ID="GUID1">
        <InternalElement Name="ObjectA_1" ID="GUID2"/>
        <InternalElement Name="ObjectA_2" ID="GUID3">
            <InternalElement Name="ObjectA_2_1" ID="GUID4"/>
        </InternalElement>
    </InternalElement>
</InstanceHierarchy>
```

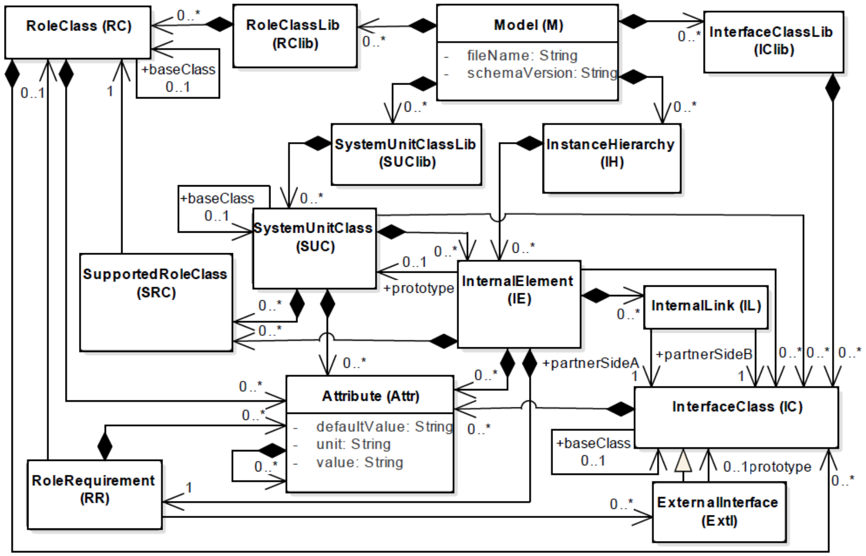
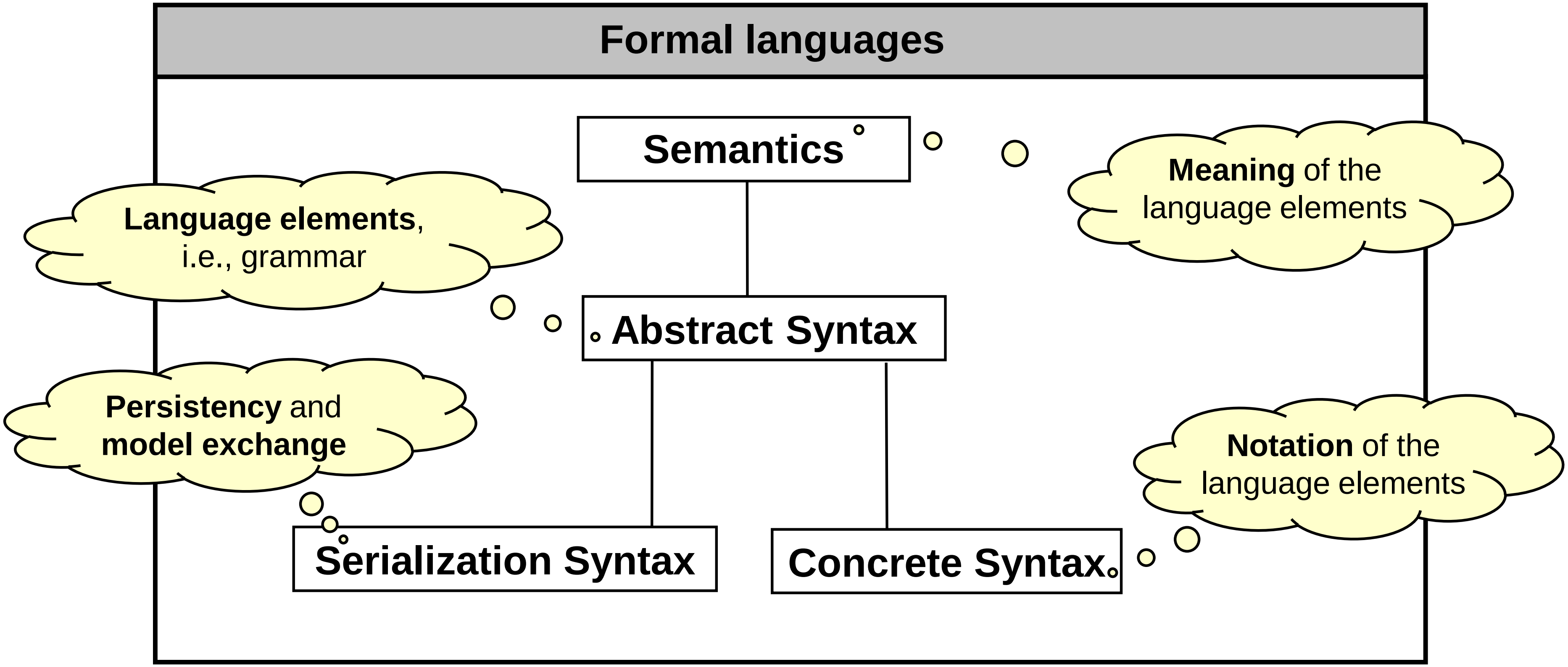


Fig. 4: AutomationML Metamodel Excerpt.

