Business Informatics Group

Vienna University of Technology

Multi-Level Modeling in the Wild with AutomationML

Invited Talk @ Multi Level Modeling Workshop, MODELS 2018
Copenhagen, October 2018

Manuel Wimmer

CDL-MINT
Business Informatics Group
Institute of Information Systems Engineering
Vienna University of Technology

Favoritenstraße 9-11/188-3, 1040 Vienna, Austria
phone: +43 (1) 58801-18804 (secretary), fax: +43 (1) 58801-18896
office@big.tuwien.ac.at, www.big.tuwien.ac.at

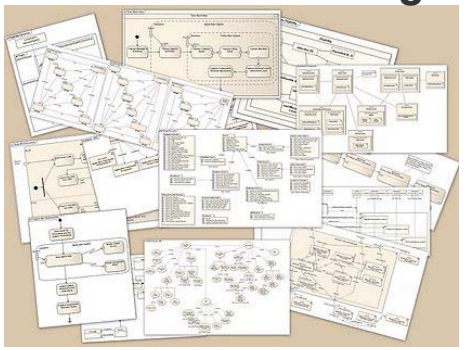
Preview: What this talk is about?

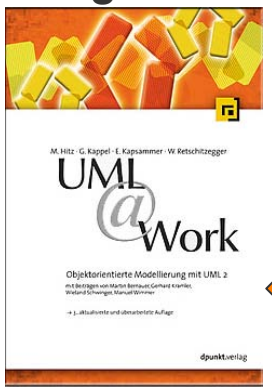
My Background
&
Ongoing Work

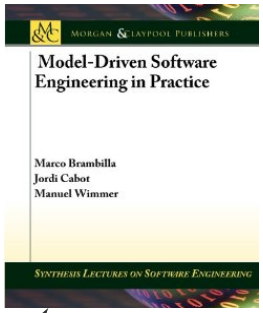
My Background

Data Engineering

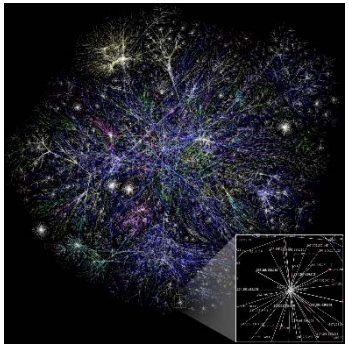
Model Engineering





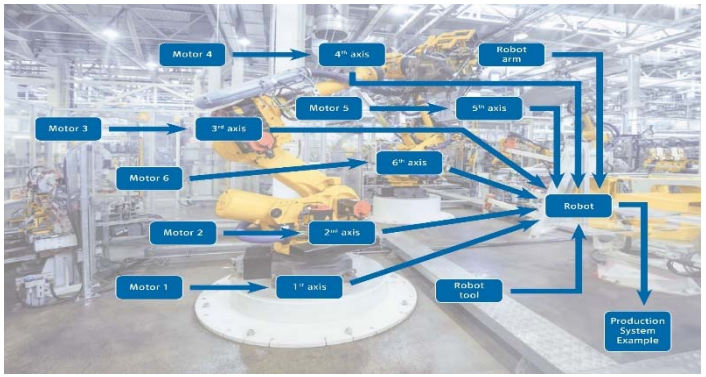


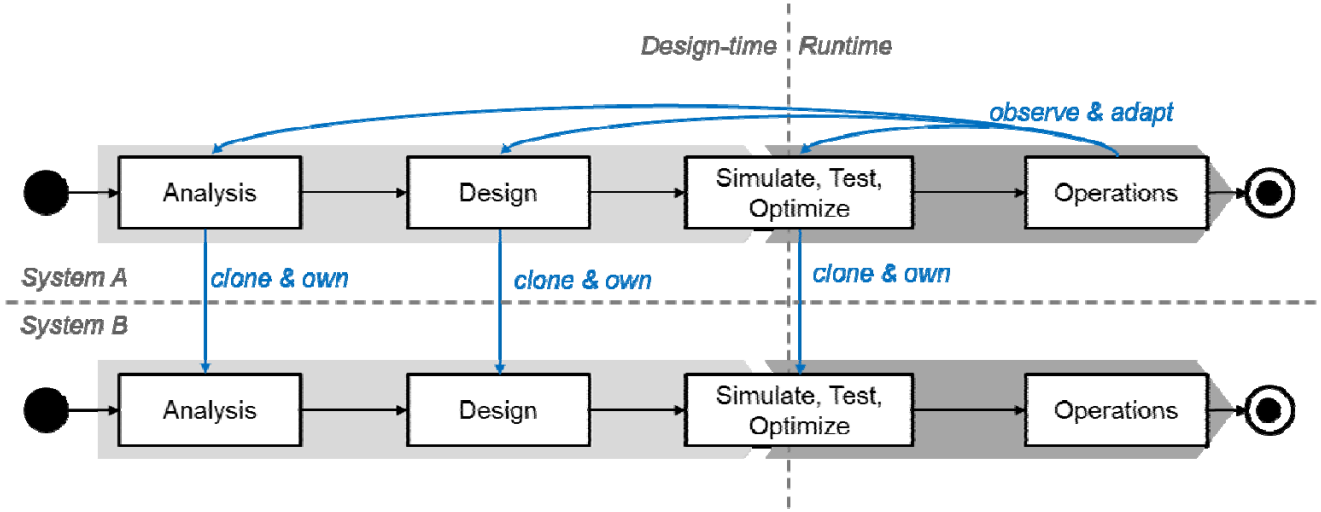
Web Engineering



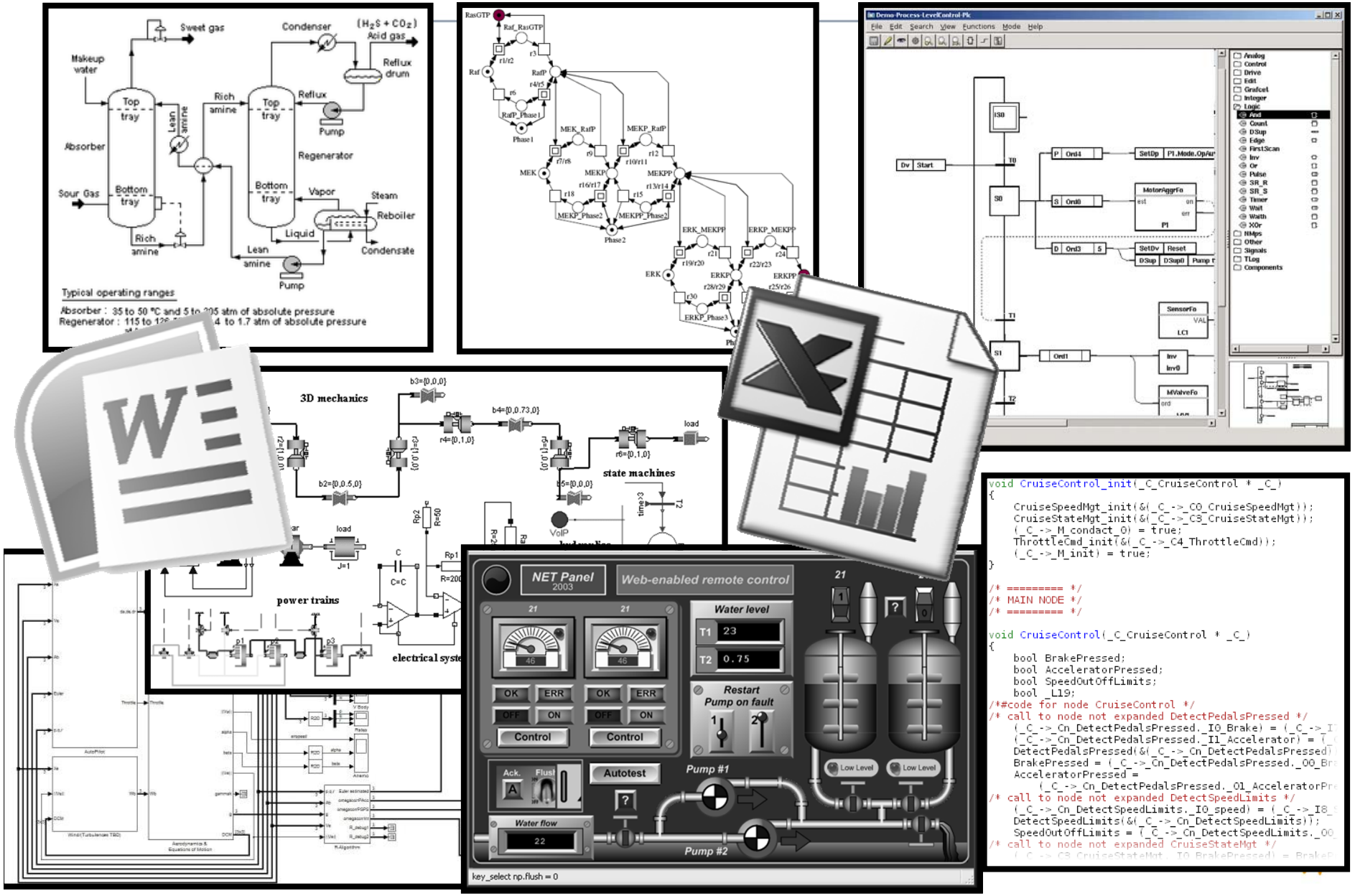
Industrial Engineering

Christian Doppler Laboratory
Model-Integrated Smart Production (CDL-MINT)



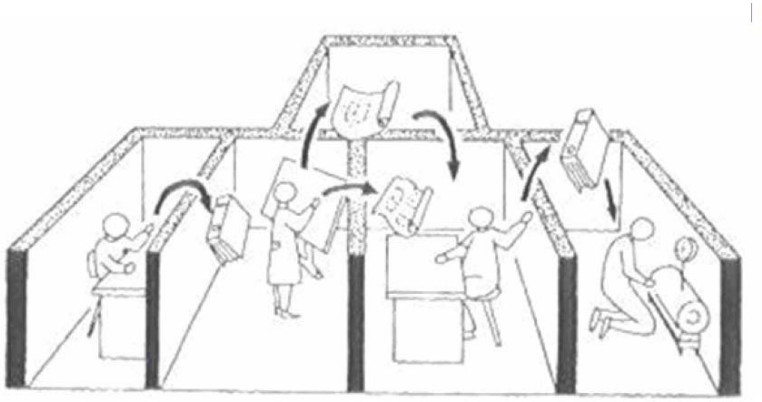


Artefacts in CPPS Engineering Process

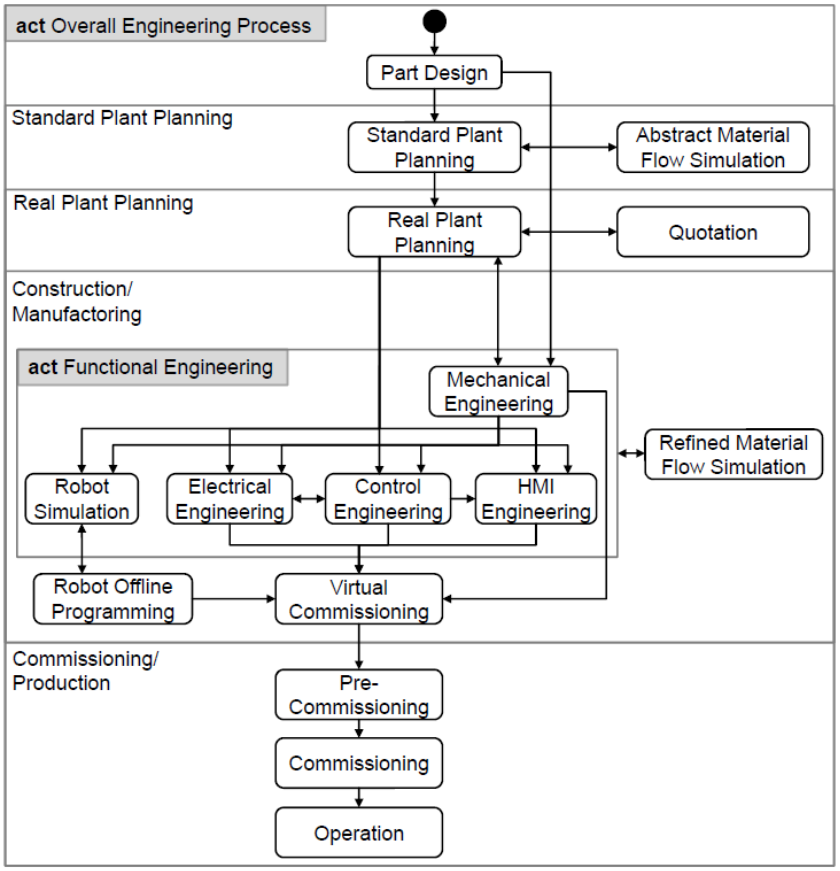


Some Challenges in CPPS (1/2)

- **Various engineering disciplines** are involved in the engineering process



- Solutions often **lack support** for…
  - **Model exchange**
  - Model versioning
  - Model linking
  - Model consistency
  - …



Typical industrial plant engineering process taken from: R. Drath, B. Schröter, and M. Hoernicke, "***Datenkonsistenz im Umfeld heterogener Engineering-Werkzeuge***", in Proc. of the Automation Conference, 2011, pp. 29-32.

Some Challenges in CPPS (2/2)

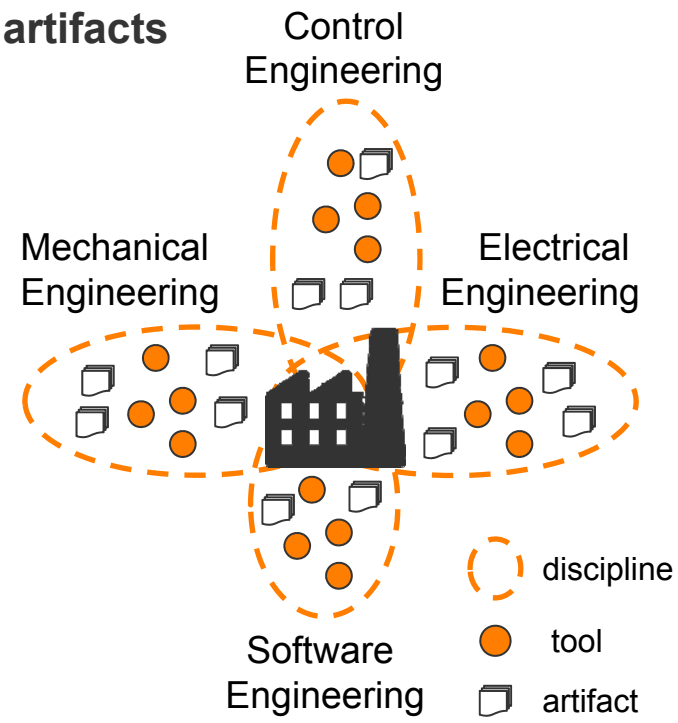
- **Multidisciplinary** process involving various engineering disciplines
  - Different disciplines and engineering activities require **specialized software tools** and produce **specific artifacts**

  > Heterogeneous engineering tool landscape

- **Multi-staged** and **parallel** process
  - Different engineering activities depend on each other, i.e., require results of other engineering activities

  > Need for a lossless data exchange

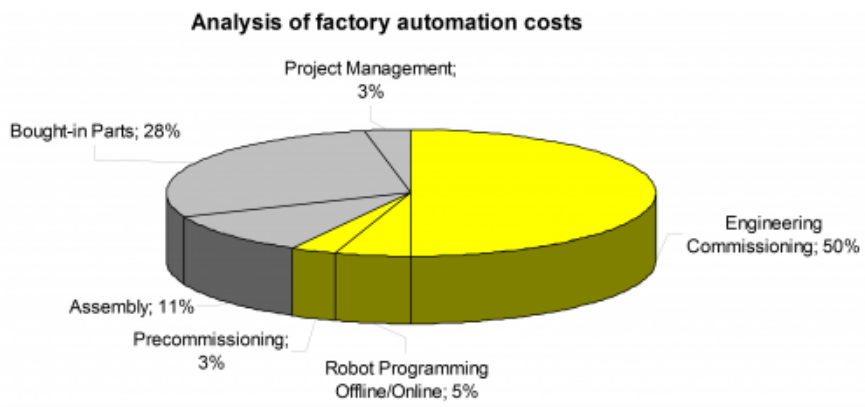
- **How is the data exchange achieved?**
  - PDFs, printouts, manual transfer

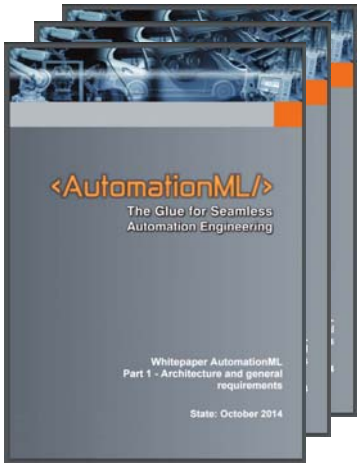


> Inefficient and error-prone process leading to substantial costs
> Current movements towards smart production aggravate this problem

AutomationML Association

- **Standardize data exchange** in the engineering process of production systems





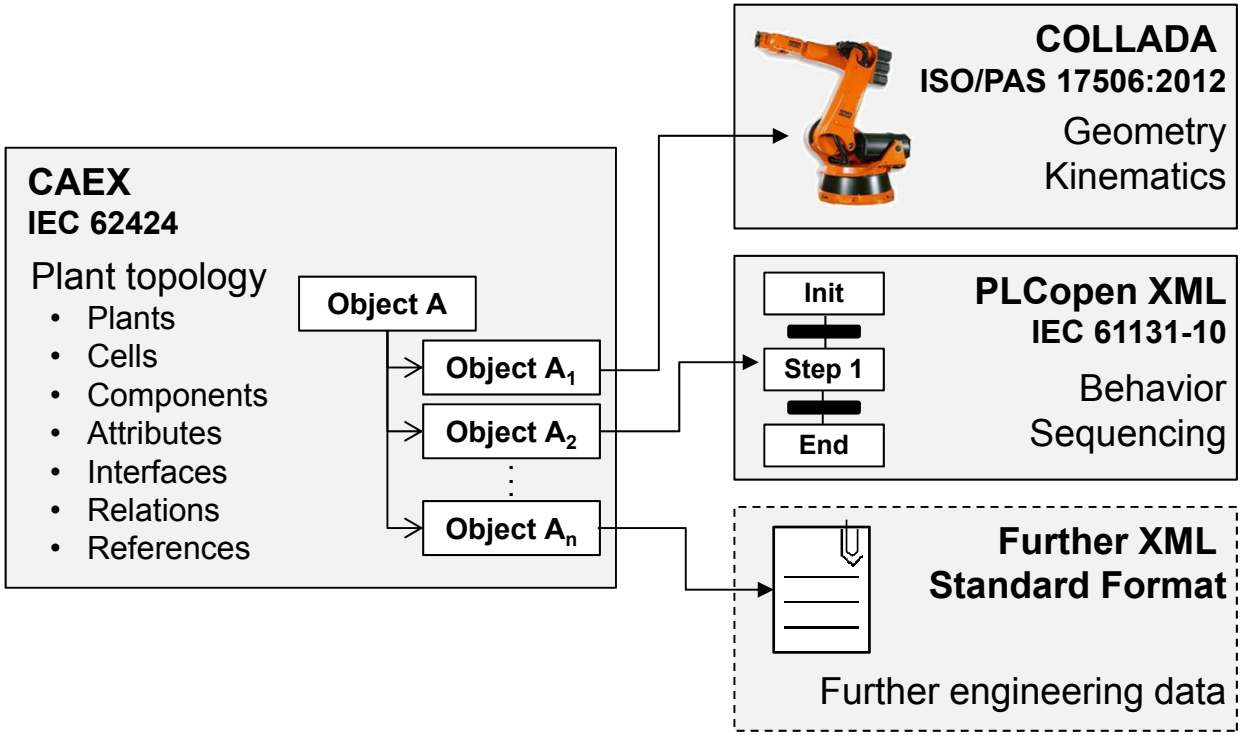
- Whitepapers
- Application Recommendations
- Best Practice Recommendations

- Large **industrial** and **academic** consortium (over 50 members)

- AutomationML website: http://www.automationml.org
- IEC 62714 - Engineering data exchange format for use in industrial automation systems engineering - AutomationML, www.iec.ch, 2014.

