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
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
  - ...

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

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

- **Whitepapers**
  - Application Recommendations
  - Best Practice Recommendations

- AutomationML website: [http://www.automationml.org](http://www.automationml.org)
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

CAEX
IEC 62424
Plant topology
- Plants
- Cells
- Components
- Attributes
- Interfaces
- Relations
- References

Object A

Object A₁

Object A₂

...  

Object Aₙ

COLLADA
ISO/PAS 17506:2012
Geometry
Kinematics

Init

Step 1

End

PLCopen XML
IEC 61131-10
Behavior
Sequencing

Further XML
Standard Format

AutomationML in a Nutshell
https://www.automationml.org/o.red/uploads/dateien/1447420977-
AutomationML%20in%20a%20Nutshell_151104.pdf
AutomationML = Automation (Markup | Modeling) Language?

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**

**CAEX**

**IEC 62424**

Plant topology
- Plants
- Cells
- Components
- Attributes
- Interfaces
- Relations
- References

**COLLADA**

ISO/PAS 17506:2012
Geometry
Kinematics

**PLCopen XML**

IEC 61131-10
Behavior
Sequencing

Further XML
Standard Format

Further engineering data
AutomationML: Automation Modeling Language?

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

CAEX
IEC 62424
Plant topology
- Plants
- Cells
- Components
- Attributes
- Interfaces
- Relations
- References

Object A
- Object A₁
- Object A₂
- ... Object Aₙ

COLLADA
ISO/PAS 17506:2012
- Geometry
- Kinematics

PLCopen XML
IEC 61131-10
- Behavior
- Sequencing

Further XML
Standard Format
- Further engineering data
<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.

```
<ObjectA>
  <ObjectA_1>
  <ObjectA_2>
    <ObjectA_2_1>
  </ObjectA_2>
</ObjectA>
```

```
<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>
</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:element 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:sequence>
  </xs:element>
</xs:schema>
```
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](https://github.com/amlModeling)

![Diagram of metamodel creation process](image)

CAEX is a language which is now also specified by a metamodel.

CAEX,.ecore

«conforms to»

example.aml

https://eclipse.org/modeling/emf/
Example: Pick And Place Unit (PPU)

- 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

System Model

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

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

System Model

IClib Set of Interface Classes
IC Interface Class

Interface Types
SysInterfaces ICLib
MechOutput IC
MechInput IC

PPU
1. Stack

Attr quantity: 1

Out

IE

In

2. Crane

IE

In

3. Ramp

IE

Out

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

System Model

Prototyping

SysComp

SUC

Stack

SUC

Crane

SUC

Ramp

SUC

Interface Types

SysInterfaces

IClib

MechOutput

IC

MechInput

IC

PPU

1.Stack

Attr

tquantity: 1

Extl

Out

IE

IL

2.Crane

Extl

In

IE

IL

3.Ramp

Extl

Out

IE

IL

SUC

Set of reusable modeling elements

SUC

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

System Model

Prototyping
- Stack
- Crane
- Ramp

Interface Types
- SysComp
- SUC
- MechOutput
- MechInput

Example: PPU in CAEX

Semantics Anchoring
- RCLib
- RC

Set of Role Classes
- 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

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-formedness 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

Diverse Plant Behavior Modeling Languages

- Tool A
  - Gantt Chart
- Tool B
  - PERT Chart
  - Impulse Diagram
- Tool C
  - ...

PLCopen XML (IEC 61131-10)
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

```
<IFC>
<actionBlock>
<action>
<data name="http://AutomationML"/>
<ActionStatus>
current
</ActionStatus>
</AutomationML>
</data>
</action>
</actionBlock>
```

![Diagram of IML and Gantt Chart, PERT Chart, Impulse Diagram, Tool A, Tool B, Tool C, Diverse Plant Behavior Modeling Languages, Intermediate Modeling Layer (IML), PLCopen XML (IEC 61131-10)]
IML as Semantic Domain and Integration Layer for Plant Behavior Models

Model Execution

Model Simulation
Consistency Analysis
Refinement Analysis

Tool A
Tool B
Tool C

Diverse Plant Behavior Modeling Languages

Intermediate Modeling Layer (IML)