How to get from **Data** to **Models?**

Anatomy of modeling languages

- Although languages have, in general, divergent **orientations** and **application fields**, they still share a **common** language definition structure

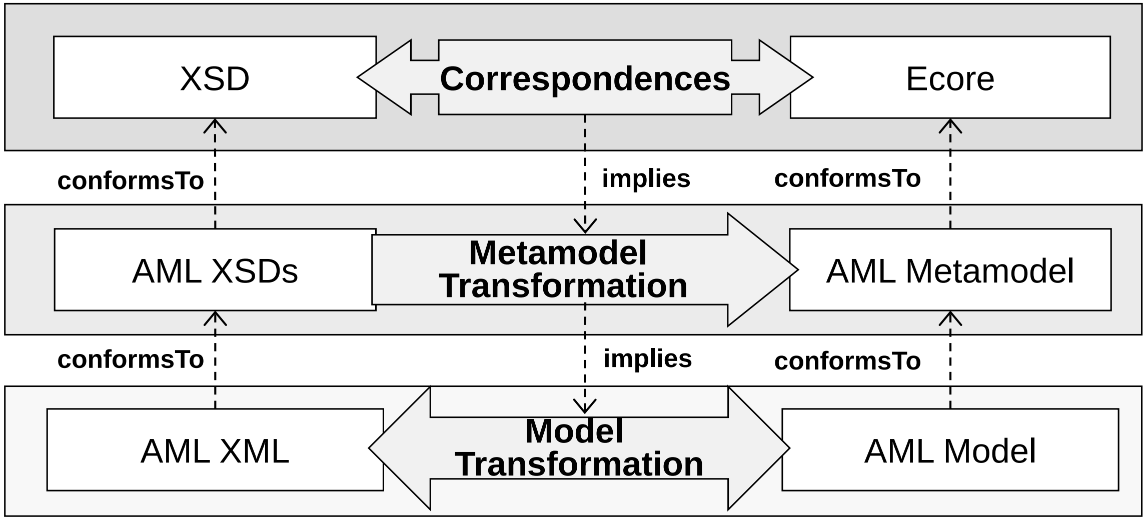

Formal languages
Semantics
Abstract Syntax
Serialization Syntax
Concrete Syntax
Language elements, i.e., grammar
Meaning of the language elements
Persistency and model exchange
Notation of the language elements

Anatomy of modeling languages

- **Semantics**: Defines the **meaning** of the language concepts
  - How language concepts are interpreted
- **Abstract syntax**: Defines the language concepts and how these concepts can be combined (~ grammar)
  - However, it **does not define** the **notation** or **meaning** of the concepts
- **Concrete syntax: Notation** to illustrate the language concepts intuitively
  - 2 ways: **textual** or **graphical** (or mixture)
- **Serialization syntax**: For persistent storage and model exchange between tools
  - XML, proprietary format, …

Language Engineering via Metamodeling

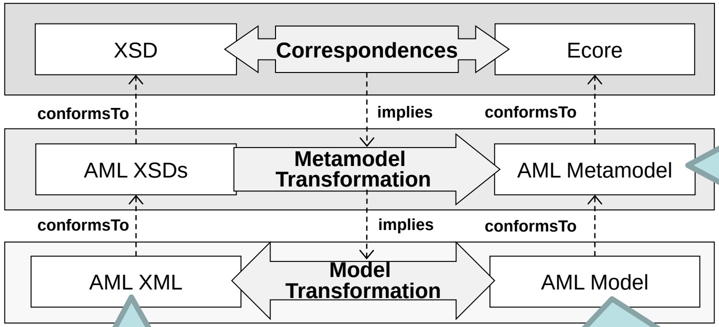
- **AutomationML** family is **defined by** a set of **XML Schemas**
- **Systematic metamodel creation process**
  - **Step 1**: **Generative** approach to produce initial Ecore-based metamodel
  - **Step 2**: **Refactorings** for improving language design
- Resulting metamodels
  - are **complete** and **correct** with respect to XML Schemas
  - allow to **import/export** data from/to XML data

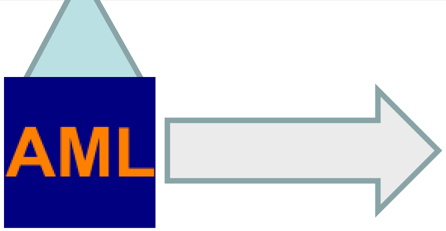


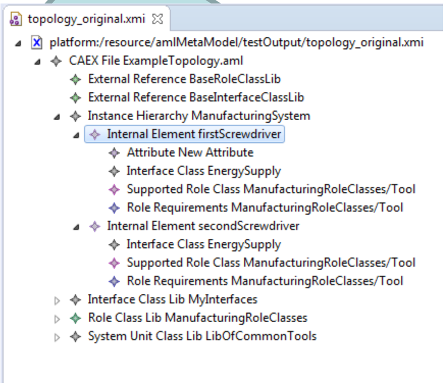
- A. Schauerhuber, M. Wimmer, E. Kapsammer, W. Schwinger, W. Retschitzegger: ***Bridging WebML to Model-Driven Engineering: From DTDs to MOF***. IET Software 1(3), 2007.

Language Engineering via Metamodeling

First Interactive session







```
topology_original.xmi ✕
◢ platform:/resource/amlMetaModel/testOutput/topology_original.xmi
  ◢ CAEX File ExampleTopology.aml
      External Reference BaseRoleClassLib
      External Reference BaseInterfaceClassLib
    ◢ Instance Hierarchy ManufacturingSystem
      ◢ Internal Element firstScrewdriver
          Attribute New Attribute
          Interface Class EnergySupply
          Supported Role Class ManufacturingRoleClasses/Tool
          Role Requirements ManufacturingRoleClasses/Tool
      ◢ Internal Element secondScrewdriver
          Interface Class EnergySupply
          Supported Role Class ManufacturingRoleClasses/Tool
          Role Requirements ManufacturingRoleClasses/Tool
    ▷ Interface Class Lib MyInterfaces
    ▷ Role Class Lib ManufacturingRoleClasses
    ▷ System Unit Class Lib LibOfCommonTools
```

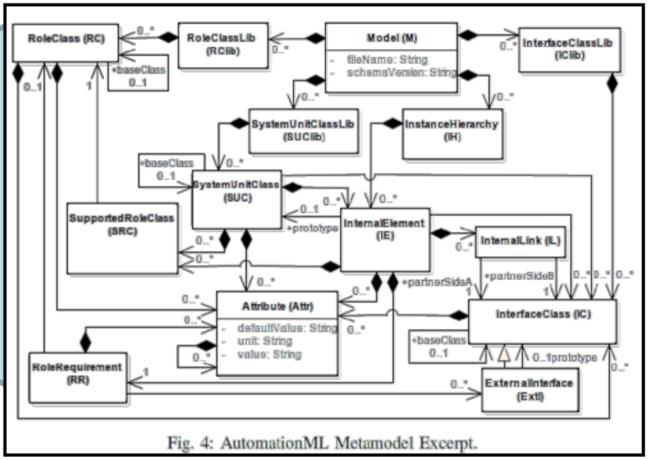


Fig. 4: AutomationML Metamodel Excerpt.