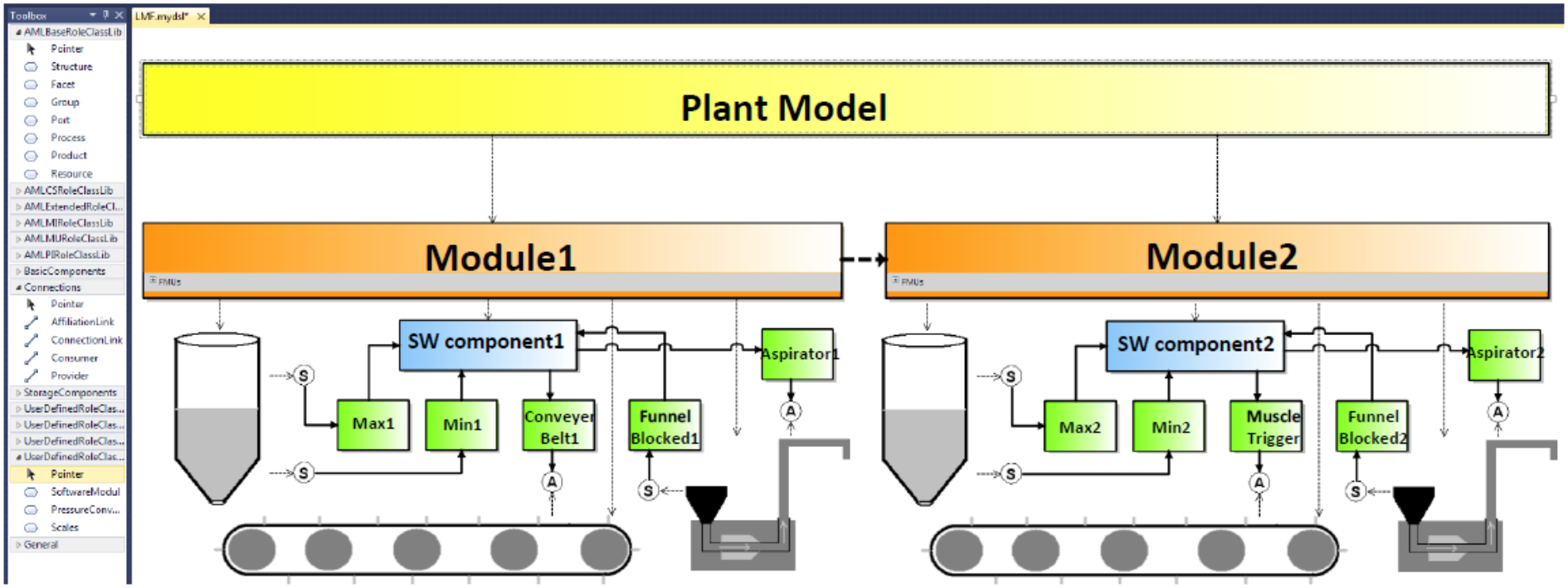
Data Exchange: AutomationML (AML) as Common Format

- Foundation for harmonizing engineering data coming from heterogeneous tool networks by means of a unified format and data model
- **Collection and integration of existing standards**



AutomationML in a Nutshell
https://www.automationml.org/o.red/uploads/dateien/1447420977-AutomationML%20in%20a%20Nutshell_151104.pdf

AutomationML =
Automation (Markup | Modeling) Language?



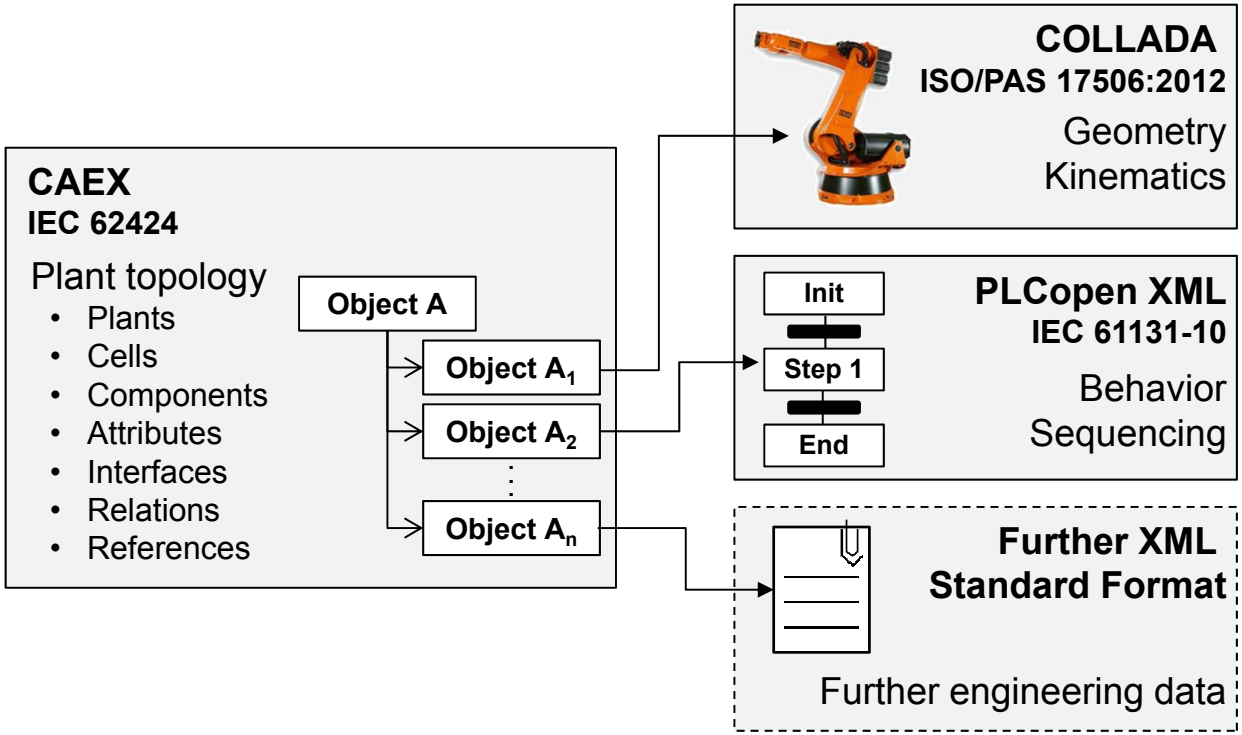
S. Faltinski, O. Niggemann, N. Moriz, A. Mankowski***: AutomationML: From data exchange to system planning and simulation***, in Proc. of ICIT, 2012, pp. 378–383.

<AutomationML/>

A Model-Driven Perspective

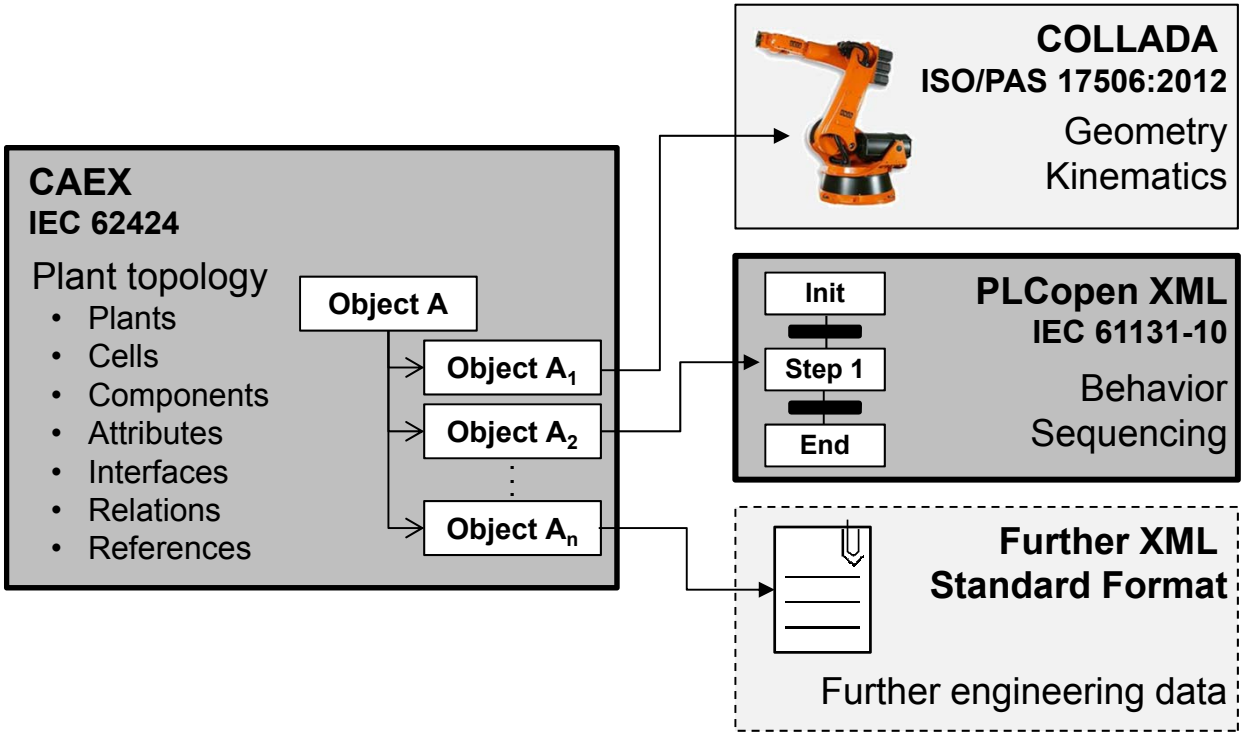
AutomationML: Automation Markup Language

- **Open, vendor neutral, free XML format** for the storage and exchange of plant engineering data
- Integrates, extends and adapts **existing standard XML data formats**



AutomationML: Automation Modeling Language?

- How can **we** best **support AutomationML?**
  - **Language engineering viewpoint**
  - Production systems engineering viewpoint



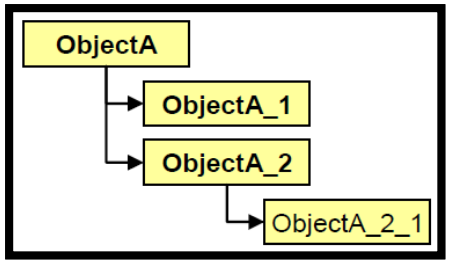
<AutomationML/>

CAEX

IEC 62424

AutomationML = Automation (Markup | Modeling) Language?

- **Object-Oriented Format**
  - *Automation object: physical or logical entity in the automated system*
- **Currently it is a Tree-Based Format**
  - *Plant topology information: The plant topology acts as the top-level data structure of the plant engineering information and shall be modelled by means of the data format CAEX according to IEC 62424:2008, Clause 7, Annex A and Annex C. Semantic extensions of CAEX are described separately.* ***Multiple and crossed hierarchy structures shall be used by means of the mirror object concept*** *according to IEC 62424:2008, A.2.14. Mirror objects shall not be modified; all changes shall be done at the master object.*