PLCopen XML (IEC 61131-10)

MM Syntax
OS Semantics

M2M Syntax
Semantics
M2M OS

Gantt Chart
PERT Chart
Impulse Diagram

...
Tool Support for IML based on [https://github.com/moliz/moliz\_gemoc/tree/master/examples/iml](https://github.com/moliz/moliz\_gemoc/tree/master/examples/iml)
A Multi-Level Perspective
AutomationML Realization: Important Concepts/Relationships

Ecore/XSD/…

RoleClass (RC)

InternalElement (IE)

SystemUnit Class (SUC)
What is modelled with AutomationML?

Model

- **RC**: AutomationMLObject
- **RC**: Resource
- **RC**: Robot
- **RC**: MobileRobot
- **SUC**: KUKA Mobile Robot Model
- **IE**: A KUKA Mobile Robot Configuration
- **IE**: A KUKA Mobile Robot Exampler
- **IE**: A KUKA Mobile Robot Exampler @ RunTime

Metamodel

- **RoleClass**
- **SystemUnitClass**
- **InternalElement**

Ontology (Standardization)

Components (Vendors)

Systems (Engineers/Operators)
Most Basic Pattern: SUC <-> IE

- **Model**: SUC
  - RoleClass
  - SUC
  - KUKA Mobile Robot Model
  - IE
  - A KUKA Mobile Robot Exampler
  - A KUKA Mobile Robot Exampler @ Runtime

- **Metamodel**: RoleClass
  - SUC
  - IE
  - IE@RT

**Ontology (Standardization)**
- Resource
- Robot
- MobileRobot

**Components (Vendors)**
- AutomationMLObject

**Systems (Engineers/Operators)**
- RoleClass
Challenge 1: Relationship between IEs and SUCs

Metamodel

```
SystemUnitClass
  \------\------\
  \     /     /  baseSystemUnit
  \   /     /   0..1
  \ /     /   \
\ /     /   \
InternalElement
```

Model

- **SUC**: KUKA Mobile Robot Model
- **IE**: A KUKA Mobile Robot Exampler

Challenge 1: Relationship between IEs and SUCs

Metamodel

SystemUnitClass

InternalElement

baseSystemUnit 0..1

Model

SUC KUKA Mobile Robot Model

IE A KUKA Mobile Robot Exampler

PROTOTYPE

«conformsTo»
«manifestation»
«??»

CLONE

Semantics?

Pragmatics?
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

- SUC: KUKA Mobile Robot Model
- IE: A KUKA Mobile Robot Exampler

KUKA Mobile Robot Model
A KUKA Mobile Robot Exampler
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

ObjectStore

+ createObject() : Object
+ deleteObject() : void

+ clones *
+ prototype 0..1

Object

- id : int
- name : String

+ createClone(): Object
+ addSlot() : Slot
+ deleteSlot() : void
+ modifySlot() : void

+ objects

Slot

- name : String
- Value : Object[*]

objstore1: ObjectStore

obj1: Object

- id = 1
- name = "Motor"

s1: Slot

- name = "nominal speed"
- value = 9000

obj2: Object

- id = 2
- name = "Motor"

s2: Slot

- name = "nominal speed"
- value = 9000
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

**objstore1:ObjectStore**

**obj1: Object**
- id = 1
- name = "Motor"

**s1: Slot**
- name = "nominal speed"
- value = 9000

**obj2: Object**
- id = 2
- name = “Motor”

**s2: Slot**
- name = "nominal speed"
- value = 9000
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

context Object [self.prototype <> OclUndefined]
inv: self.prototype.slots \rightarrow forAll(pS \mid self.slots \rightarrow one(cS \mid pS.name = cS.name and pS.value = cS.value))

Level 1b: Restriction

inv: self.slots \rightarrow forAll(cS \mid self.prototype.slots \rightarrow one(pS \mid cS.name = pS.name and cS.value = pS.value))