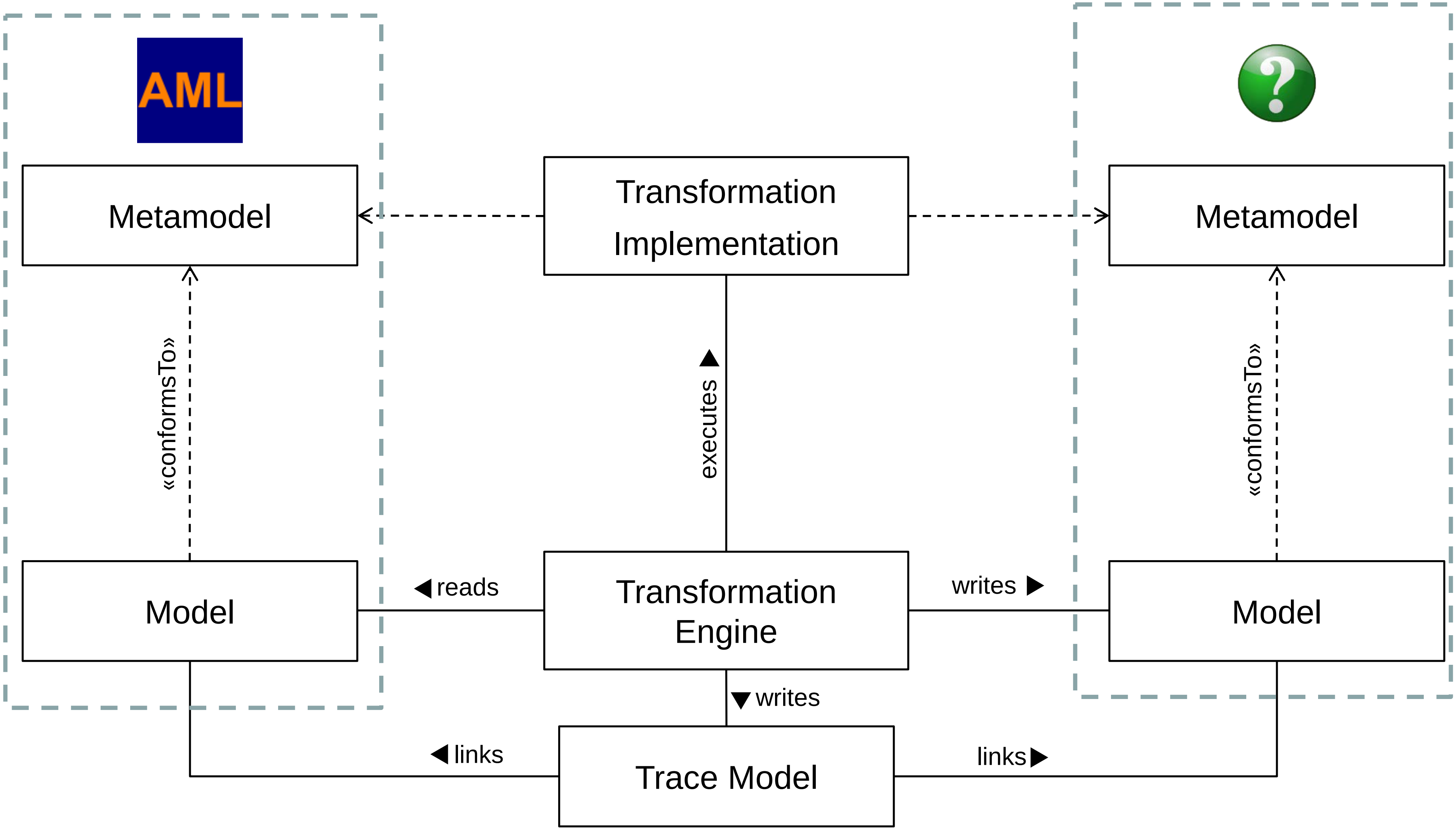
conforms to

- **Reqirements**
  - Eclipse Modeling Framework (https://eclipse.org/modeling/emf/)
  - AutomationML Metamodel (https://github.com/amlModeling)
  - Xtext for textual concrete syntax (https://eclipse.org/Xtext)
  - AmlText (https://github.com/patrickneubauer/XMLText/tree/master/AUTOMATIONML)

Interactive Session (1/2)

Model Transformations

Pattern



K. Czarnecki, S. Helsen. ***Feature-based survey of model transformation approaches***. IBM Systems Journal 45(3), pages 621-646, 2006.

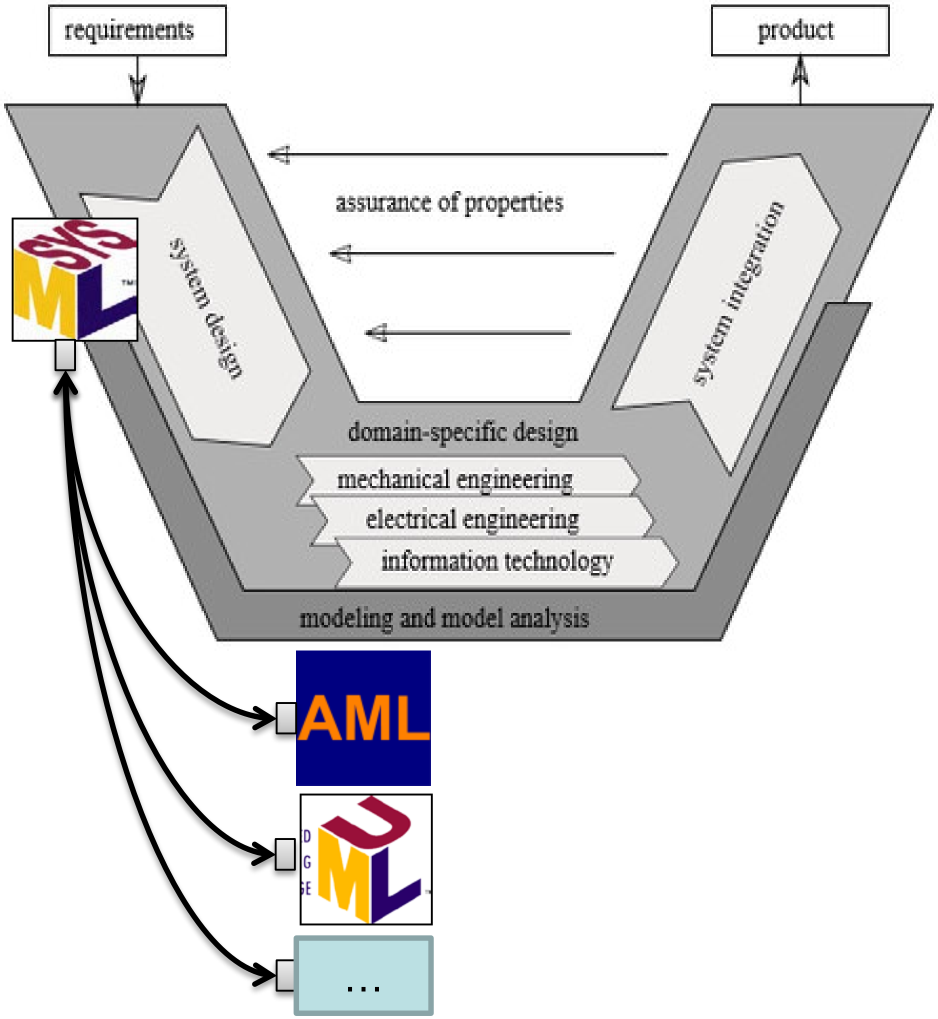
Model Transformations: AML and SysML

Two Unrelated Modeling Standards

- SysML is a graphical modeling language standardized by OMG for the development of large-scale, complex, and **multi-disciplinary systems in a model-based approach**.

- It provides modeling concepts for representing the **requirements, structure, and behavior** of a systems.