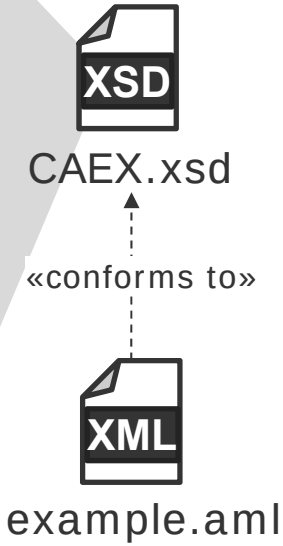



```
<InstanceHierarchy Name="Parent child relations example">
    <InternalElement Name="ObjectA" ID="GUID1">
        <InternalElement Name="ObjectA_1" ID="GUID2"/>
        <InternalElement Name="ObjectA_2" ID="GUID3">
            <InternalElement Name="ObjectA_2_1" ID="GUID4"/>
        </InternalElement>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

CAEX: Language Realization (1/3)

- **CAEX** is a language which is **specified with XSD**

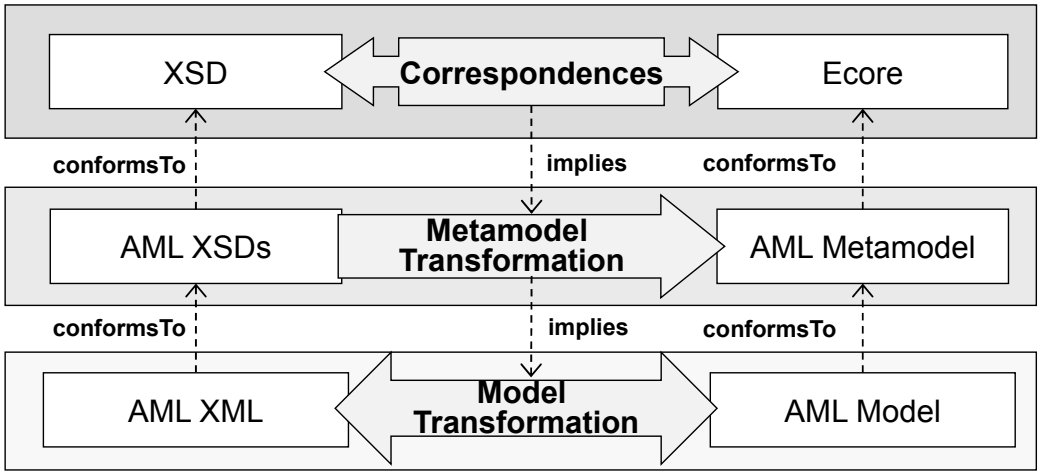
```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- CAEX - Computer Aided Engineering Data-Exchange-Metamodel -->
<!-- Version 2.15, 16.05.2007 -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xs:simpleType name="ChangeMode">
    <xs:annotation>
      <xs:documentation>Optionally describes the change state of a CAEX object. If used, the ChangeMode
shall have the following value range: state, create, delete and change. This information should be used for further
change management applications.</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:enumeration value="state"/>
      <xs:enumeration value="create"/>
      <xs:enumeration value="delete"/>
      <xs:enumeration value="change"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:group name="Header">
    <xs:annotation>
      <xs:documentation>Defines a group of organizational information, like description, version, revision,
copyright, etc.</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="Description" minOccurs="0">
        <xs:annotation>
          <xs:documentation>Textual description for CAEX objects.</xs:documentation>
        </xs:annotation>
```



technology **<xml />** **JAXP** **Xerces**

CAEX: Language Realization (2/3)

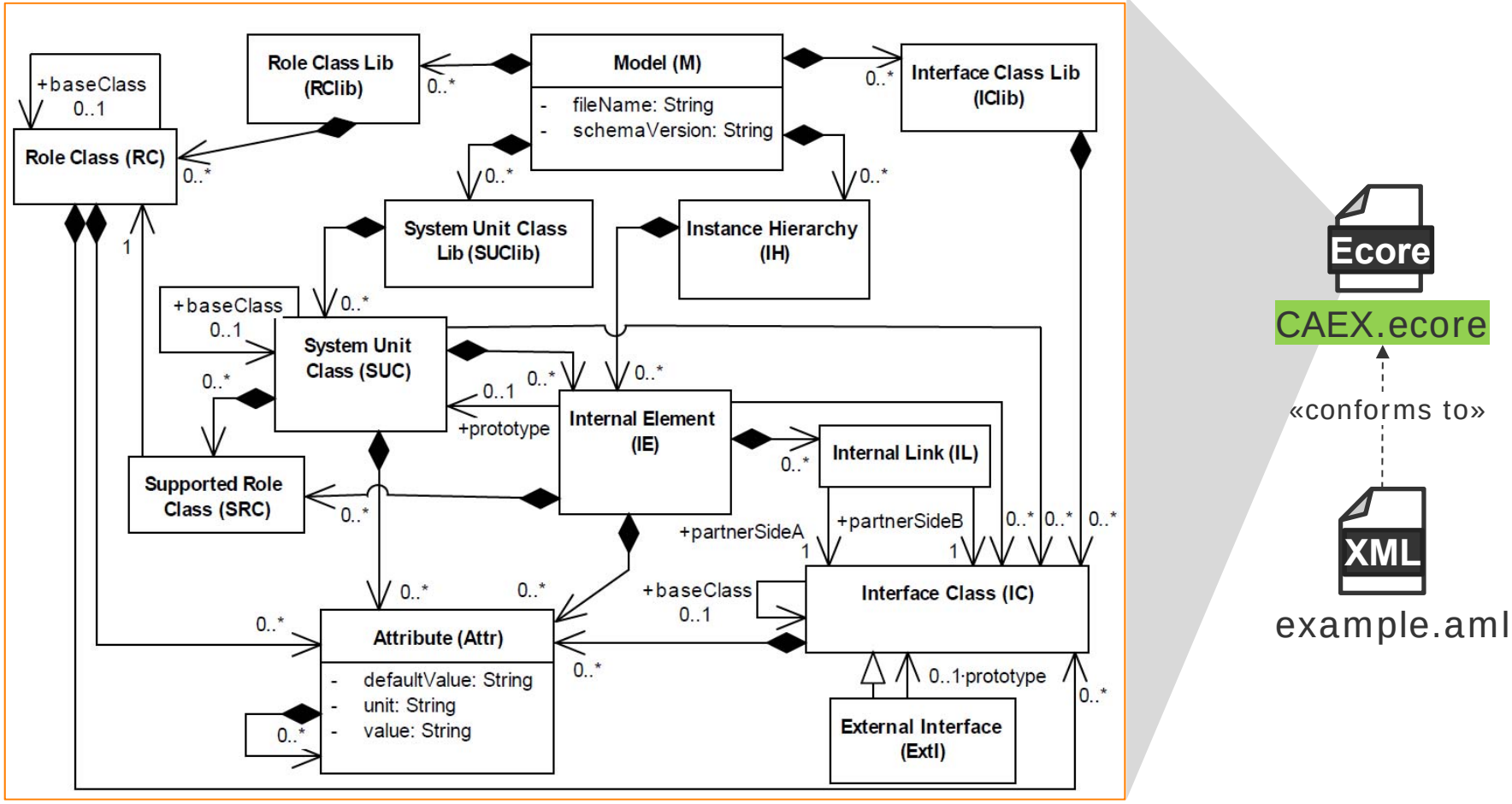
- **AutomationML** family is **defined by** a set of **XML Schemas**
- **Systematic metamodel creation process**
  - **Step 1**: **Generative** approach to produce initial Ecore-based metamodel
  - **Step 2**: **Refactorings** for improving language design
- Resulting metamodels
  - Are **complete**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



- 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.
- P. Neubauer, A. Bergmayr, T. Mayerhofer, J. Troya, M. Wimmer: ***XMLText: from XML schema to xText***. In Proc. of SLE 2015: 71-76

CAEX: Language Realization (3/3)

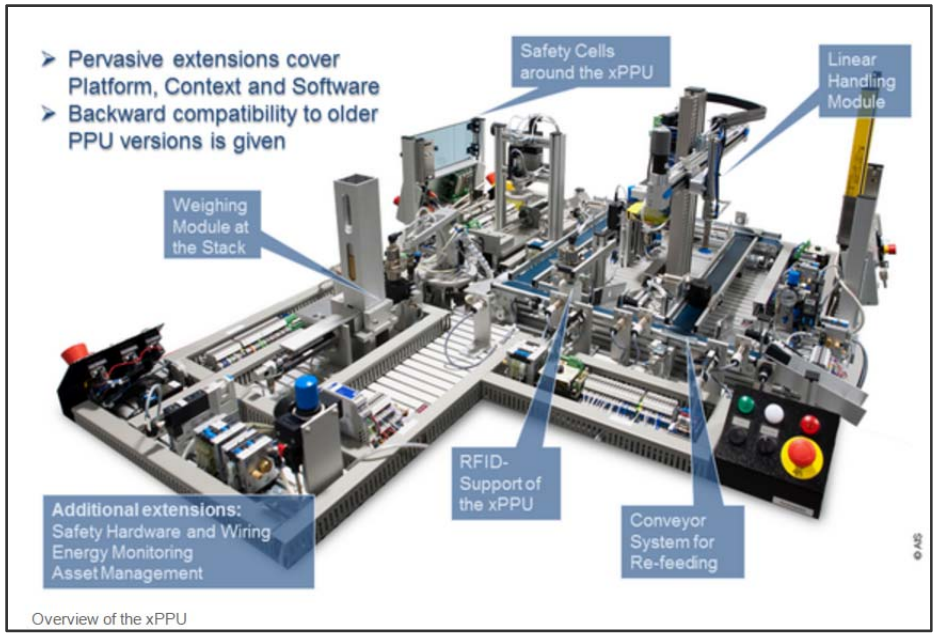
- **CAEX** is a language which is **now also specified by a metamodel**



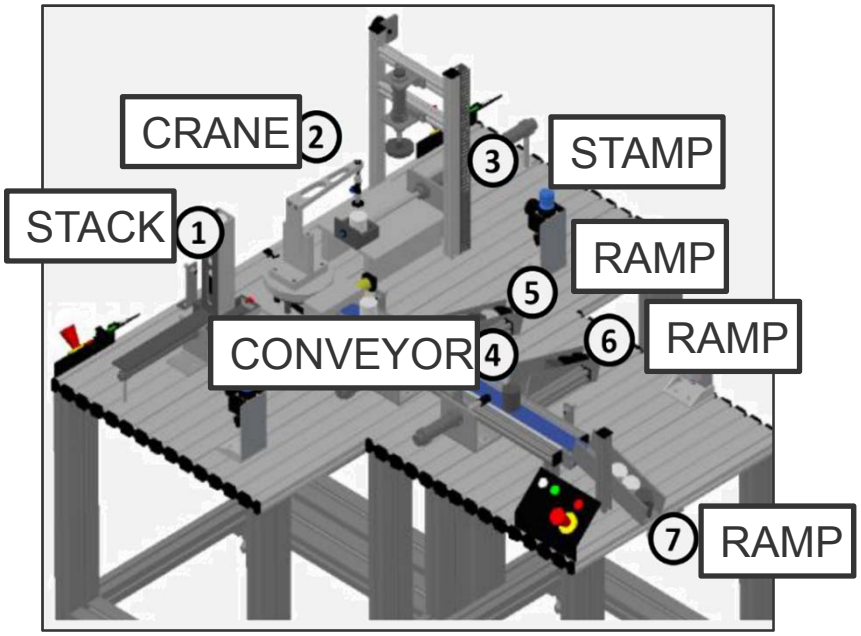
technology

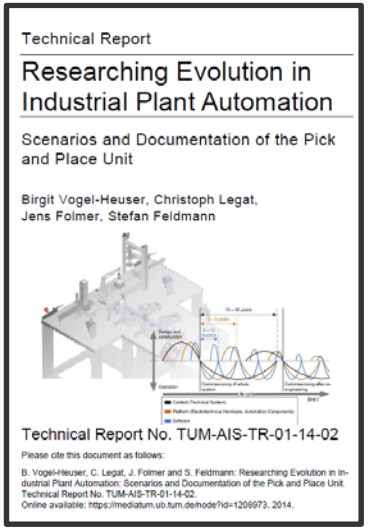
https://eclipse.org/modeling/emf/

Example: Pick And Place Unit (PPU)



Overview of the xPPU




Technical Report

Researching Evolution in Industrial Plant Automation

Scenarios and Documentation of the Pick and Place Unit

Birgit Vogel-Heuser, Christoph Legat, Jens Folmer, Stefan Feldmann

Technical Report No. TUM-AIS-TR-01-14-02

Please cite this document as follows:

B. Vogel-Heuser, C. Legat, J. Folmer and S. Feldmann: Researching Evolution in Industrial Plant Automation: Scenarios and Documentation of the Pick and Place Unit. Technical Report No. TUM-AIS-TR-01-14-02. Online available: https://mediatum.ub.tum.de/node?id=1206973, 2014.

- PPU **produces** different types of **tanks**
- PPU **resources** are its **stations** of different types
- **TUM provides set of models** and descriptions for the PPU

The Pick and Place Unit
http://www.ppu-demonstrator.org
Institute for Automation and Information Systems (AIS)
TUM - TU München

Example: PPU in CAEX

IH Whole system as
Instance Hierachy

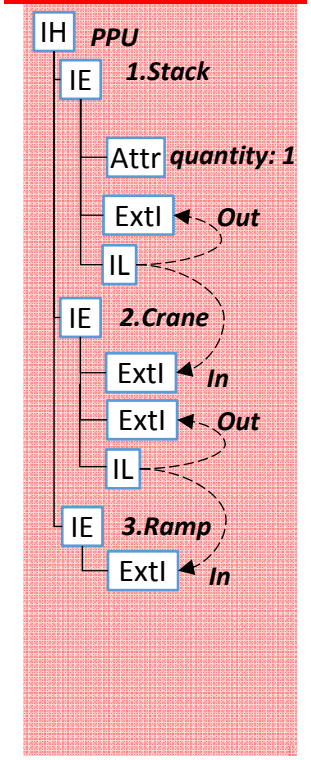
IE Its constituting **elements** as
Internal Element

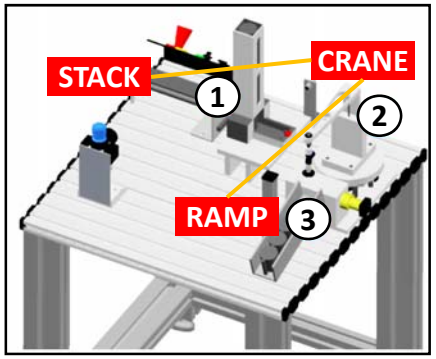
ExtI Elements' **connection points** as
External Interface

IL Elements' **connection** as
Internal Link

Attr Attributes as name=value pairs

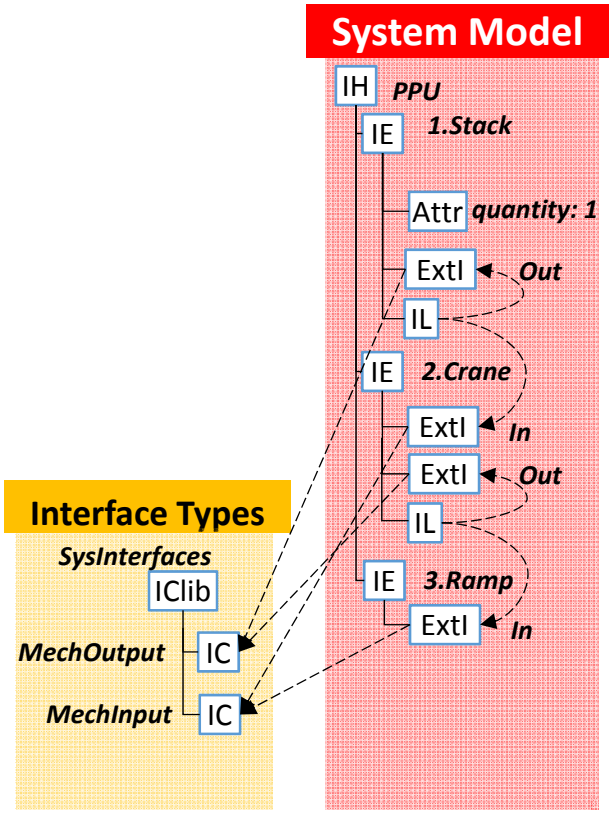
System Model





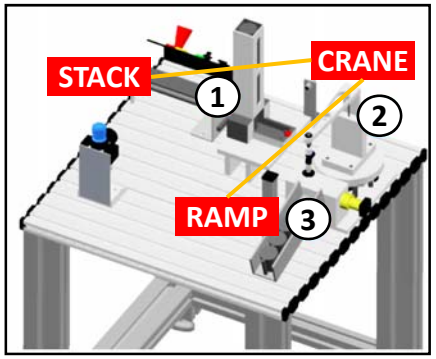
The Pick and Place Unit
http://www.ppu-demonstrator.org
Institute for Automation and Information Systems (AIS)
TUM - TU München

Example: PPU in CAEX



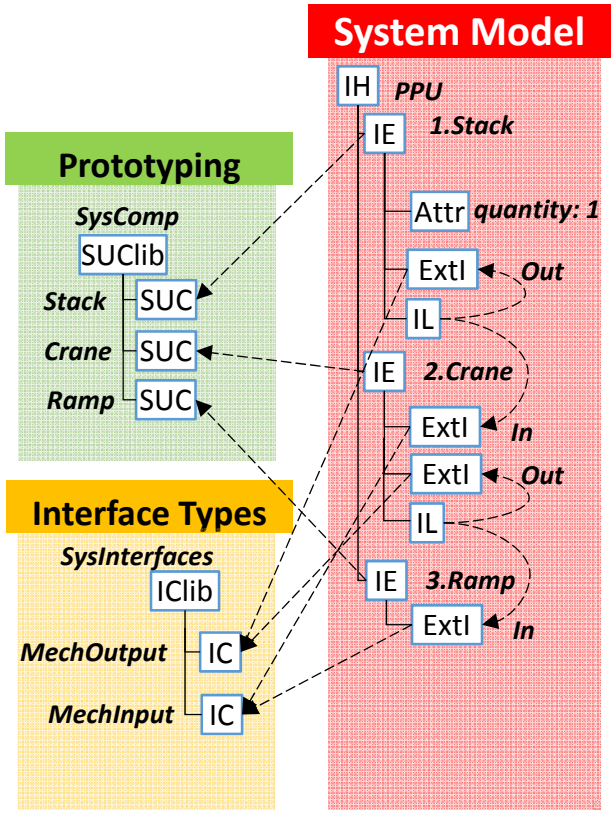
ICLib Set of Interface Classes

IC Interface Class



The Pick and Place Unit
http://www.ppu-demonstrator.org
Institute for Automation and Information Systems (AIS)
TUM - TU München

Example: PPU in CAEX

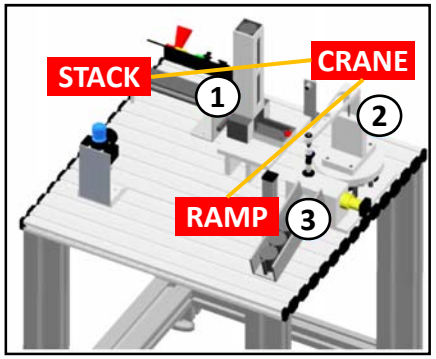


SUC Lib Set of reusable modeling elements

System Unit Class Library

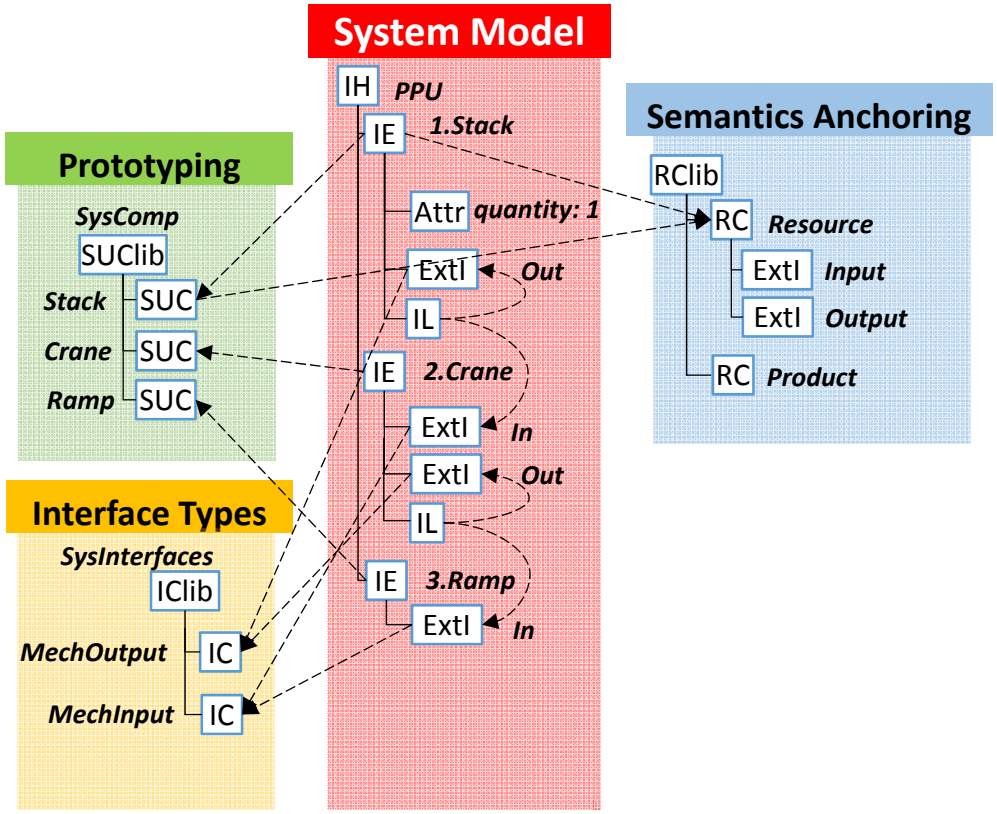
SUC Reusable Modeling Elements as

System Unit Class



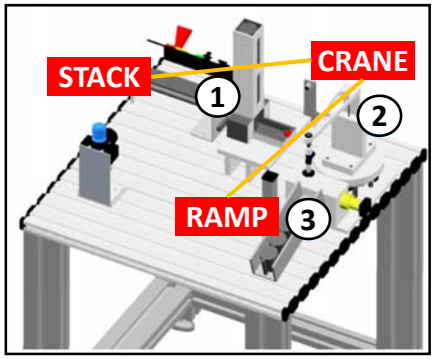
The Pick and Place Unit
http://www.ppu-demonstrator.org
Institute for Automation and Information Systems (AIS)
TUM - TU München

Example: PPU in CAEX



RCLib Set of Role Classes

RC Role Class



The Pick and Place Unit
http://www.ppu-demonstrator.org
Institute for Automation and Information Systems (AIS)