Level 1c: Redefinition

context Object [self.prototype <> OclUndefined]
inv: self.prototype.slots \rightarrow forAll(cS \mid self.prototype.slots \rightarrow one(pS \mid cS.name = pS.name))
inv: self.slots \rightarrow forAll(pS \mid self.slots \rightarrow one(cS \mid pS.name = cS.name))

Level 2: Full Compliance
Post-condition of createClone() operation must always hold
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

ObjectStore

+ createObject() : Object
+ deleteObject() : void

Object

- id : int
- name : String
+ createClone() : Object
+ addSlot() : Slot
+ deleteSlot() : void
+ modifySlot() : void

Slot

- name : String
- Value : Object[*]

<table>
<thead>
<tr>
<th>Operation</th>
<th>L0</th>
<th>L1a</th>
<th>L1b</th>
<th>L1c</th>
<th>L2</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjectStore::createObject</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
</tr>
<tr>
<td>ObjectStore::deleteObject</td>
<td>≠</td>
<td>≠</td>
<td>≠</td>
<td>≠</td>
<td>≠</td>
</tr>
<tr>
<td>Object::addSlot</td>
<td>↑</td>
<td>≠</td>
<td>↑</td>
<td>≠</td>
<td>≠</td>
</tr>
<tr>
<td>Object::deleteSlot</td>
<td>↑</td>
<td>↑</td>
<td>≠</td>
<td>≠</td>
<td>≠</td>
</tr>
<tr>
<td>Object::modifySlot</td>
<td>↑</td>
<td>≠</td>
<td>≠</td>
<td>↑</td>
<td>↑</td>
</tr>
</tbody>
</table>

↑ non-breaking
≠ breaking

change on prototype
impact on consistency of clones
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

<table>
<thead>
<tr>
<th>Operation</th>
<th>L0</th>
<th>L1a</th>
<th>L1b</th>
<th>L1c</th>
<th>L2</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjectStore::createObject</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
</tr>
<tr>
<td>ObjectStore::deleteObject</td>
<td>≠</td>
<td>≠</td>
<td>≠</td>
<td>≠</td>
<td>≠</td>
</tr>
<tr>
<td>Object::addSlot</td>
<td>↑</td>
<td>≠</td>
<td>↑</td>
<td>≠</td>
<td>≠</td>
</tr>
<tr>
<td>Object::deleteSlot</td>
<td>↑</td>
<td>↑</td>
<td>≠</td>
<td>≠</td>
<td>≠</td>
</tr>
<tr>
<td>Object::modifySlot</td>
<td>↑</td>
<td>≠</td>
<td>≠</td>
<td>↑</td>
<td>≠</td>
</tr>
</tbody>
</table>

*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

---

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

```
Motor: Object
- id = 1
- name = "Motor"

M1: Object
- id = 2
- name = "M1"

: Slot
- name = "nominal speed"
- value = 9000

: Slot
- name = "min rotation speed"
- value = 5800

fix title: "Add missing slots from prototype"
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)

Metamodel

SystemUnitClass

InternalElement

baseSystemUnit

0..1

Model

SUC  KUKA Mobile Robot Model

A KUKA Mobile Robot Configuration

IE  A KUKA Mobile Robot Configuration

SUC  KUKA Mobile Robot Configuration

IE  A KUKA Mobile Robot Exampler
Challenge 3: Multiple Instantiations (2/2)

Metamodel

SystemUnitClass

baseSystemUnit 0..1

InternalElement

Model

KUKA Mobile Robot Model

A KUKA Mobile Robot Exampler

Operational Viewpoint

A KUKA Mobile Robot Exampler @ T1
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

Example 1

Example 2

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

- **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
  - ...

```
ObjectStore
+ createObject() :Object
+ deleteObject() :void

Object
- id :int
- name :String
+ createClone():Object
+ addSlot() :Slot
+ deleteSlot() :void
+ modifySlot() :void

+objects
+clones *

Slot
- name :String
- value :Object[*]
```
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
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 Manufacturing 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?

»Pre-Knowledge«

Implicit Knowledge

Explicit Knowledge

Community Knowledge

MDE Practice in CPPS

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

MDE Research in CPPS

- 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, ...

Consolidation, Integration, Verification, Communication, and Industrialization

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