- Captures the overall design of a system on a high level of abstraction and traces this design to the discipline-specific models



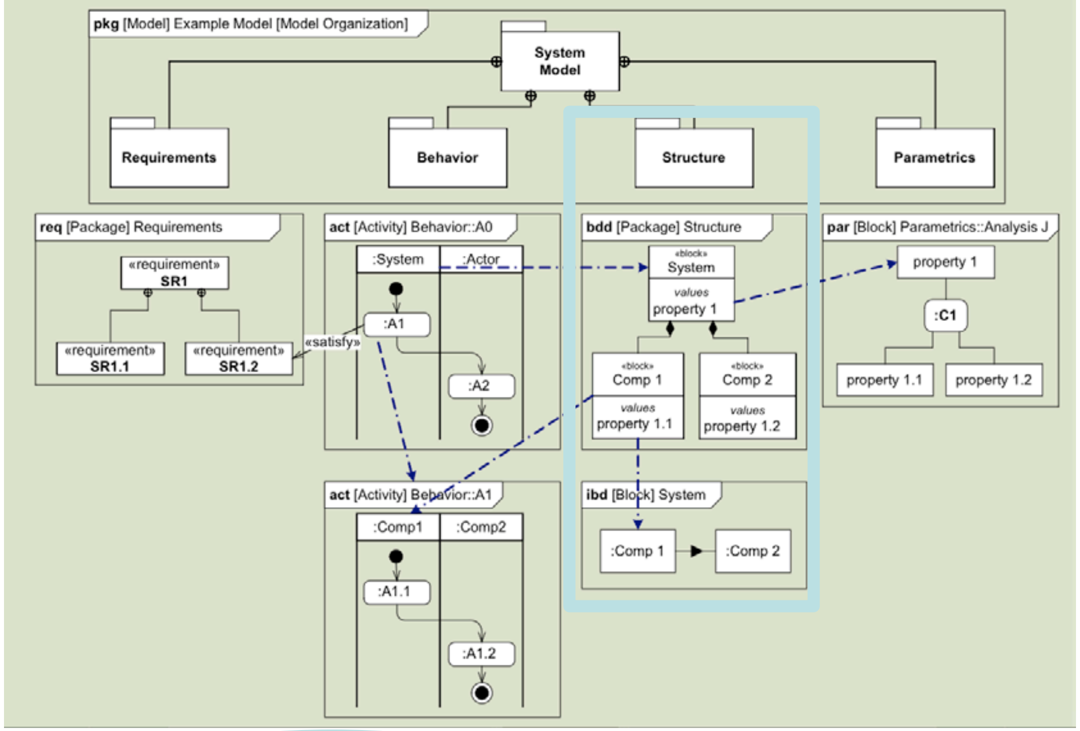
Model Transformations: AML and SysML

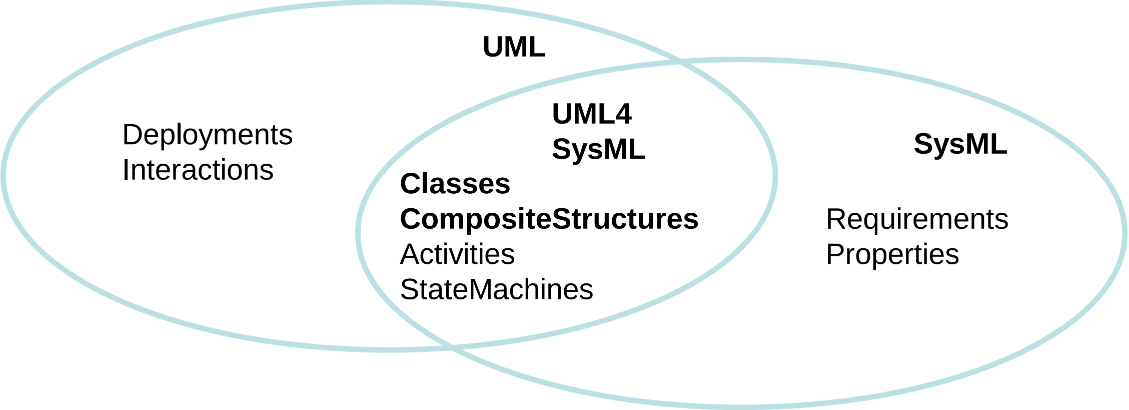
SysML in a Nutshell

- **Additions to UML** for Requirements and Properties
  - **Requirement**: SysML provides modeling constructs to represent text-based requirements and relate them to other modeling elements.
  - Constraints and Parametric Diagram (constraint analysis)
- **Customization of UML** for structural modeling through Classes and Composite Structures
  - **Block** derives from CompositeStructures::Class

«requirement»
Requirement name

text="The system shall do"
Id="62j32."





UML

Deployments
Interactions

UML4 SysML

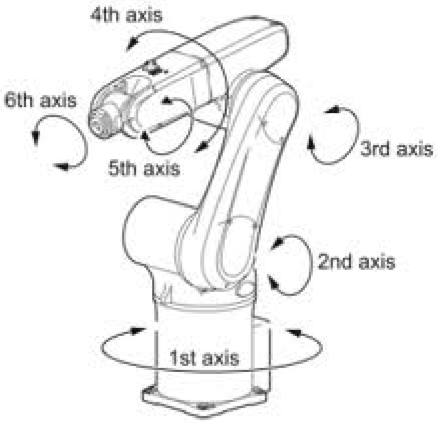
Classes
CompositeStructures
Activities
StateMachines

SysML

Requirements
Properties

Model Exchange: AML as Common Format

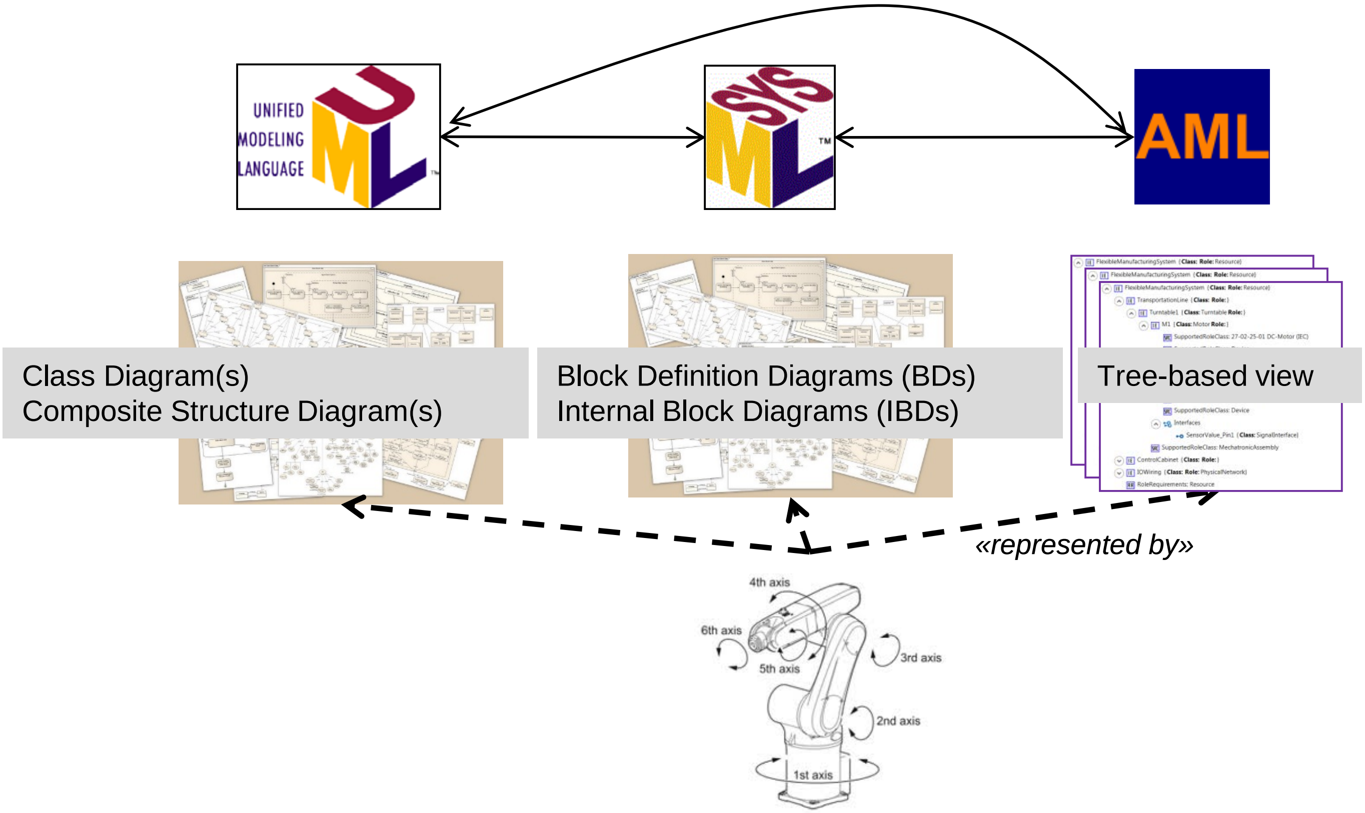
Case Study: Six Axes Robot



Model Transformations: AML and SysML

From AutomationML to Enterprise Architect and Back again: Example

- **Commonalities and differences** between the structural modeling sublanguages of AML (CAEX) and SysML (Block Diagrams)
- **AML metamodel and profiles** for UML and SysML
- **Transformations** between AML and SysML (UML/SysML already available)



Class Diagram(s)
Composite Structure Diagram(s)

Block Definition Diagrams (BDs)
Internal Block Diagrams (IBDs)

Tree-based view

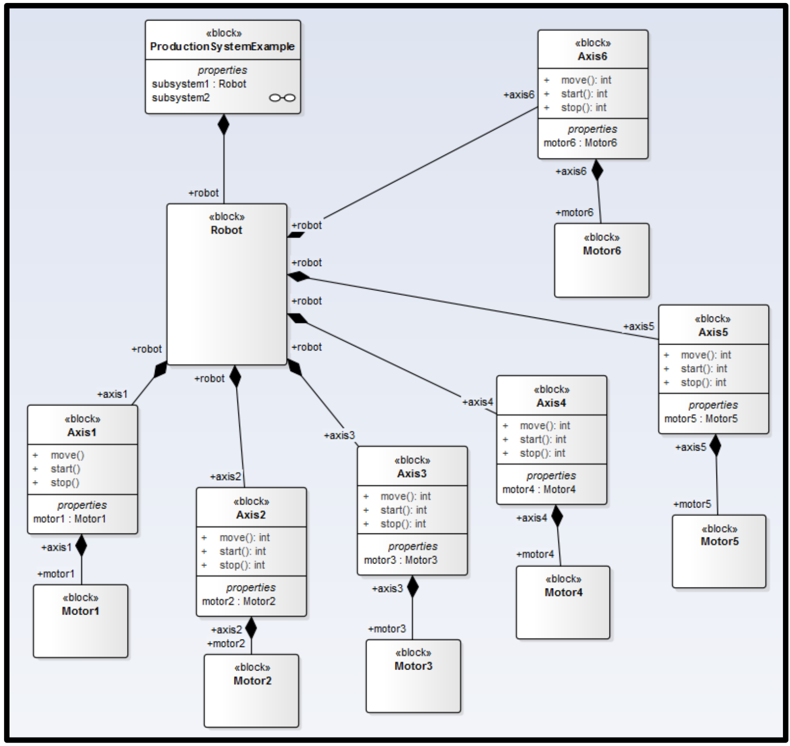
«represented by»

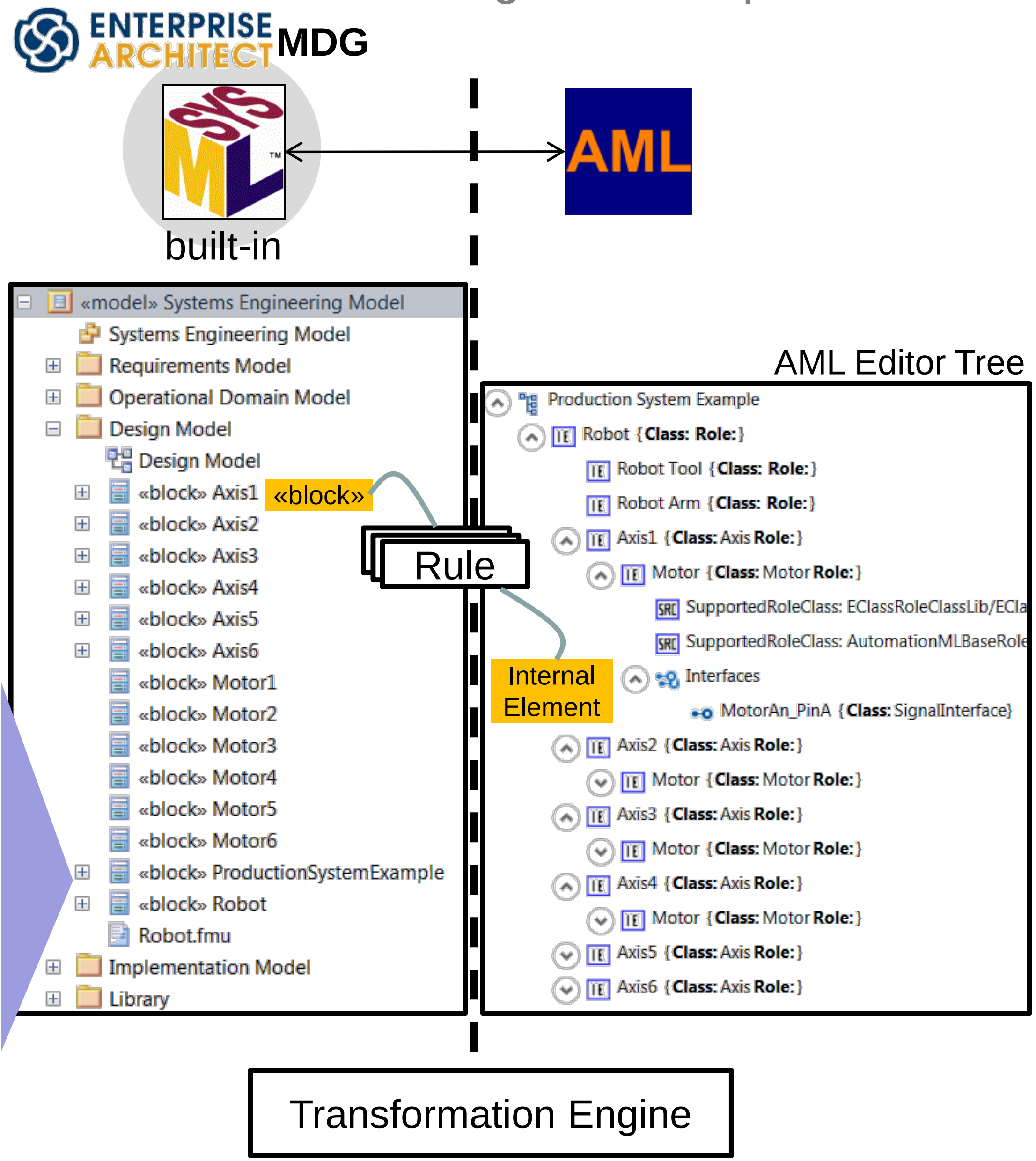
UNIFIED
MODELING
LANGUAGE
UML
SYS ML
AML
FlexibleManufacturingSystem {Class: Role: Resource}
FlexibleManufacturingSystem {Class: Role: Resource}
FlexibleManufacturingSystem {Class: Role: Resource}