TUM - TU München

AML: Standard Editor

AutomationML Editor 4.7.0.0

File Edit Tools Plugins Settings Help

Fileproperties | Consistency Check | All internal links

InstanceHierarchy

- PPU
  - Stack_instance_1 { **Class** Stack **Role** Resource }
    - Stack_instance_1-Interfaces
      - MechOutput_instance_1 { **Class** MechOutput }
  - Crane_instance_1 { **Class** Crane **Role** Resource }
    - Crane_instance_1-Interfaces
      - MechInput_instance_1 { **Class** MechInput }
      - MechOutput_instance_1 { **Class** MechOutput }
  - Ramp_instance_1 { **Class** Ramp **Role** Resource }
    - Ramp_instance_1-Interfaces
      - MechInput_instance_1 { **Class** MechInput }

SystemUnitClassLib

- SysComp
  - Stack { **Class** }
  - Crane { **Class** }
  - Ramp { **Class** }

RoleClassLib

- AutomationMLBaseRoleClassLib
  - AutomationMLBaseRole { **Class** }
    - Group { **Class** AutomationMLBaseRole }
    - Facet { **Class** AutomationMLBaseRole }
    - Port { **Class** AutomationMLBaseRole }
    - Resource { **Class** AutomationMLBaseRole }
    - Product { **Class** AutomationMLBaseRole }

InterfaceClassLib

- SysInterfaces
  - MechInput { **Class** }
  - MechOutput { **Class** }
- AutomationMLInterfaceClassLib
  - AutomationMLBaseInterface { **Class** }
    - Order { **Class** AutomationMLBaseInterface }
    - PortConnector { **Class** AutomationMLBaseInterface }

Attributes : Stack_instance_1

Attributes

Attribute1

Attribute Detail: Attribute1

Description	quantity
Unit	
AttributeDataType	xs:int
DefaultValue	
Value	1

RefSemantic [0]

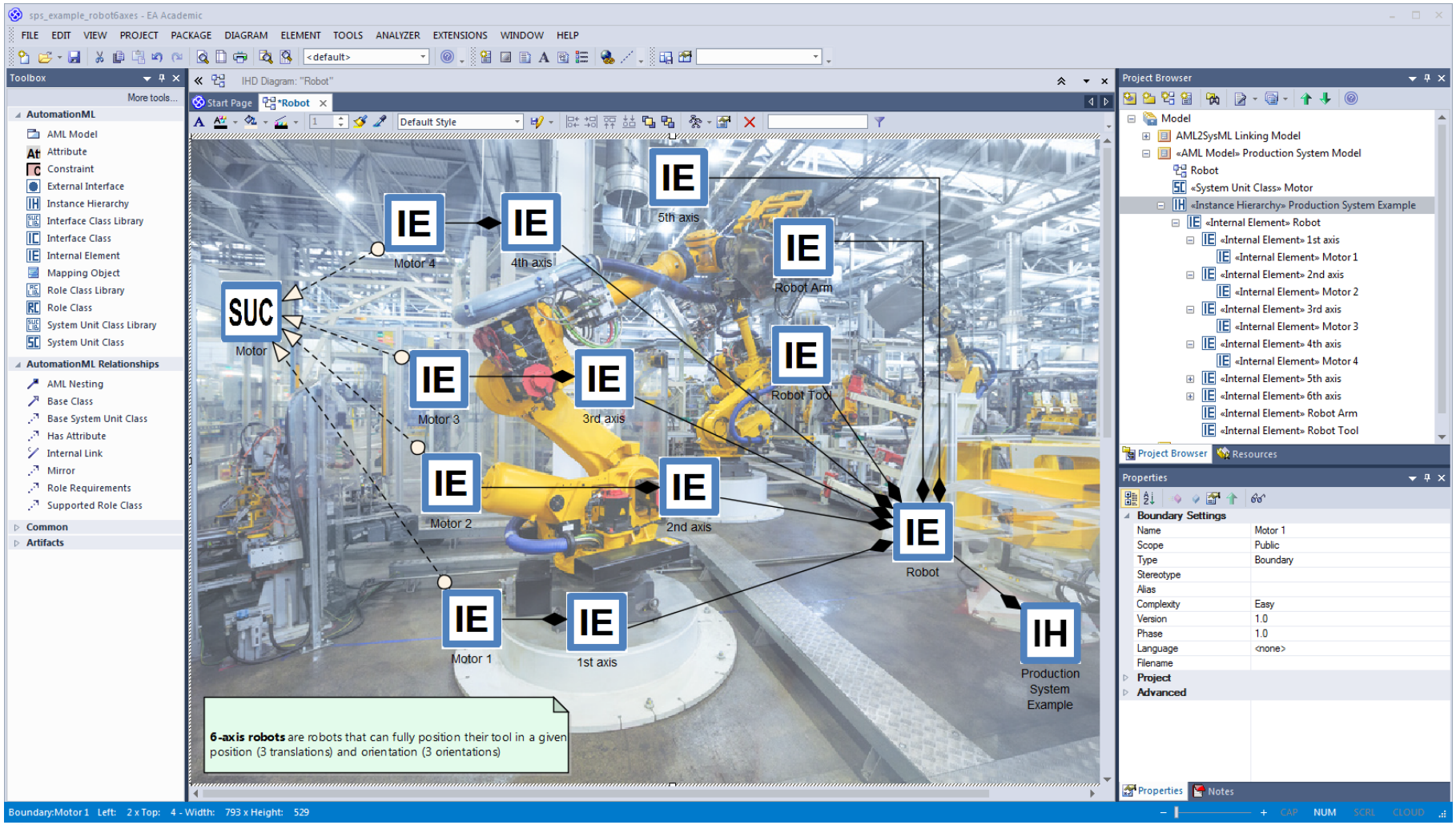
Constraint [0]

Header | Attributes | Internal Links

Zoom D:\presentations\MULTI_2018\demo.aml

Available at: www.automationml.org

AML: SysML-based Realization in EA



Available at: www.sysml4industry.org

AML: Research conducted in the last 4 years…

- Modeling Support
  - AML Metamodel
  - AML UML Profile
  - AML SysML Profile
  - AML Diagram
  - AML Text
- Evolution Support
  - AML Language Evolution
  - AML Library Evolution
- Constraints and Query Support
  - Automation Query Language
  - Cardinalities
  - Well-formdness rules

For references please check:
https://big.tuwien.ac.at/publications/?pub_searchType=full&pub_search=AutomationML&pub_year=all&pub_commit=Search

<AutomationML/>

IML

IEC 62424

Plant Behavior Models

PLCopen XML

- Standard XML format for exchanging PLC code
- Supports all programming languages defined in IEC 61131-3 standard
  - Structured text, instruction list, ladder diagram, function block diagram, sequential function charts

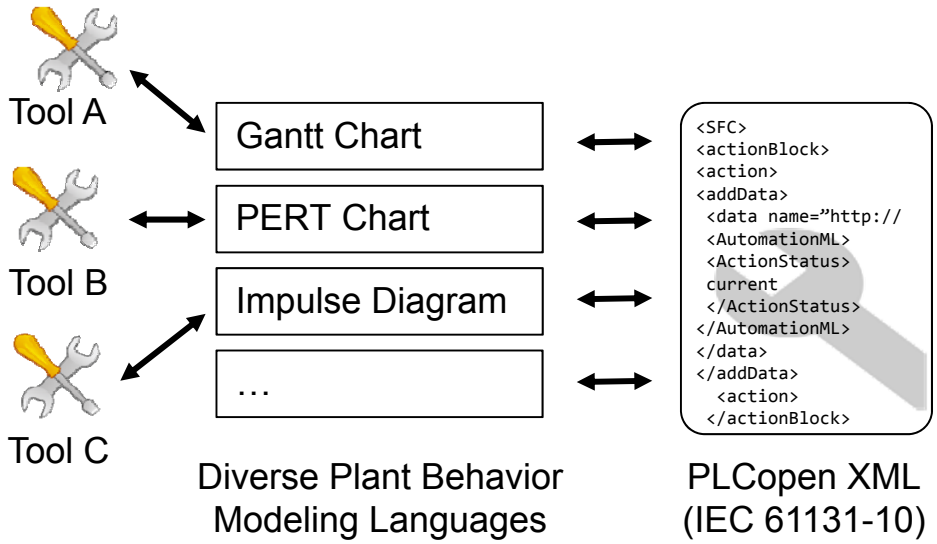
Common Plant Behavior Modeling Languages

- AutomationML supports five modeling languages for defining plant behaviors on different levels of abstraction
  - **Abstract process planning**: Gantt charts, PERT charts
  - **System behavior design**: Impulse diagrams
  - **Detailed component behavior design**: Sequential function charts, state charts

> Requires translations from/to PLCopen XML

Plant Behavior Models

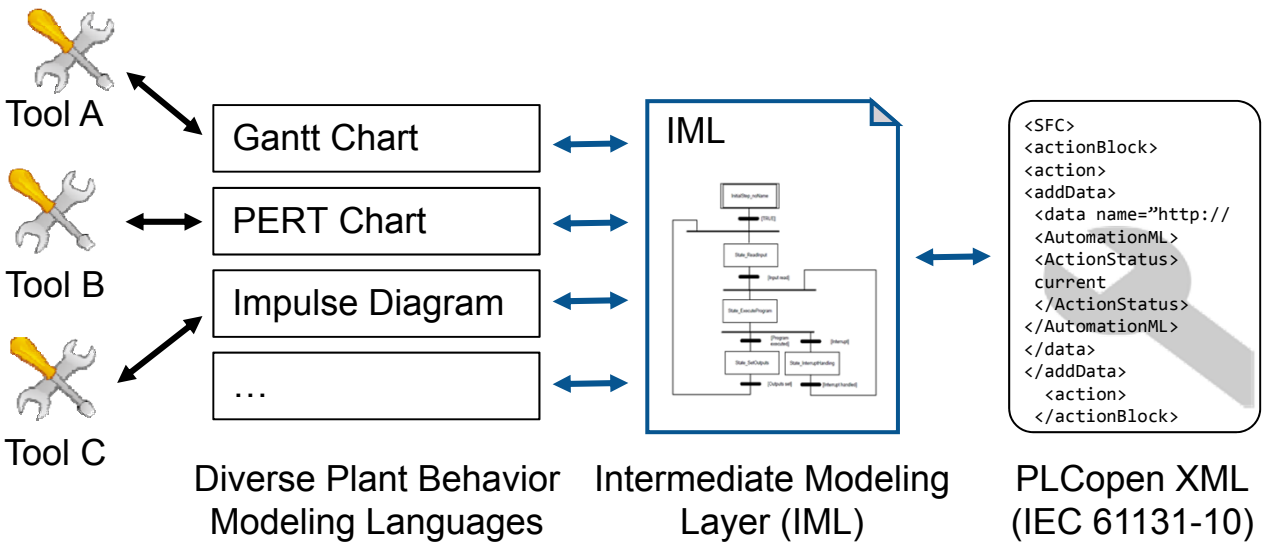
Translations from/to PLCopen XML



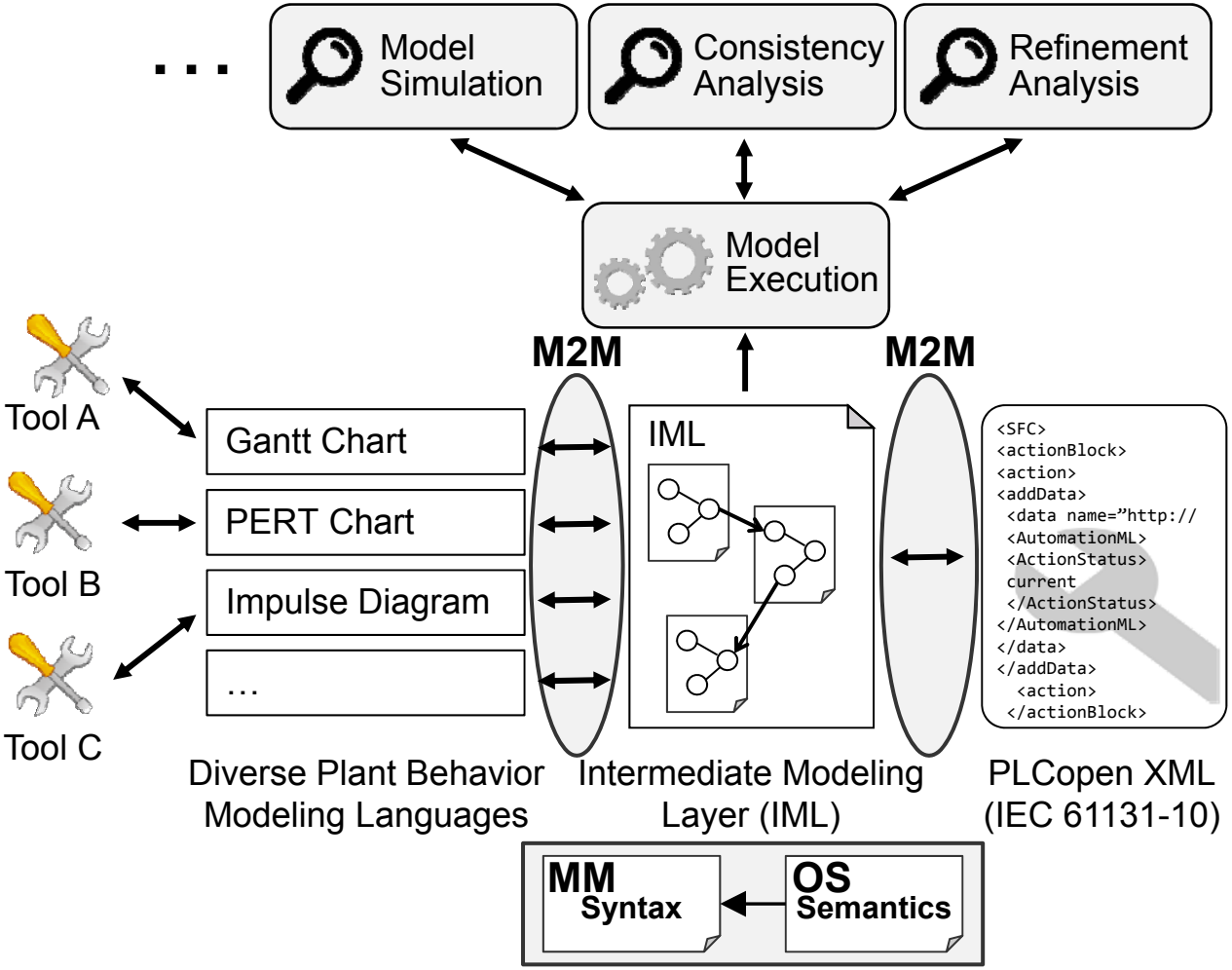
Intermediate Modeling Layer

AutomationML introduces Intermediate Modeling Layer (IML)

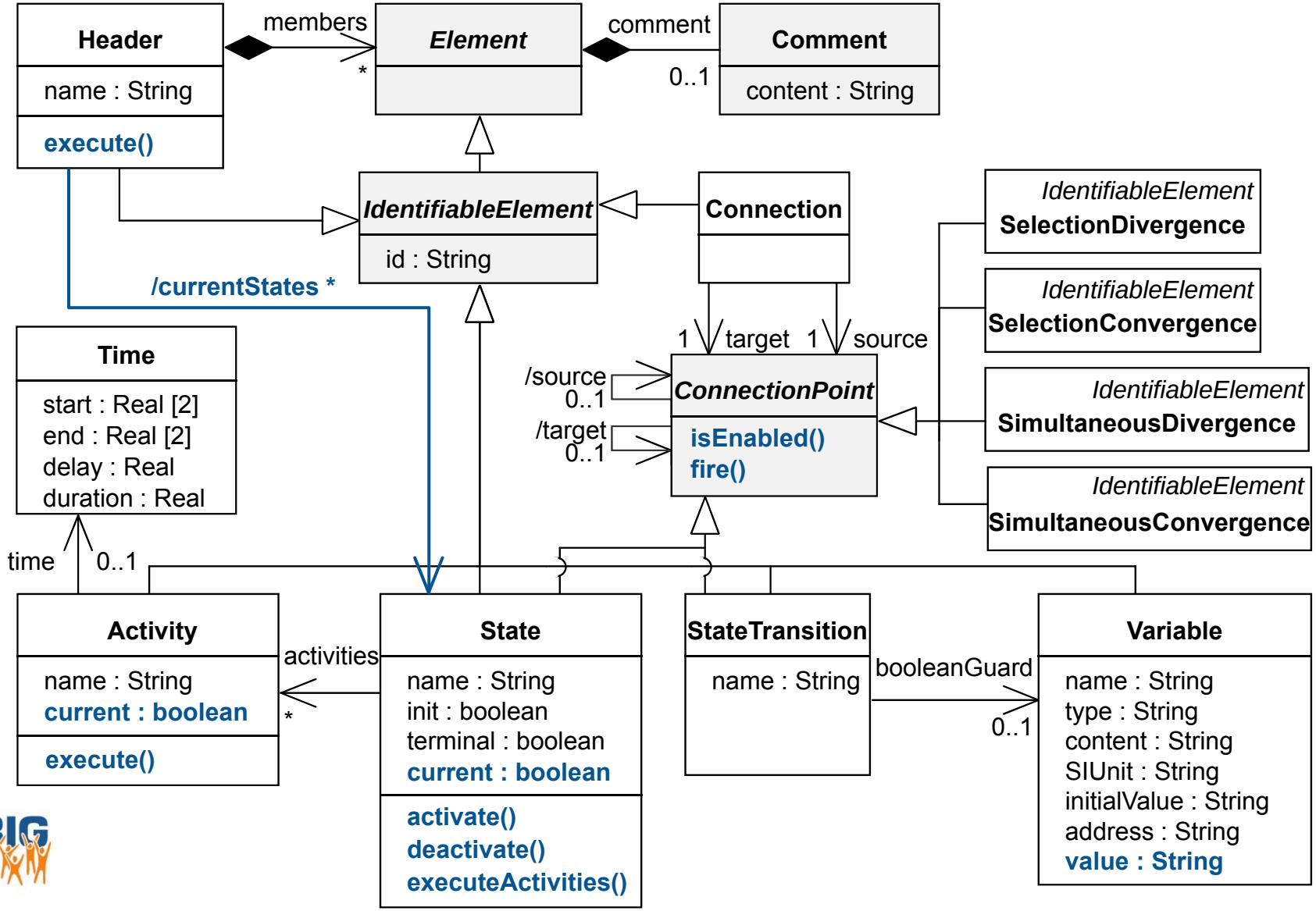
- IML is based on **sequential function charts** defined by IEC 61131-3 standard
- **Simplifies transformations** from/to PLCopen XML
- **Decouples PLCopen XML** from plant behavior modeling languages
  - Facilitates extensibility with new plant behavior modeling languages
  - Enables automated transformations between plant behavior modeling languages



IML as Semantic Domain and Integration Layer for Plant Behavior Models

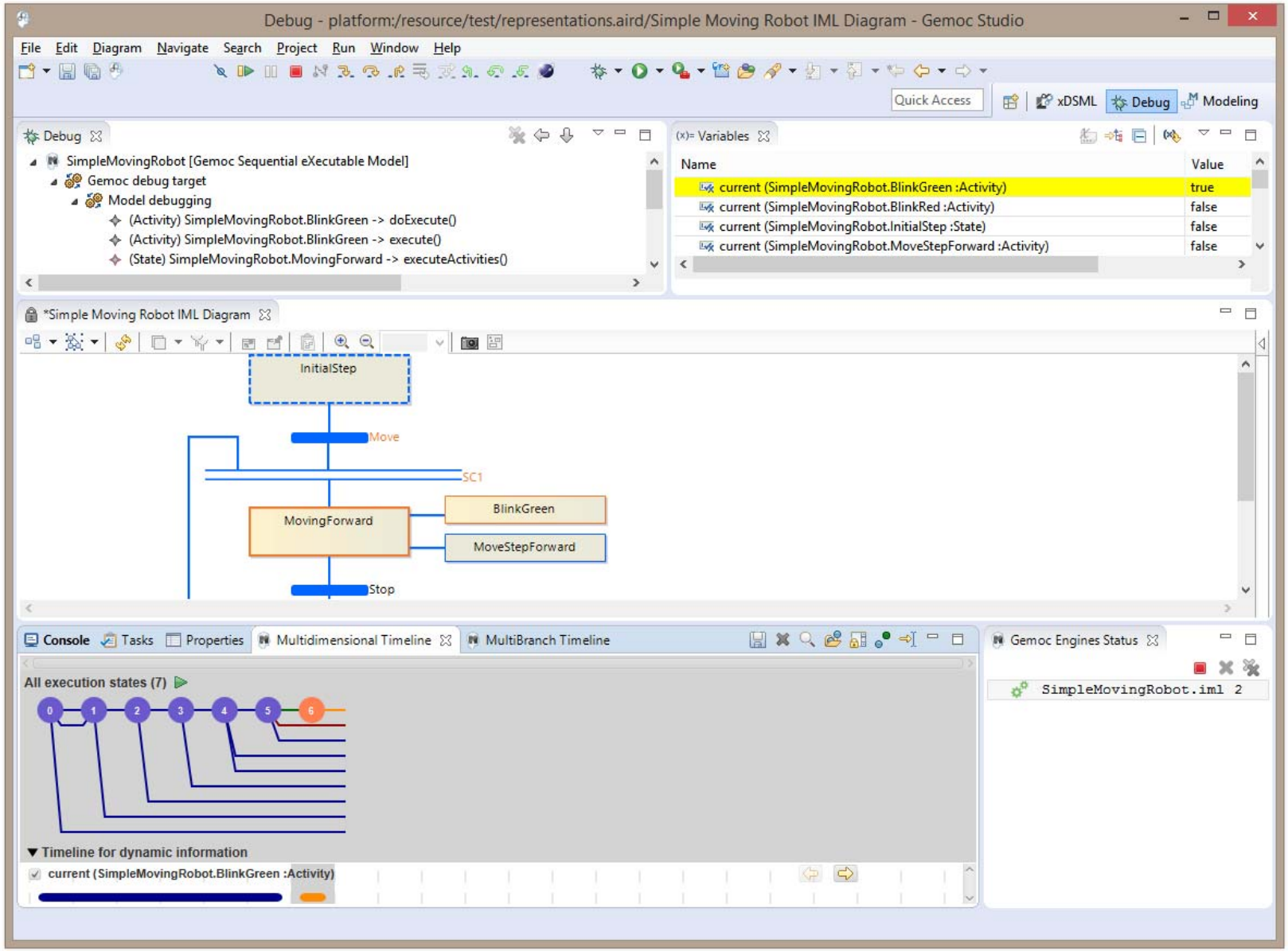


Operational Semantics for IML



Tool Support for IML based on

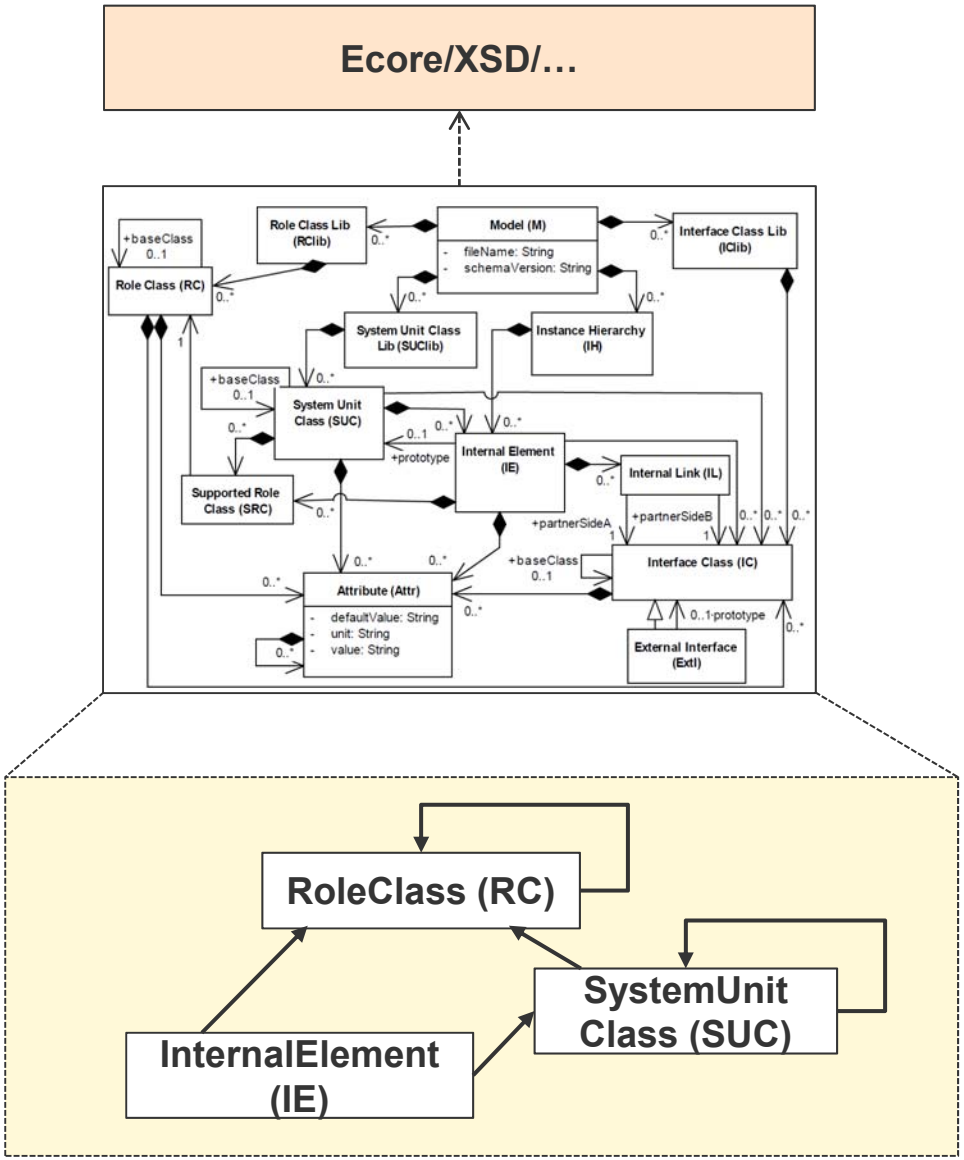
https://github.com/moliz/moliz.gemoc/tree/master/examples/iml



<AutomationML/>

A Multi-Level Perspective

AutomationML Realization: Important Concepts/Relationships



What is modelled with AutomationML?

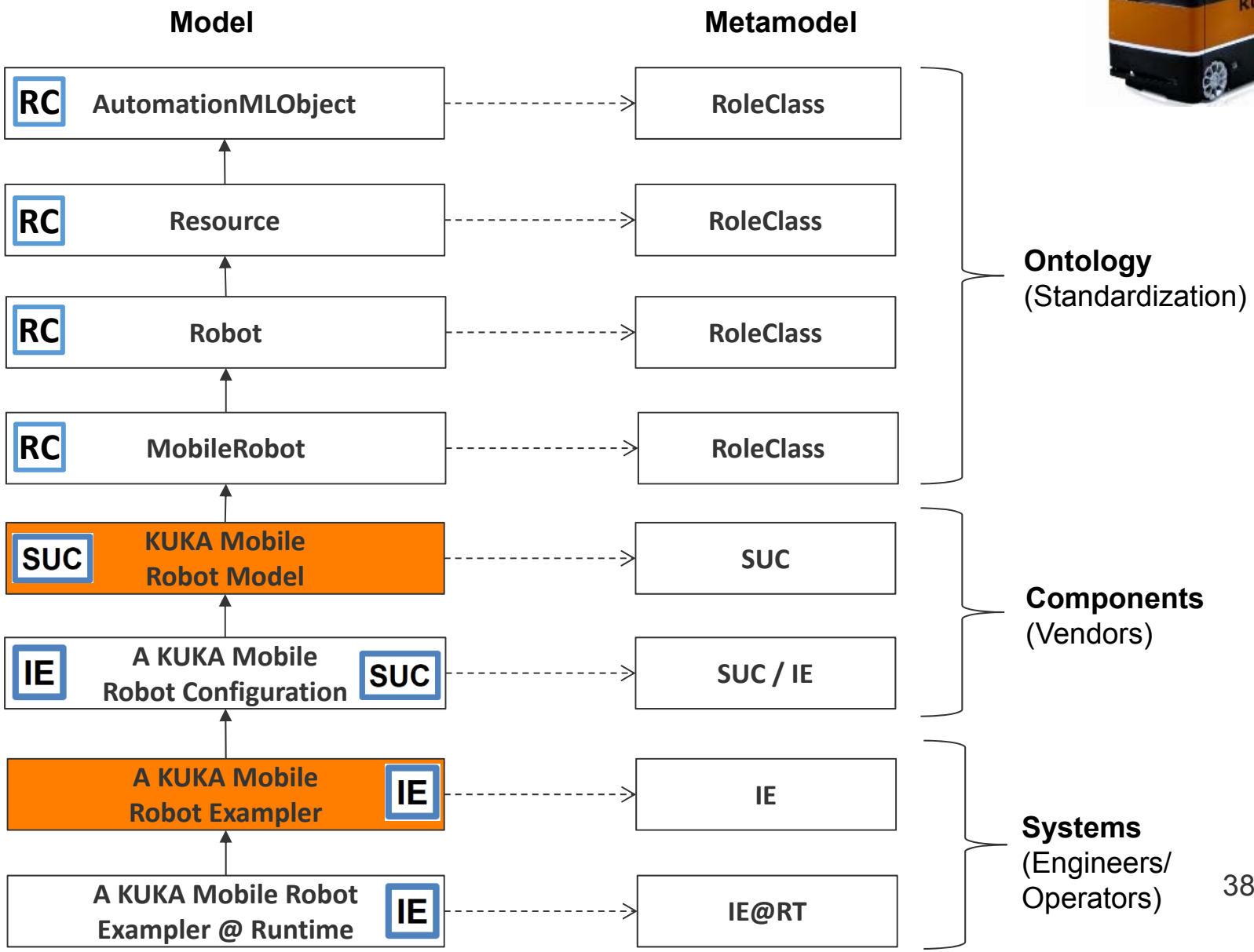
Model	Metamodel
RC AutomationMLObject	RoleClass
RC Resource	RoleClass
RC Robot	RoleClass
RC MobileRobot	RoleClass
SUC KUKA Mobile Robot Model	SystemUnitClass
IE A KUKA Mobile Robot Configuration SUC	InternalElement/ SystemUnitClass
A KUKA Mobile Robot Exampler IE	InternalElement
A KUKA Mobile Robot Exampler @ RunTime IE	InternalElement

Ontology (Standardization)

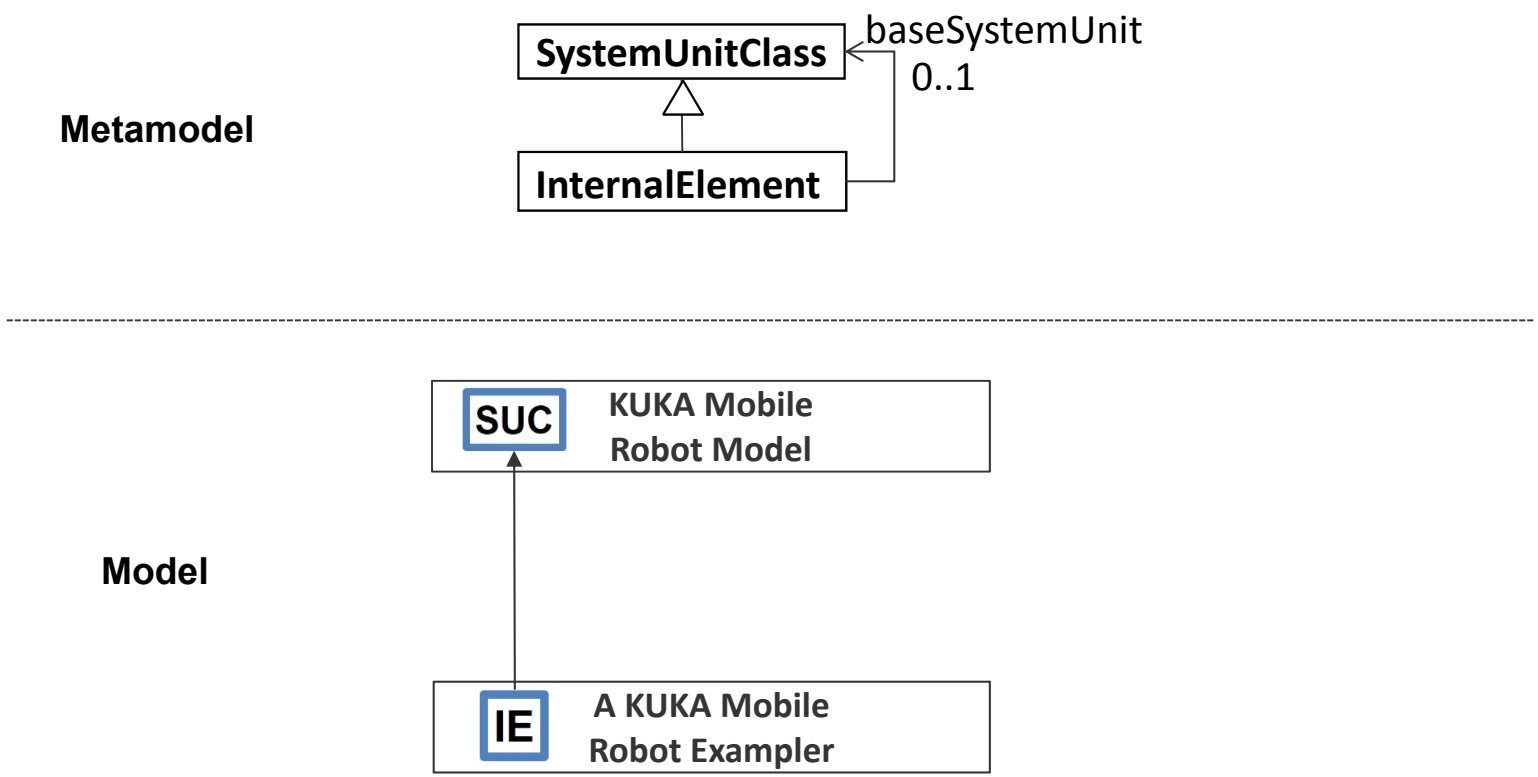
Components (Vendors)

Systems (Engineers/ Operators)

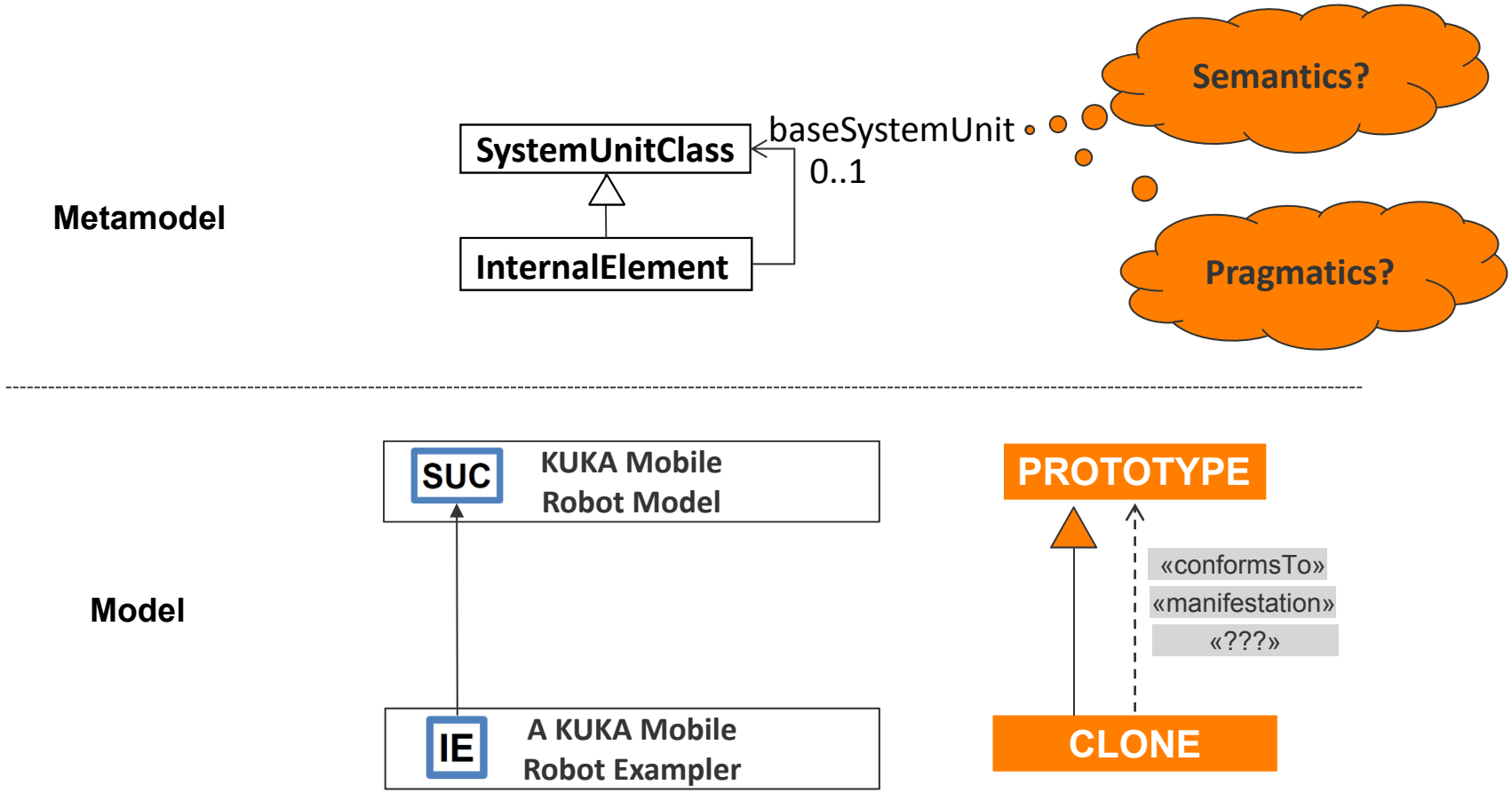
Most Basic Pattern: SUC ← IE



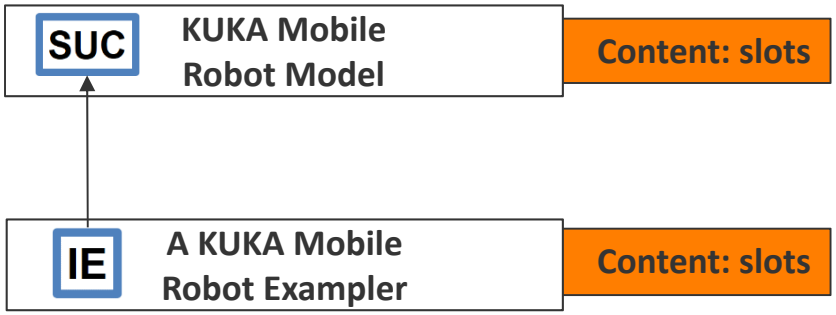
Challenge 1: Relationship between IEs and SUCs



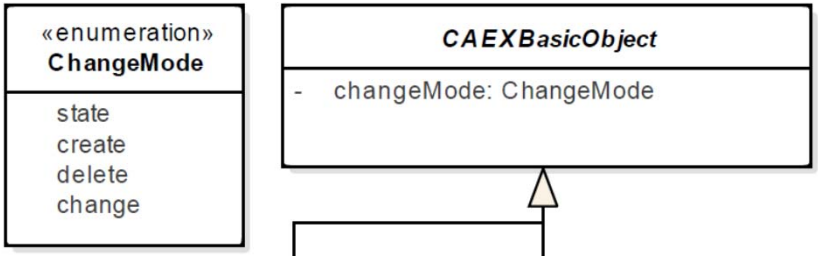
Challenge 1: Relationship between IEs and SUCs



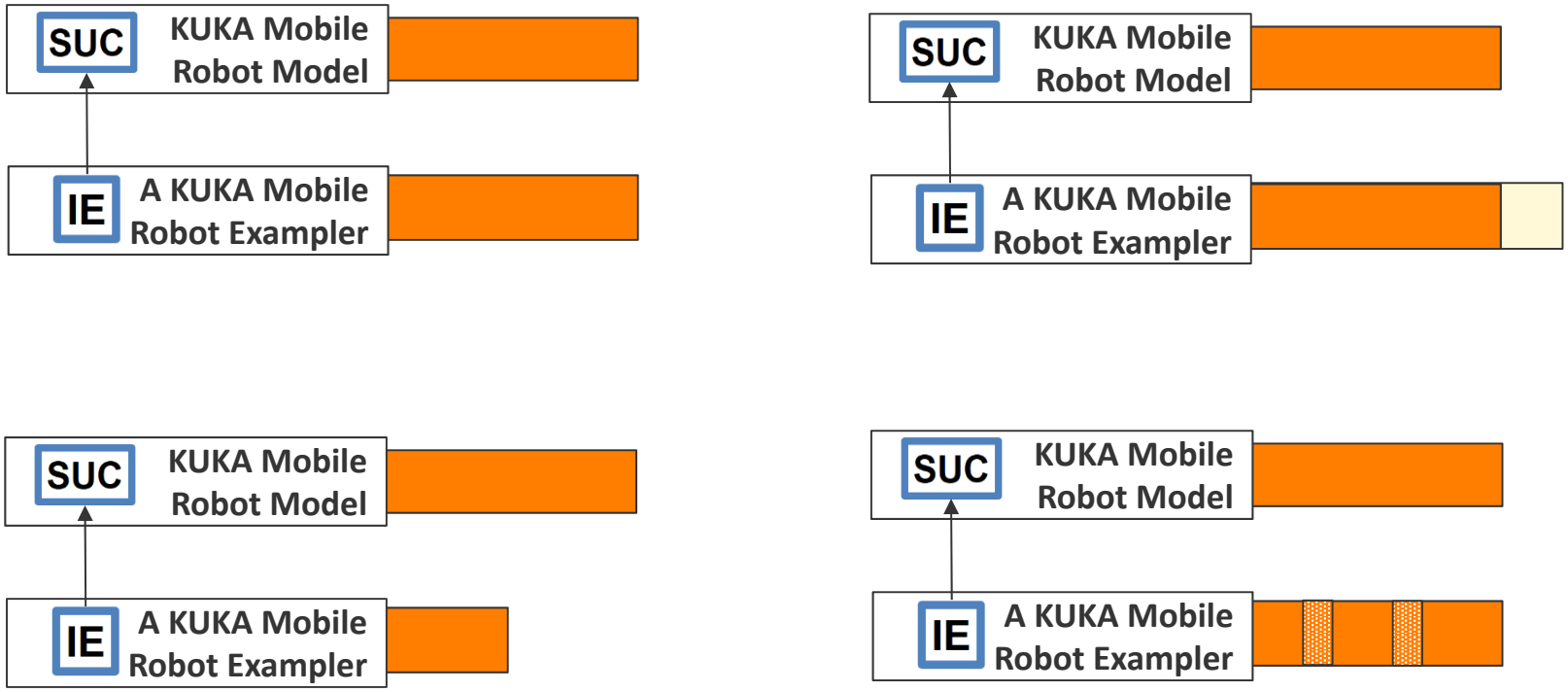
Challenge 1: Relationship between IEs and SUCs