TransportationLine {Class: Role:}
Turntable1 {Class: Turntable Role:}
M1 {Class: Motor Role:}
SupportedRoleClass: 27-02-25-01 DC-Motor (IEC)
SupportedRoleClass: Device
Interfaces
SensorValue_Pin1 {Class: SignalInterface}
SupportedRoleClass: MechatronicAssembly
ControlCabinet {Class: Role:}
IOWiring {Class: Role: PhysicalNetwork}
RoleRequirements: Resource
4th axis
6th axis
5th axis
3rd axis
2nd axis
1st axis

Model Transformations: AML and SysML

From AutomationML to Enterprise Architect and Back again: Example

Block Definition Diagram





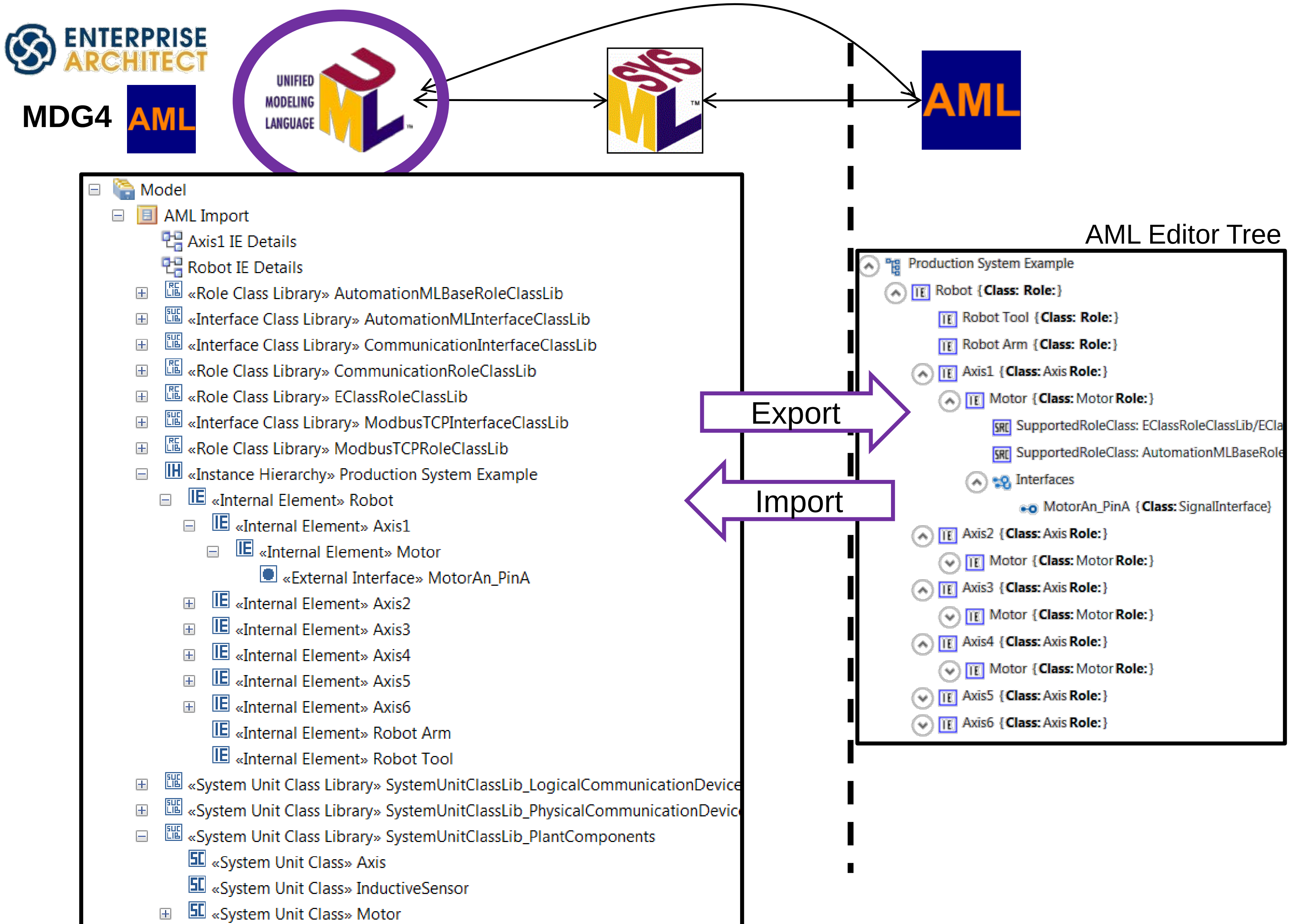
AML Editor Tree

- «model» Systems Engineering Model
  - Systems Engineering Model
  - Requirements Model
  - Operational Domain Model
  - Design Model
    - Design Model
    - «block» Axis1
    - «block» Axis2
    - «block» Axis3
    - «block» Axis4
    - «block» Axis5
    - «block» Axis6
    - «block» Motor1
    - «block» Motor2
    - «block» Motor3
    - «block» Motor4
    - «block» Motor5
    - «block» Motor6
    - «block» ProductionSystemExample
    - «block» Robot
    - Robot.fmu
  - Implementation Model
  - Library

- Production System Example
  - Robot {**Class: Role:**}
    - Robot Tool {**Class: Role:**}
    - Robot Arm {**Class: Role:**}
    - Axis1 {**Class:** Axis **Role:**}
      - Motor {**Class:** Motor **Role:**}
        - SupportedRoleClass: EClassRoleClassLib/ECla
        - SupportedRoleClass: AutomationMLBaseRole
        - Interfaces
          - MotorAn_PinA {**Class:** SignalInterface}
    - Axis2 {**Class:** Axis **Role:**}
      - Motor {**Class:** Motor **Role:**}
    - Axis3 {**Class:** Axis **Role:**}
      - Motor {**Class:** Motor **Role:**}
    - Axis4 {**Class:** Axis **Role:**}
      - Motor {**Class:** Motor **Role:**}
    - Axis5 {**Class:** Axis **Role:**}
    - Axis6 {**Class:** Axis **Role:**}