```
context SUC::createClone()
post: new c:IE |
-- clone refers to its prototype
c.prototype = self and
-- clone contains all prototype slots
self.slots -> forAll(pS | c.slots -> one(cS | pS.name = cS.name
and pS.value = cS.value)) and
-- clone only contains prototype slots
c.slots -> forAll(cS | self.slots -> one(pS | cS.name = pS.name
and cS.value = pS.value))
```

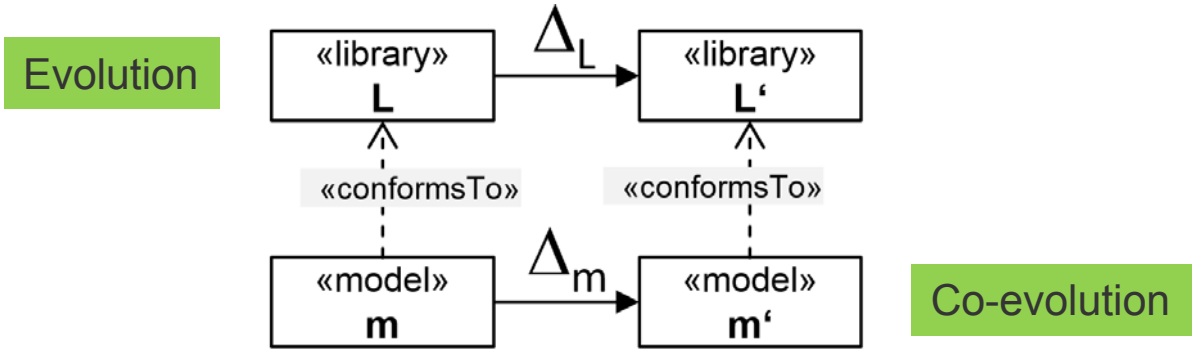


Challenge 1: Relationship between IEs and SUCs



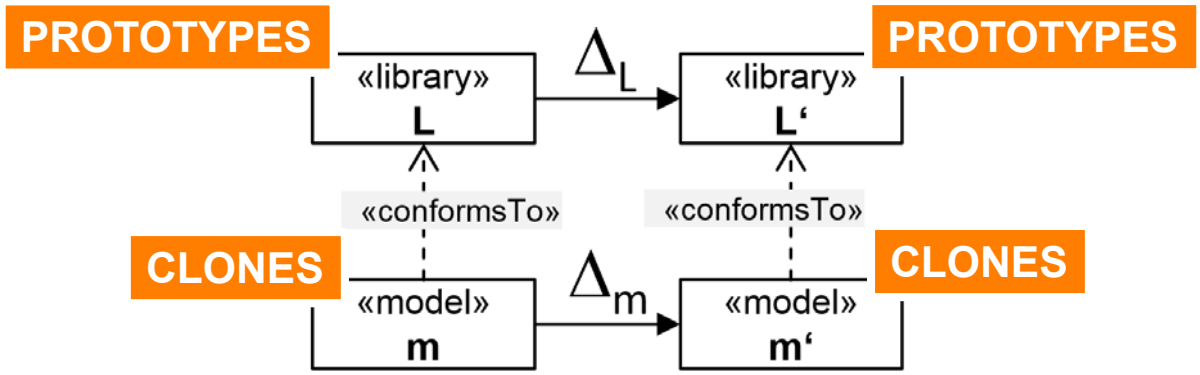
Challenge 2: Co-Evolution

- **Evolution** of **engineering data** has to be managed
  - Engineers from diverse domains working in parallel
- **Co-evolution** of **prototypes** and **clones** has to be managed

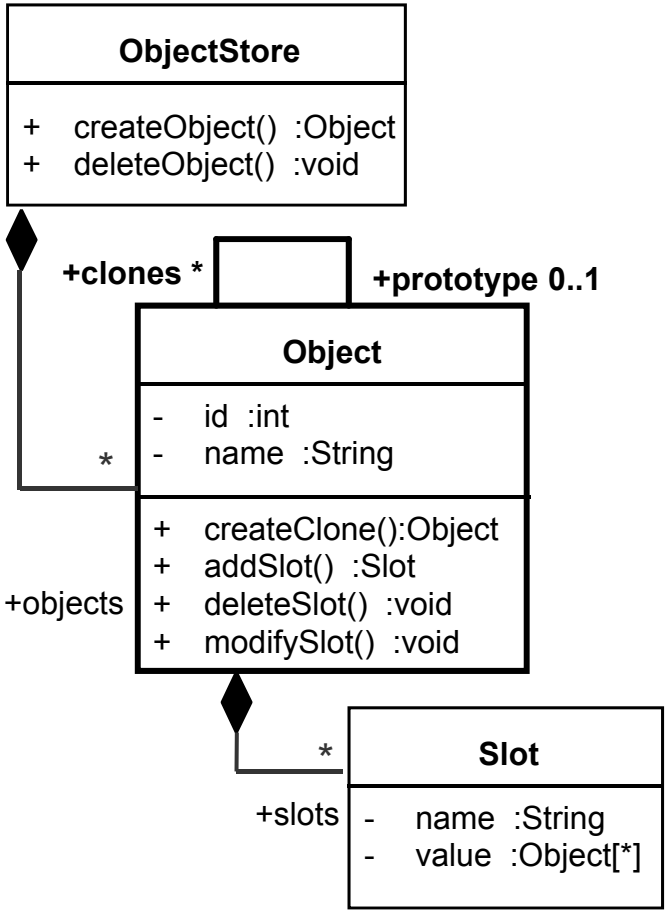


Framework for prototype/clone co-evolution

1. **Generic metamodel** for prototype-based languages
2. **Levels of consistency rigor** between PROTOTYPES and CLONES
3. **Change types** on PROTOTYPES and **their impact** on prototype/clone consistency
4. **Repair operations** to re-establish prototype/clone consistency

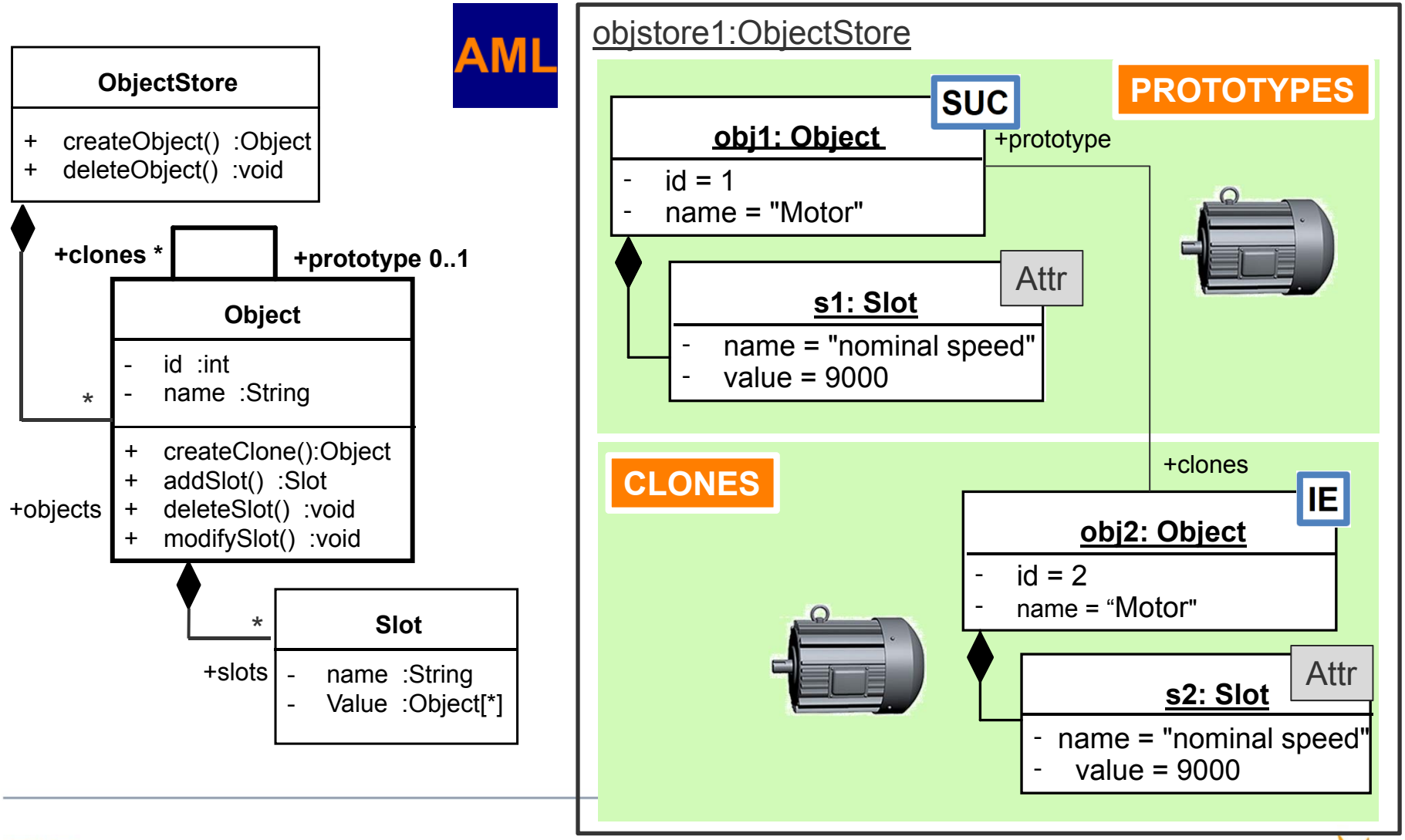


Generic Metamodel for Prototype-Based Languages



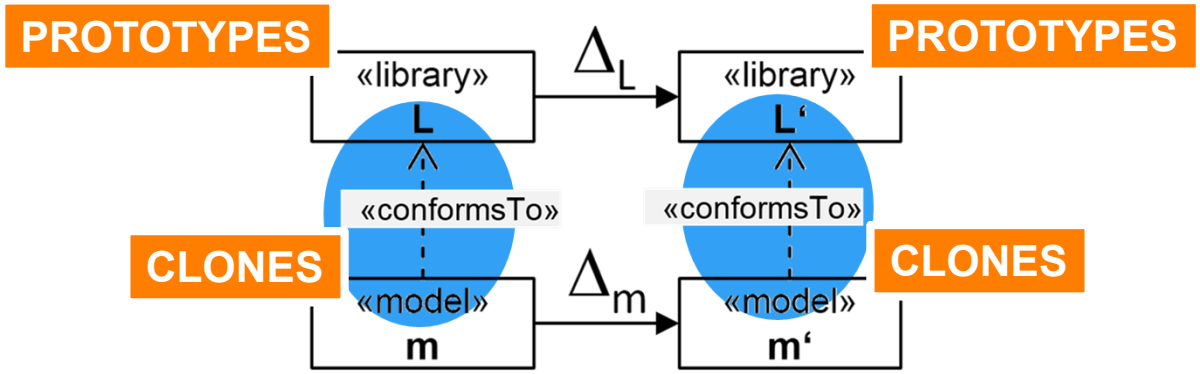
- A **general metamodel for prototypes and clones whose concepts can be adopted** for several modeling languages.
- PROTOTYPES and CLONES as **roles**
- A **Object can play different roles**, depending on its current relationships

Generic Metamodel for Prototype-Based Languages

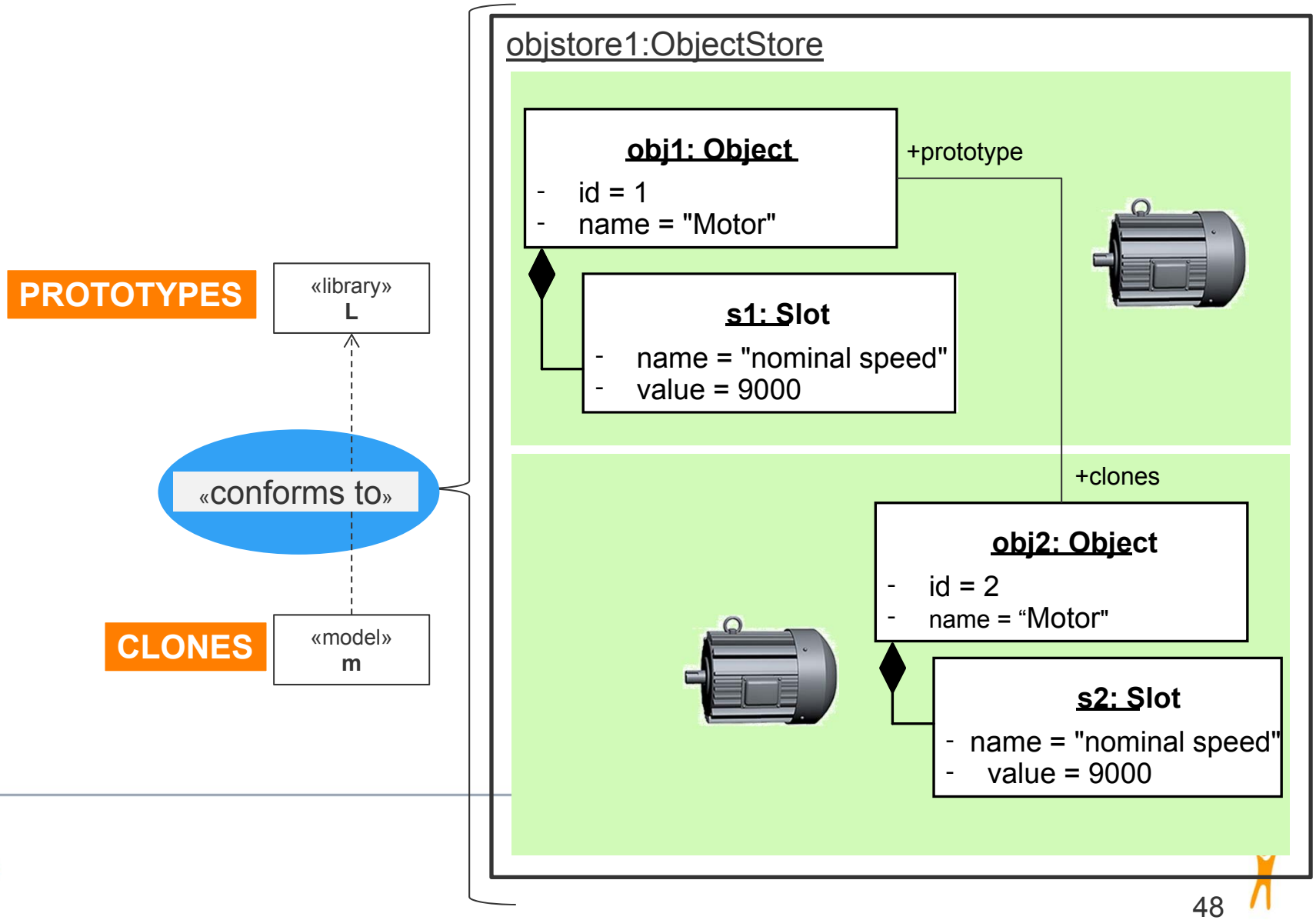


Framework for prototype/clone co-evolution

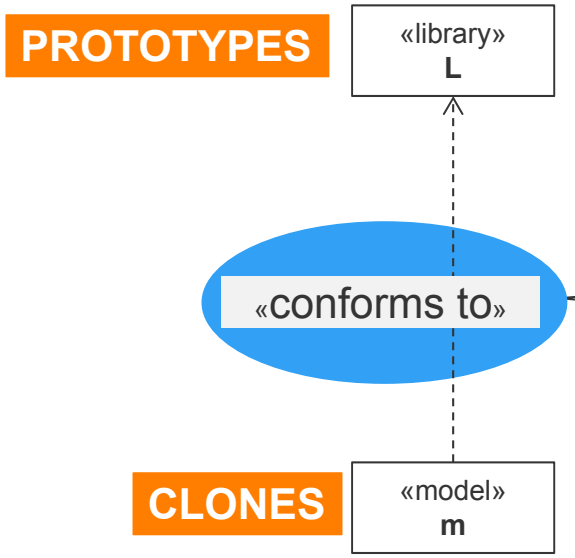
1. **Generic metamodel** for prototype-based languages
2. **Levels of consistency rigor** between PROTOTYPES and CLONES
3. **Change types** on PROTOTYPES and **their impact** on prototype/clone consistency
4. **Repair operations** to re-establish prototype/clone consistency



Levels of Consistency Rigor between Prototypes and Clones



Levels of Consistency Rigor between Prototypes and Clones



Level 0: Uncontrolled Compliance

- **Clones** may evolve completely independent from **Prototypes**
- **Prototypes** are solely used as templates

Level 1: Substantial Compliance

- **Evolution of clones** is partially restricted

Level 1a: Extension: all or more slots

Level 1b: Restriction: at most all the slots

Level 1c: Redefinition: some slots, different values

Level 2: Full Compliance

- Clones may not evolve independently of prototypes

Formalization of Consistency Levels: Consistency Constraints

Level 0: Uncontrolled Compliance

No consistency constraint required

Level 1a: Extension

```
-- clone defines all prototype slots but may define additional slots
context Object [self.prototype <> OclUndefined]
inv: self.prototype.slots -> forAll(pS | self.slots -> one(cS | pS.name = cS.name
  and pS.value = cS.value))
```

Level 1b: Restriction

```
-- clone defines subset of prototype slots
inv: self.slots -> forAll(cS | self.prototype.slots -> one(pS | cS.name = pS.name
  and cS.value = pS.value))
```

Level 1c: Redefinition

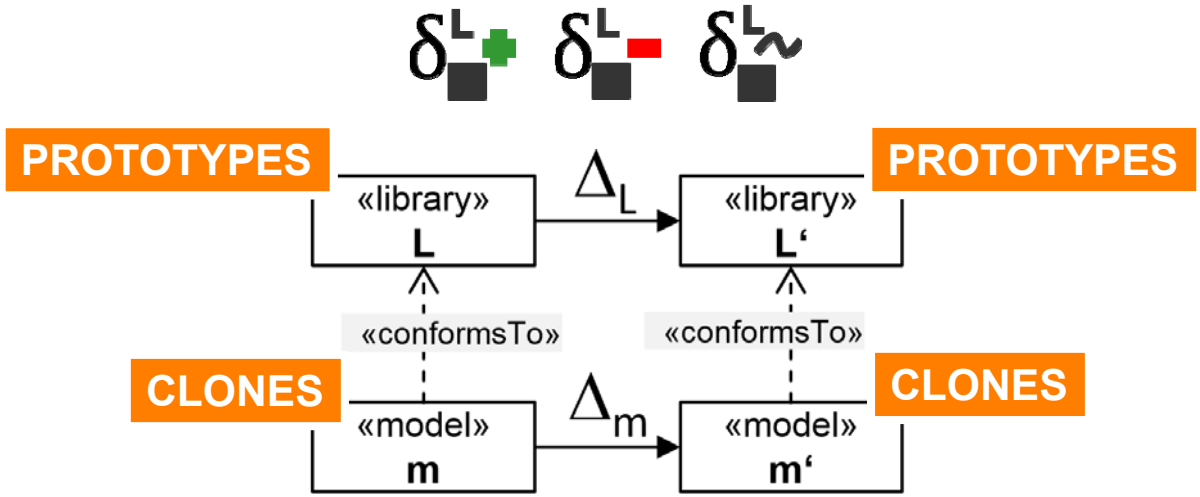
```
-- clone may redefine values of prototype slots
context Object [self.prototype <> OclUndefined]
inv: self.slots -> forAll(cS | self.prototype.slots -> one(pS | cS.name = pS.name))
inv: self.prototype.slots -> forAll(pS | self.slots -> one(cS | pS.name = cS.name))
```

Level 2: Full Compliance

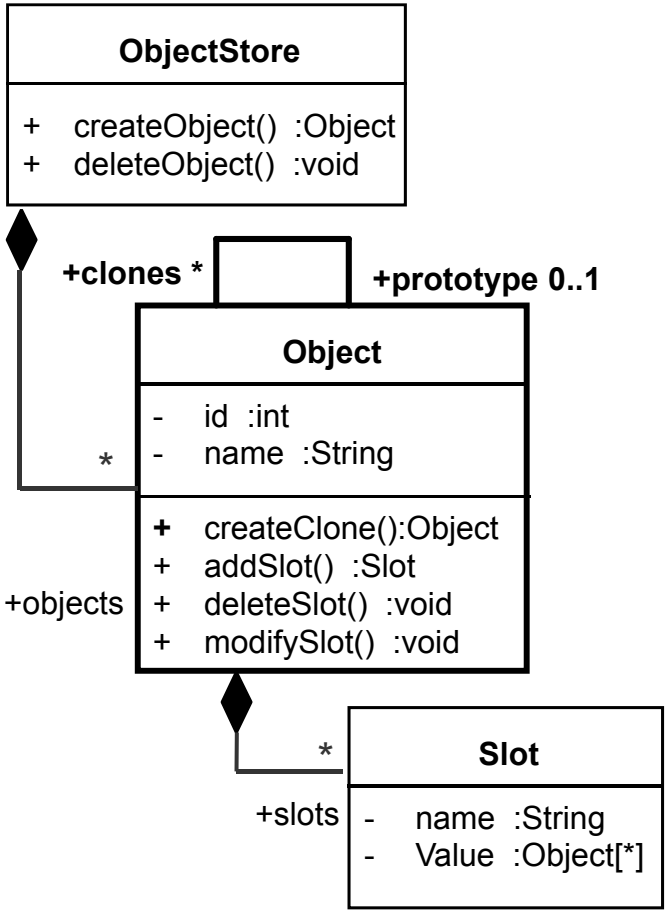
Post-condition of *createClone()* operation must always hold

Framework for prototype/clone co-evolution

1. Generic metamodel for prototype-based languages
2. Levels of consistency rigor between PROTOTYPES and CLONES
3. **Change types** on PROTOTYPES and **their impact** on prototype/clone consistency
4. Repair operations to re-establish prototype/clone consistency



Change Types on Prototypes and their Impact on Consistency



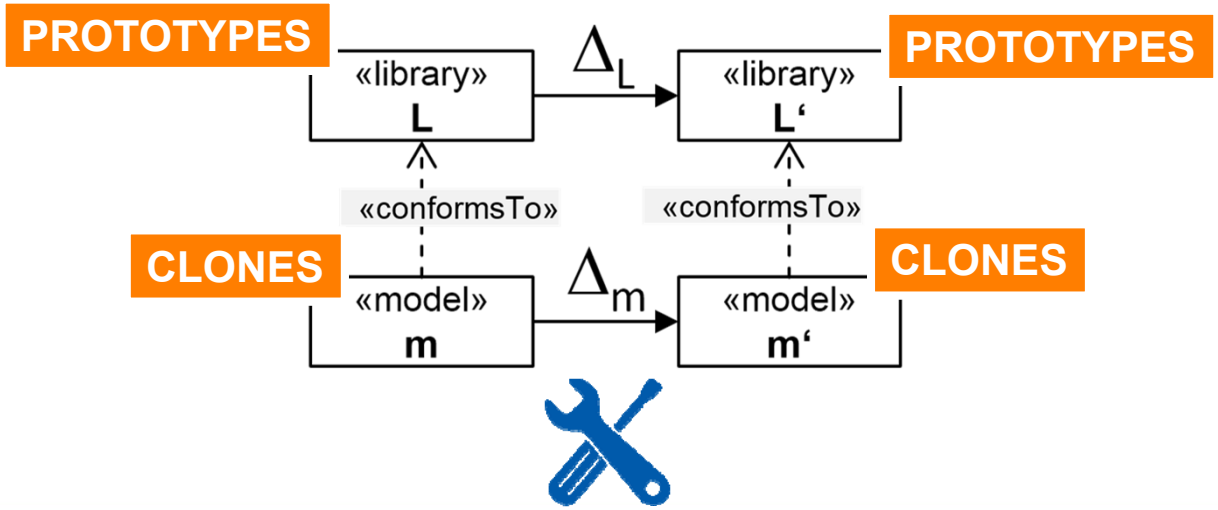
Operation	L0	L1a	L1b	L1c	L2
ObjectStore::createObject	↑	↑	↑	↑	↑
ObjectStore::deleteObject	≠	≠	≠	≠	≠
Object::addSlot	↑	≠	↑	≠	≠
Object::deleteSlot	↑	↑	≠	≠	≠
Object::modifySlot	↑	≠	≠	↑	≠

change on prototype — impact on consistency of clones

↑ non-breaking
≠ breaking

Framework for prototype/clone co-evolution

1. **Generic metamodel** for prototype-based languages
2. **Levels of consistency rigor** between PROTOTYPES and CLONES
3. **Change types** on PROTOTYPES and **their impact** on prototype/clone consistency
4. **Repair operations** to re-establish prototype/clone consistency



4. Repair Operations to re-establish Prototype/Clone Consistency

- Breaking changes lead to **inconsistencies** between prototypes and clones and **violations of consistency levels**
- Re-establishing prototype/clone consistency requires
  1. **Detection** of inconsistent clones through **consistency constraint**
  2. **Application of repair operations** on clones to resolve inconsistency

Operation	L0	L1a	L1b	L1c	L2
ObjectStore::createObject	↑	↑	↑	↑	↑
ObjectStore::deleteObject	≠	≠	≠	≠	≠
Object::addSlot	↑	≠	↑	≠	≠
Object::deleteSlot	↑	↑	≠	≠	≠
Object::modifySlot	↑	≠	≠	↑	≠

if preconditions hold

> Add slot to clones

> Remove slot from clones

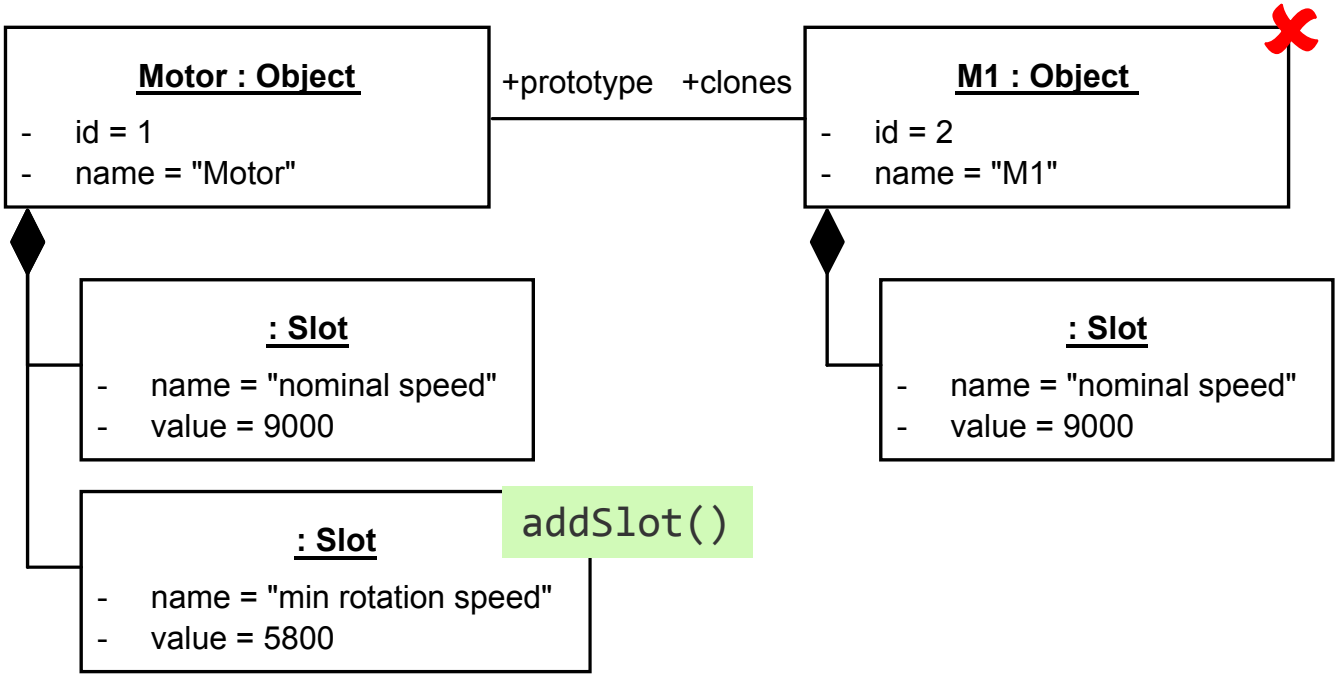
> Update slot value in clones

≠ manual resolution needed ≠ automated resolution possible

4. Repair Operations to Re-Establish Prototype/Clone Consistency

Example

Desired consistency level: L1a Extension

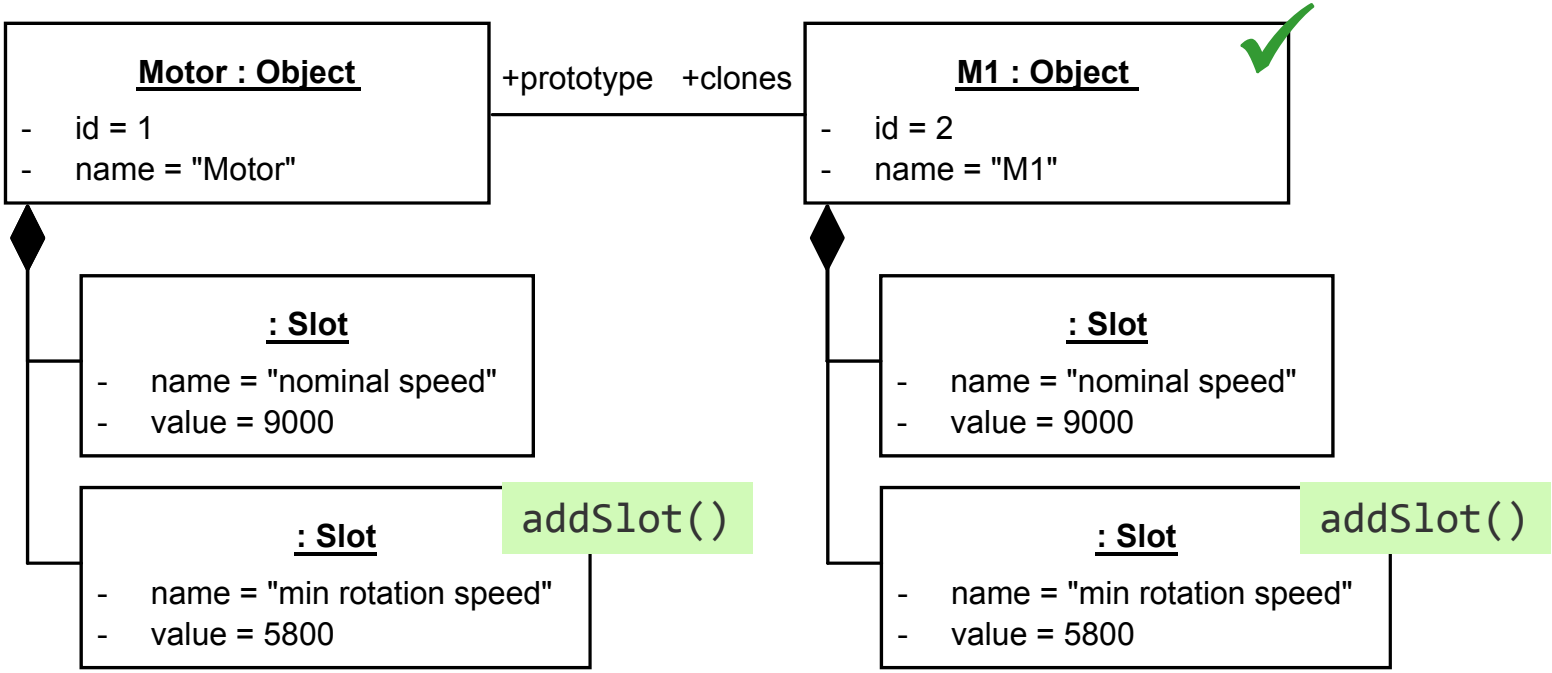