Model Transformations: AML and SysML

From AutomationML to Enterprise Architect and Back again: Example

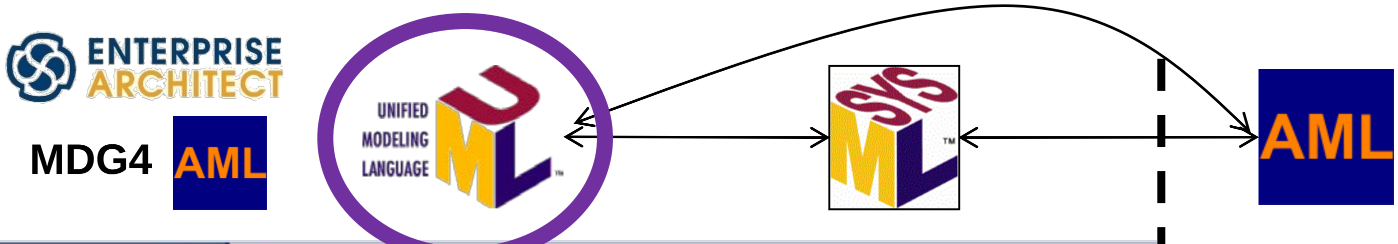


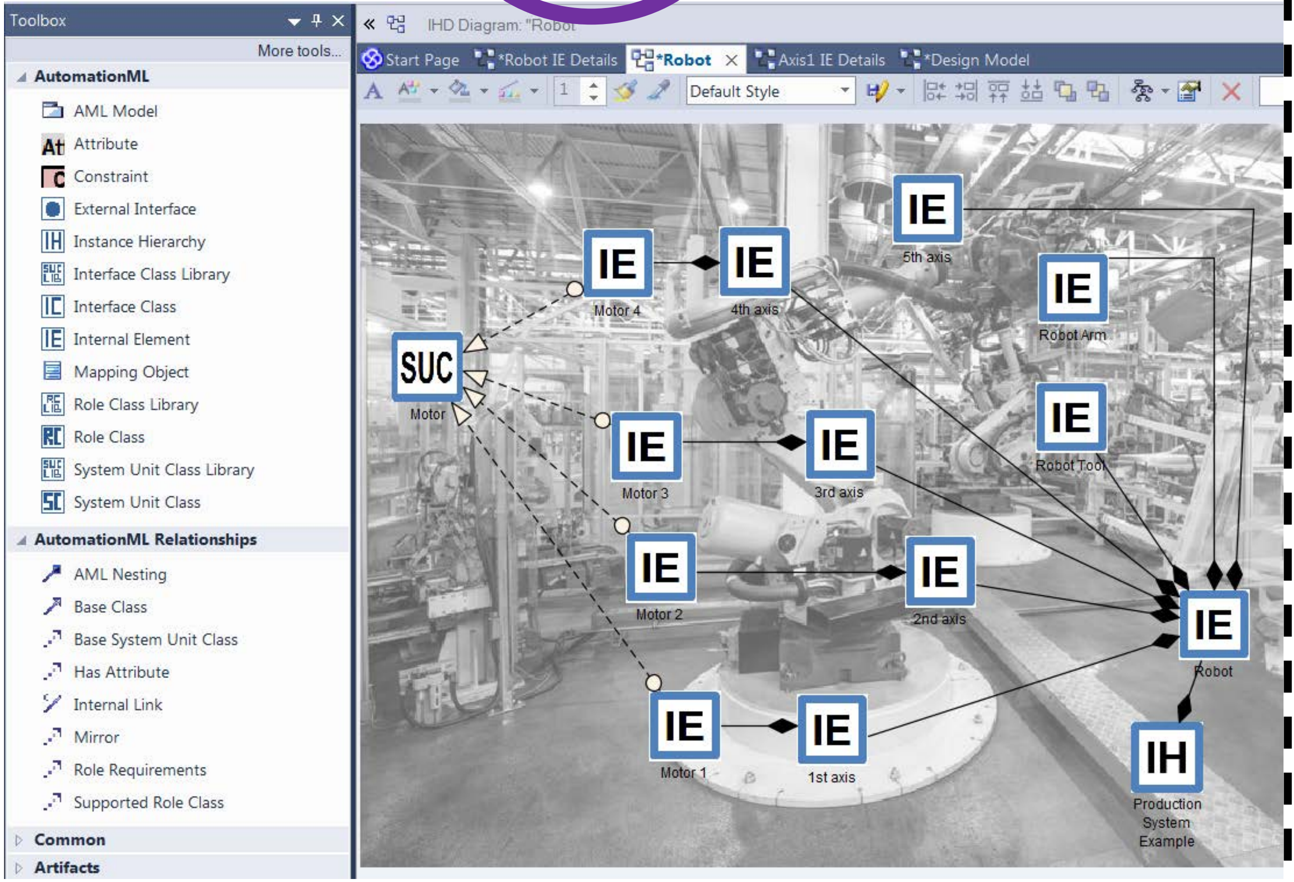
```
⊟ Model
  ⊟ AML Import
      Axis1 IE Details
      Robot IE Details
    ⊞ «Role Class Library» AutomationMLBaseRoleClassLib
    ⊞ «Interface Class Library» AutomationMLInterfaceClassLib
    ⊞ «Interface Class Library» CommunicationInterfaceClassLib
    ⊞ «Role Class Library» CommunicationRoleClassLib
    ⊞ «Role Class Library» EClassRoleClassLib
    ⊞ «Interface Class Library» ModbusTCPInterfaceClassLib
    ⊞ «Role Class Library» ModbusTCPRoleClassLib
    ⊟ «Instance Hierarchy» Production System Example
      ⊟ «Internal Element» Robot
        ⊟ «Internal Element» Axis1
          ⊟ «Internal Element» Motor
                «External Interface» MotorAn_PinA
        ⊞ «Internal Element» Axis2
        ⊞ «Internal Element» Axis3
        ⊞ «Internal Element» Axis4
        ⊞ «Internal Element» Axis5
        ⊞ «Internal Element» Axis6
          «Internal Element» Robot Arm
          «Internal Element» Robot Tool
    ⊞ «System Unit Class Library» SystemUnitClassLib_LogicalCommunicationDevice
    ⊞ «System Unit Class Library» SystemUnitClassLib_PhysicalCommunicationDevic
    ⊟ «System Unit Class Library» SystemUnitClassLib_PlantComponents
        «System Unit Class» Axis
        «System Unit Class» InductiveSensor
      ⊞ «System Unit Class» Motor
```

```
Production System Example
  Robot {Class: Role:}
      Robot Tool {Class: Role:}
      Robot Arm {Class: Role:}
    Axis1 {Class: Axis Role:}
      Motor {Class: Motor Role:}
          SupportedRoleClass: EClassRoleClassLib/ECla
          SupportedRoleClass: AutomationMLBaseRole
        Interfaces
            MotorAn_PinA {Class: SignalInterface}