```
-- clone defines all prototype slots but may define additional slots
context Object [self.prototype <> OclUndefined]
inv: self.prototype.slots -> forAll(pS | self.slots -> one(cS | pS.name = cS.name
  and pS.value = cS.value))
```

4. Repair Operations to Re-Establish Prototype/Clone Consistency

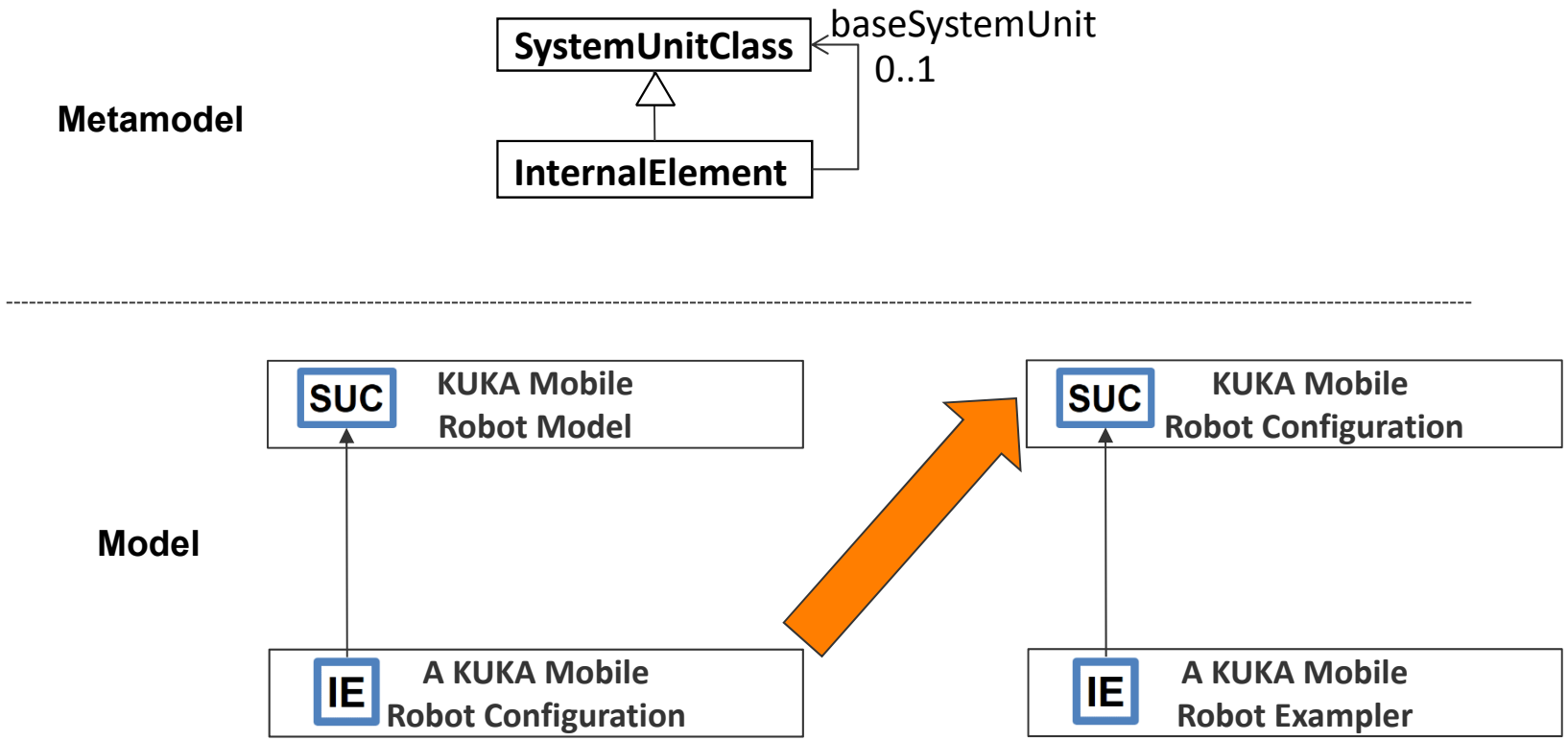
Example

Desired consistency level: L1a Extension

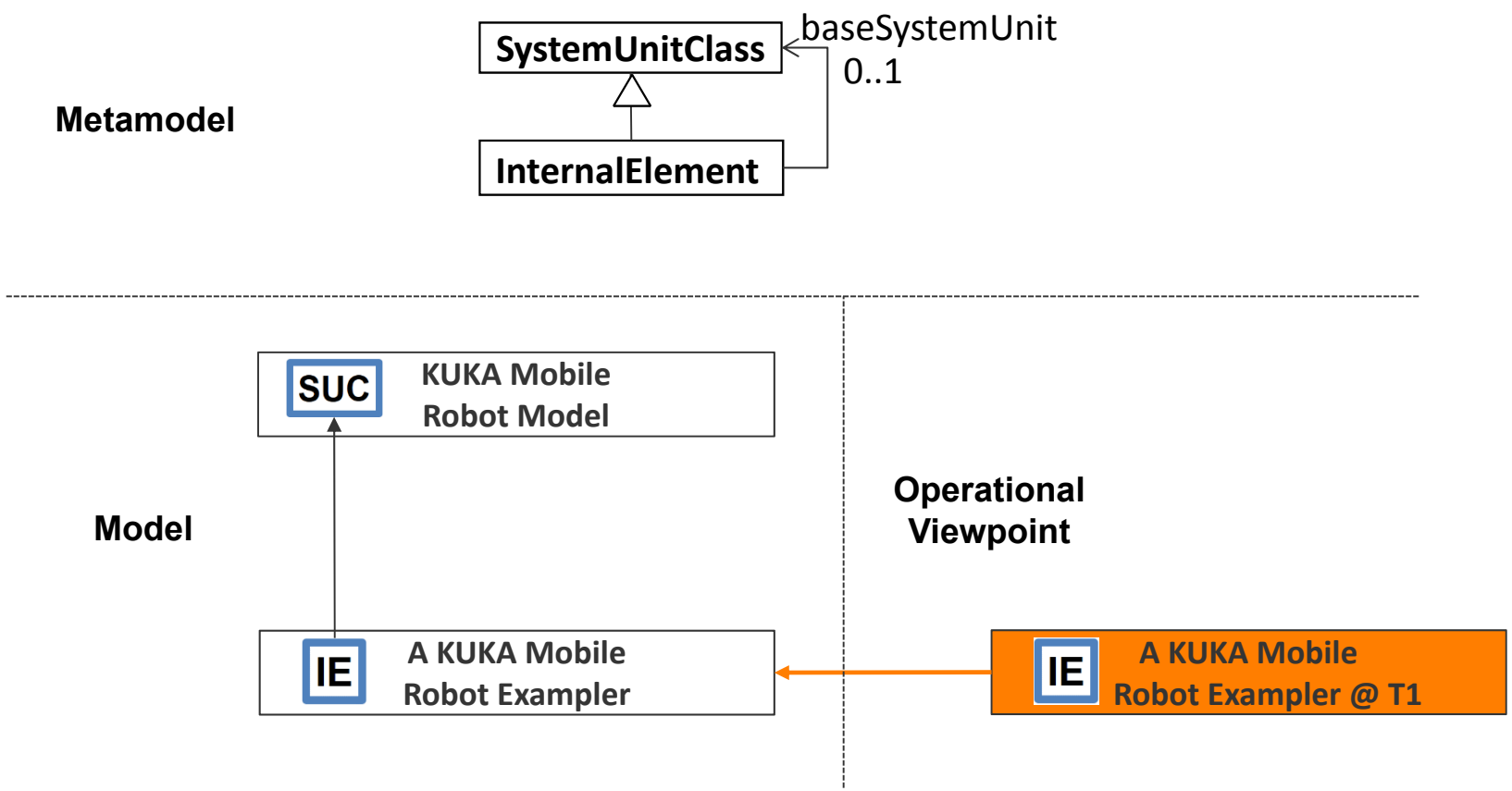



```
fix title : "Add missing slots from prototype" do
  for (pS : self.prototype.slots)
    if (not self.slots -> exists(cS | cS.name = pS.name))
      self.addSlot(pSlot.copy())
```

Challenge 3: Multiple Instantiations (1/2)

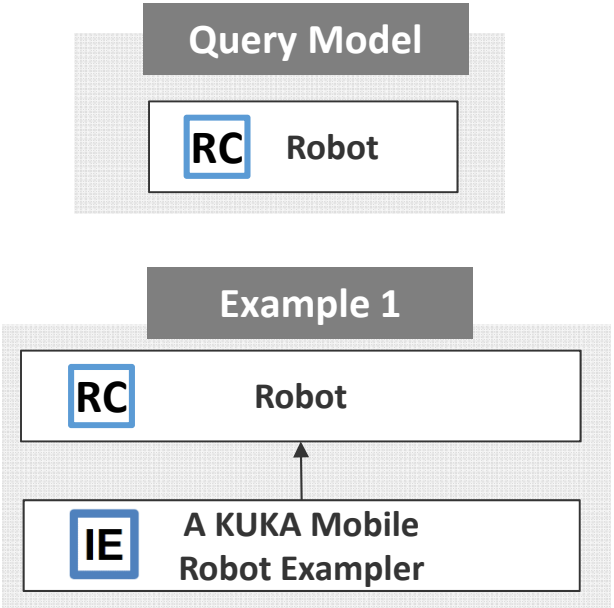


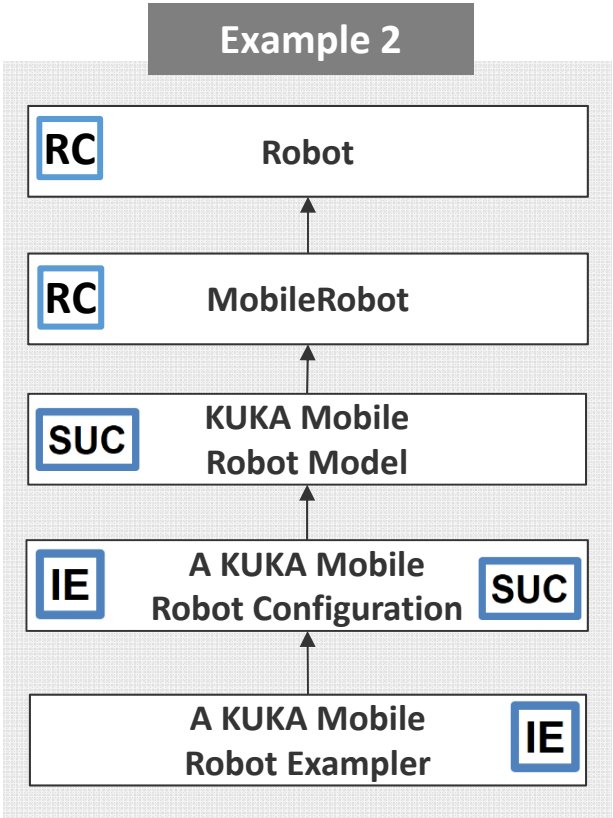
Challenge 3: Multiple Instantiations (2/2)



Challenge 5: Deep Query Support

- Domain experts need automated queries to access models
  - For instance: find all robots in a system
- Automation Query Language (AQL)
  - By-example query language for AML



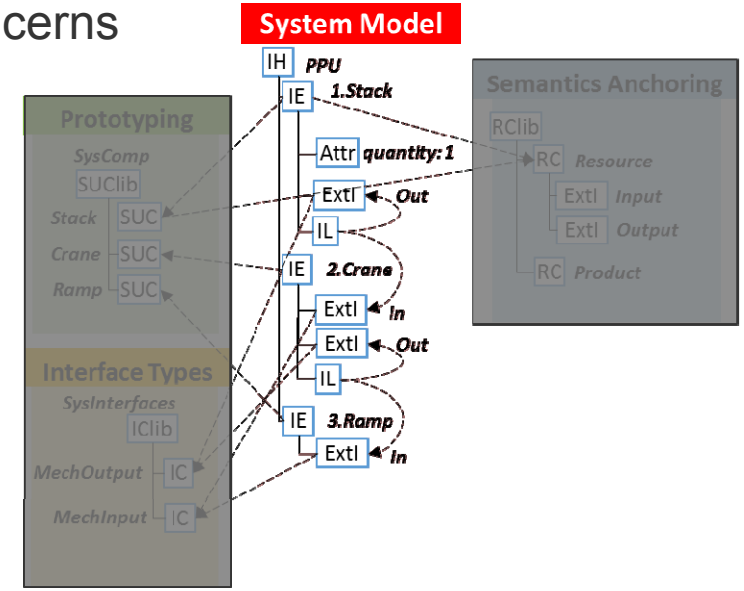


M. Wimmer, A. Mazak: "**From AutomationML to AutomationQL: A By-Example Query Language for CPPS Engineering Models**"; Proc. of the 14th International Conference on Automation Science and Engineering (CASE), (2018).

Lessons Learned

Lesson Learned #1: Technology

- Model-based technologies **speed-up tool development** drastically
  - From years to weeks
- **Two-level** based technologies seem **sufficient for data exchange**
  - Especially for syntactic concerns at a first glance
  - But may not be enough for semantic concerns
- Multi-level based technologies seem handy for providing
  - Semantics
  - Pragmatics
  - Additional modeling features
- Help is needed!



Lesson Learned #2: Organization

- **Levels help** to organize work
  - Standardization
  - Component provider
  - Component configuration
  - Systems engineering
  - Systems operation

Ontology
(Standardization Bodies)

Components
(Vendors)

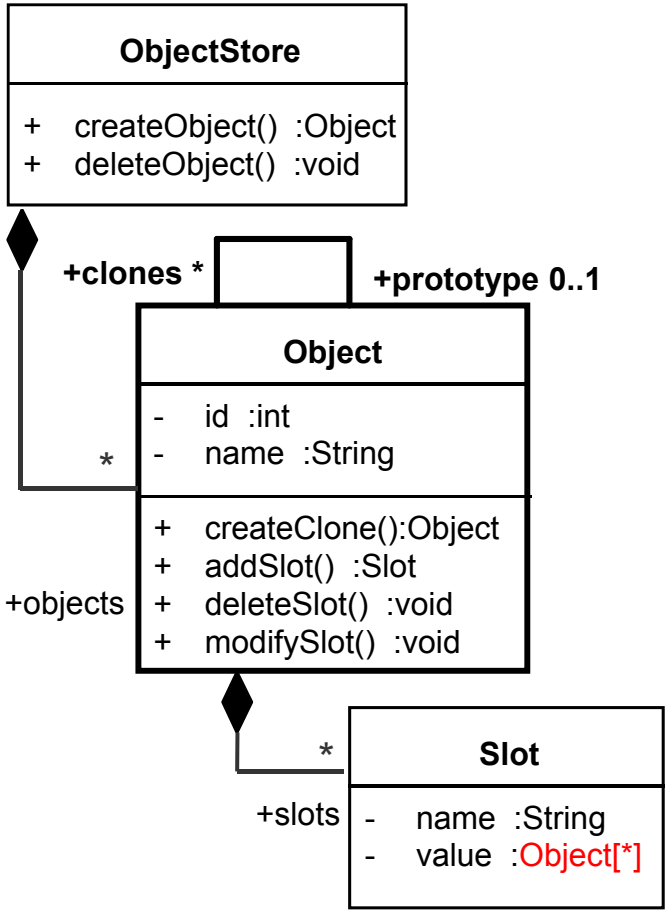
Systems
(Engineers)

- Important for
  - Keeping the language constant
  - Allowing for different model imports
  - Building up large component repositories
  - New applications: ordering a robot, virtual commissioning, etc.

- Currently not flexible enough!

Lesson Learned #3: Language Design

- **Reducing CAEX** to **Clabject** modeling language
- Constraints for modeling concepts on a particular level
- Objects are potentially trees of objects
- Modernization of CAEX modeling capabilities
  - Potency
  - Constraints
  - …



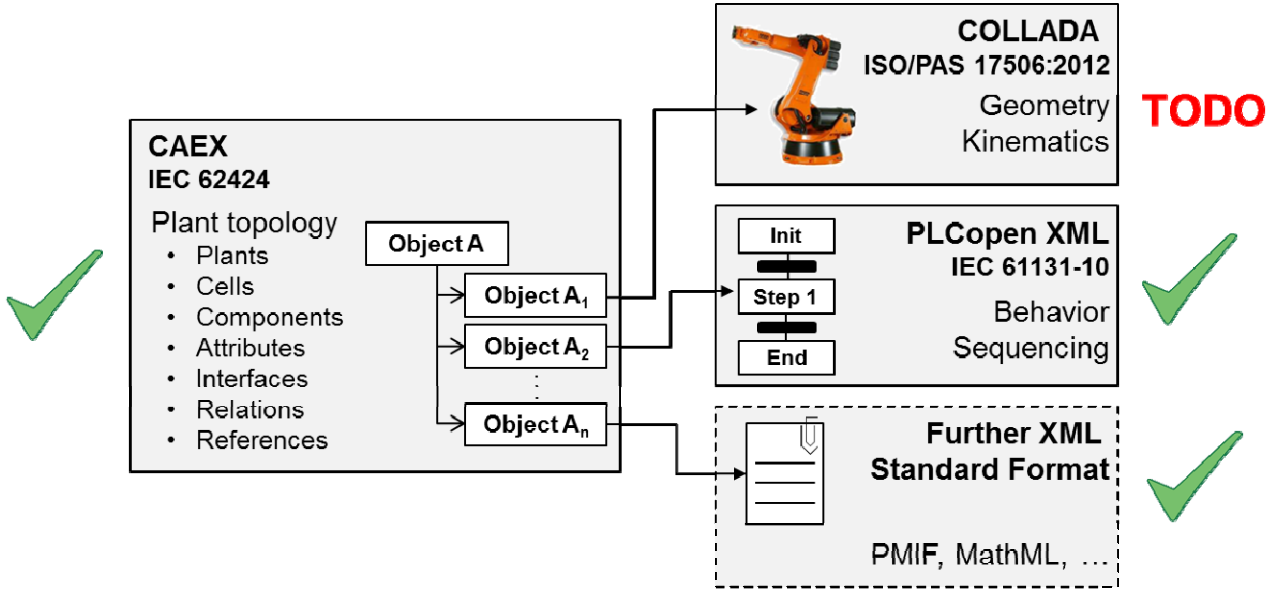
Lesson Learned #4: Hurdles

- **Conflict**: Structural guidance vs. flexibility
  - CAEX provides dedicated concepts for dedicated purposes
- Commonalities between different concepts not clearly abstracted
  - SUC <- IE, RC <- IE, RC <- SUC, …
- Type/Instance Pattern already a big step – what about Clabjects?
  - Engineers are often used to model solely on the instance level
- More expressive language needed, but CAEX is the standard
  - How to realize Industry 4.0 requirements?
  - Introduce further language concepts as role classes
- **Not-invented-here syndrom!**

Wrap-Up

Wrap-Up

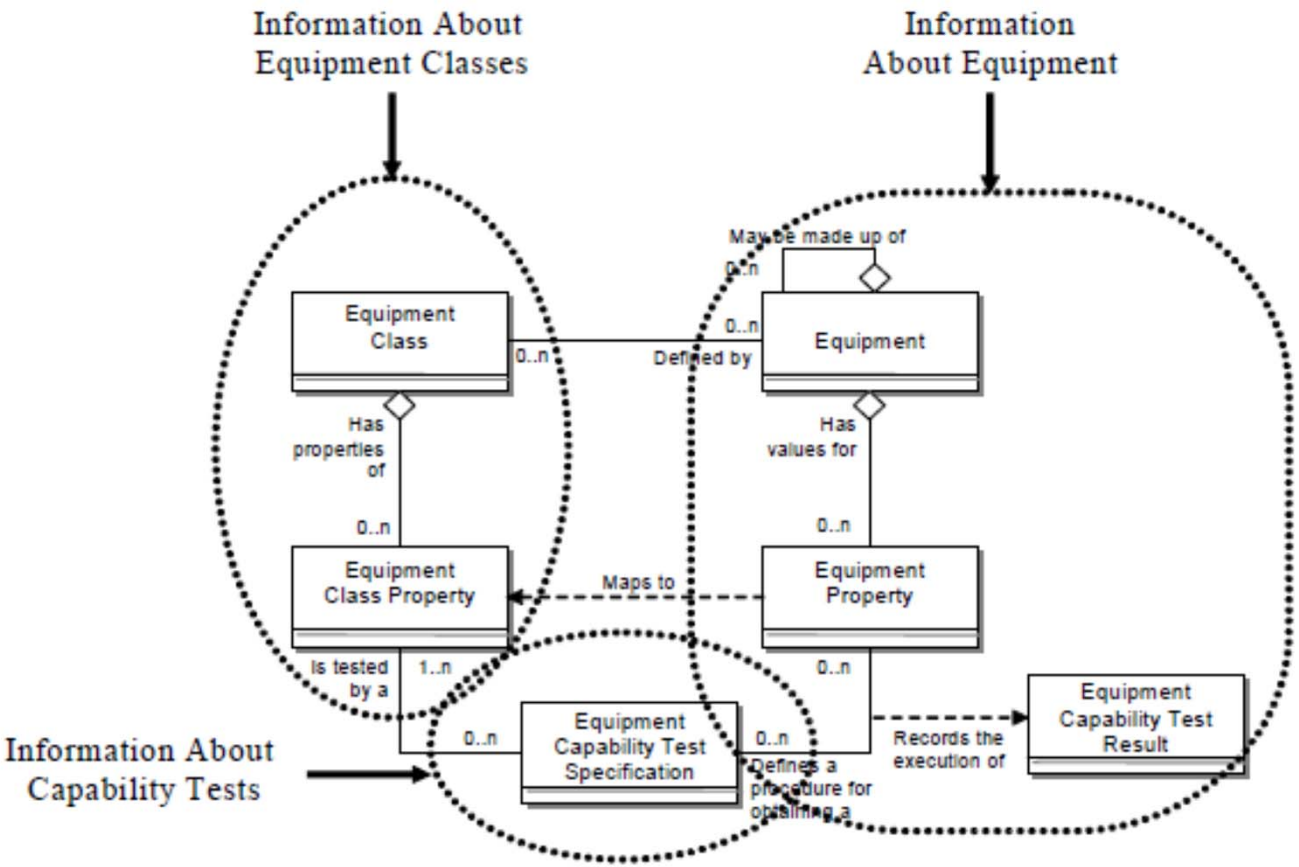
- Most of AutomationML's languages are already supported by
  - Metamodels (Ecore-based)
  - Additional Constraint Checks (OCL-based)
  - Transformations to other languages (SysML, PMIF, …)



→ Starting point for ML-AML?

Friends of AML: ISA-95 Information Models & Co.

B2MML: XML-based language for defining the Manufactoring Execution System (MES)

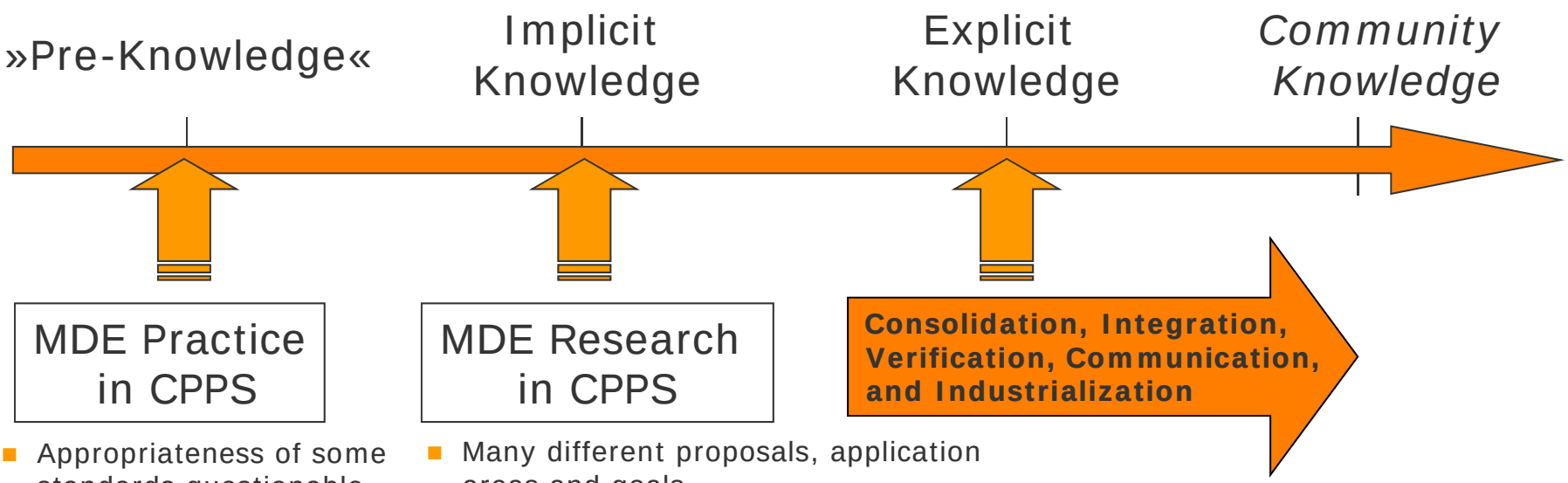


Source: IEC 62264-1
Enterprise-Control System Integration
Part 1: Models and Terminology

The Future

Our Main Challenge

How to transition from implicit to explicit knowledge about MDE in particular fields?



- Appropriateness of some standards questionable (SysML) – not yet adopted
- Automation-tool vendors jump on the MDE bandwagon

- Many different proposals, application areas and goals
  - E.g., DSMLs, Models@Runtime, Model-based Testing, Simulation, Validation, and Verification, Multi-Paradigm Modeling, Model Management, Megamodeling, **Multi-Level Modeling** …
- Emerging standards and initiatives
  - E.g., ISA-95, AutomationML, OPC-UA, …

C.W. Choo: **The Knowing Organization: How Organizations Use Information to Construct Meaning, Create Knowledge, and Make Decisions**, Oxford University Press, 1998.

Many thanks to…

- Tanja Mayerhofer
- Gerti Kappel
- Alexandra Mazak
- Sabine Wolny
- Luca Berardinelli
- Emanuel Mätzler
- Rainer Drath
- Arndt Lüder
- **and many more …**

…

Thank you!

Comments? Questions? Feedback?

Manuel Wimmer
wimmer@big.tuwien.ac.at

Christian Doppler Laboratory (CDL-MINT)

https://cdl-mint.big.tuwien.ac.at