    Axis2 {Class: Axis Role:}
      Motor {Class: Motor Role:}
    Axis3 {Class: Axis Role:}
      Motor {Class: Motor Role:}
    Axis4 {Class: Axis Role:}
      Motor {Class: Motor Role:}
    Axis5 {Class: Axis Role:}
    Axis6 {Class: Axis Role:}
```

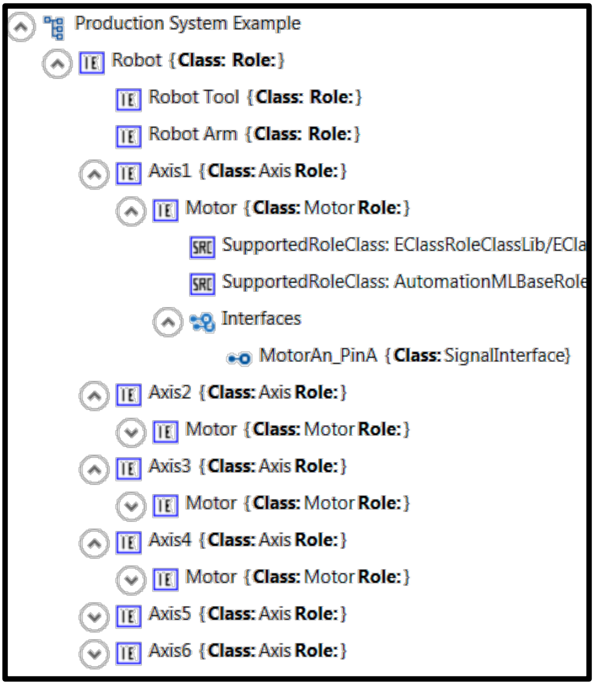
Model Transformations: AML and SysML

From AutomationML to Enterprise Architect and Back again: Example





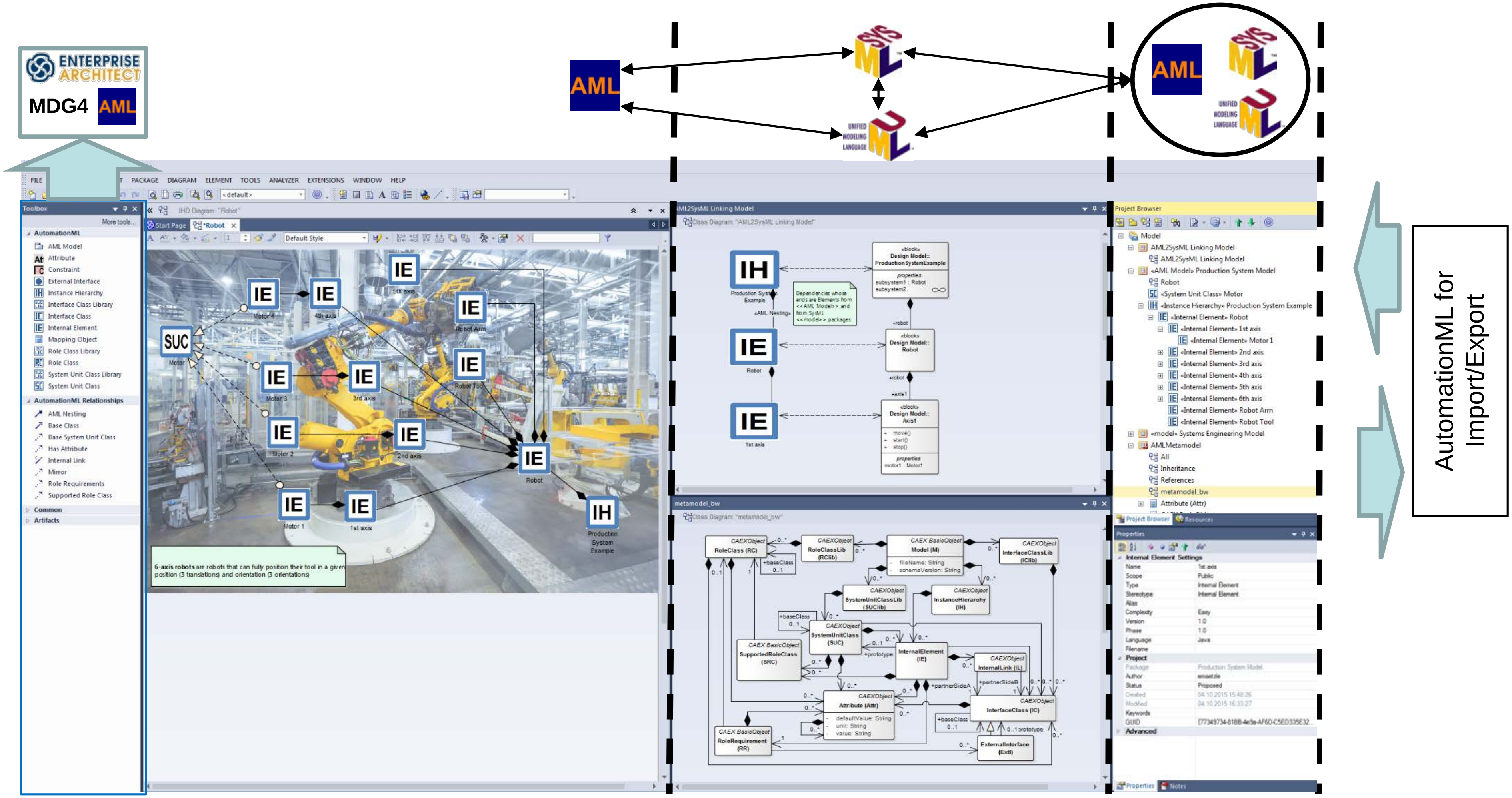
AML Editor Tree



- Production System Example
  - Robot {Class: Role:}
    - Robot Tool {Class: Role:}
    - Robot Arm {Class: Role:}
    - Axis1 {Class: Axis Role:}
      - Motor {Class: Motor Role:}
        - SupportedRoleClass: EClassRoleClassLib/ECla
        - SupportedRoleClass: AutomationMLBaseRole
        - Interfaces
          - MotorAn_PinA {Class: SignalInterface}
    - Axis2 {Class: Axis Role:}
      - Motor {Class: Motor Role:}
    - Axis3 {Class: Axis Role:}
      - Motor {Class: Motor Role:}
    - Axis4 {Class: Axis Role:}
      - Motor {Class: Motor Role:}
    - Axis5 {Class: Axis Role:}
    - Axis6 {Class: Axis Role:}

Model Transformations: AML, SysML and UML

Second Interactive session: From AutomationML to EA and Back again



6-axis robots are robots that can fully position their tool in a given position (3 translations) and orientation (3 orientations)

Dependencies whose ends are Elements from <<AML Model>> and from SysML <<model>> packages.

Project Browser

- Model
  - AML2SysML Linking Model
    - AML2SysML Linking Model
  - «AML Model» Production System Model
    - Robot
    - «System Unit Class» Motor
    - «Instance Hierarchy» Production System Example
      - «Internal Element» Robot
        - «Internal Element» 1st axis
          - «Internal Element» Motor 1
        - «Internal Element» 2nd axis
        - «Internal Element» 3rd axis
        - «Internal Element» 4th axis
        - «Internal Element» 5th axis
        - «Internal Element» 6th axis
        - «Internal Element» Robot Arm
        - «Internal Element» Robot Tool
  - «model» Systems Engineering Model
  - AMLMetamodel
    - All
    - Inheritance
    - References
    - metamodel_bw
    - Attribute (Attr)

Properties

Internal Element Settings	
Name	1st axis
Scope	Public
Type	Internal Element
Stereotype	Internal Element
Alias	
Complexity	Easy
Version	1.0
Phase	1.0
Language	Java
Filename	
Project	
Package	Production System Model
Author	emaetzle
Status	Proposed
Created	04.10.2015 15:48:26
Modified	04.10.2015 16:33:27
Keywords	
GUID	{77349734-81BB-4e3a-AF6D-C5ED335E32...
Advanced	

- **Reqirements**
  - Enterprise Architect (http://www.sparxsystems.de/uml/download-trial/)
  - AML Plugin for EA (Prototype: http://www.sysml4industry.org/?page_id=266)

Interactive Session (2/2)

Conclusions and Future Work

- **Model-Driven Engineering** is beneficial to
  - Represent modeling languages
  - Derive tool support
  - Bridging different languages
  - Providing different surface languages for one abstract language

- Resulting **modeling tools** are
  - **Open** and **extensible**
  - **Model management** support is available out-of-the-box based on common metamodeling language
  - Modeling tools are **usable in combination** based on model exchange
  - Modeling tools **allow for a mixture of modeling languages** leading to multi-paradigm modeling approaches

- **Next steps**
  - **Mappings** between the **behavioral modeling parts** of AML PLCopen XML and SysML Activity Diagrams and State Machines
  - **Generative usage** of AML models by defining **code generator** chains
  - **Analytical usage** of AML models by transforming them to **formal domains**