Business Informatics Group

Vienna University of Technology

Cross-disciplinary Modeling – the Good, the Bad, and the Ugly

QRS 2016

August 2 2016, TU Wien

Gerti Kappel & Team

Business Informatics Group

Institute of Software Technology and Interactive Systems
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

Motivation

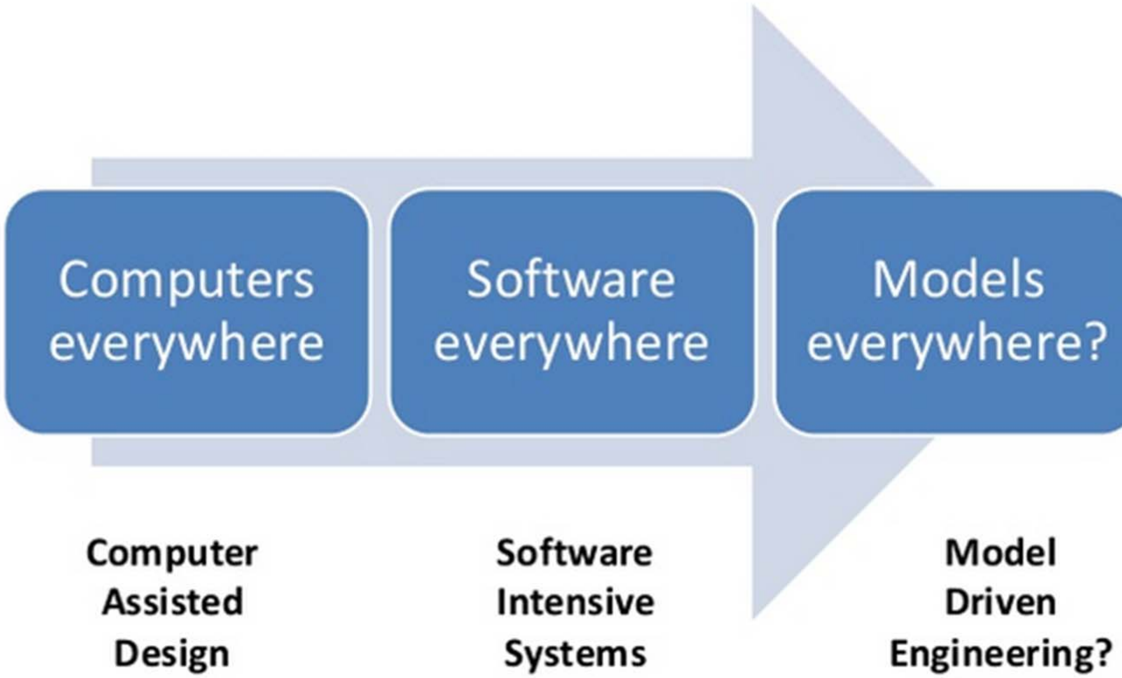
- The Good
  - Heterogeneity Engineering since Distributed Database Systems
  - Language / Transformation Engineering since Model-Driven Engineering
- The Bad
  - Dealing with Views, Interfaces, (In-)Consistencies still in its infancy
- The Ugly
  - Lots of implicit conventions, hidden knowledge around
  - Missing domain knowledge

Content

- Introduction
  - *Model-Driven Engineering in Software Engineering*
  - *Cyber-Physical Production Systems (CPPS)*
- MDE in CPPS I: *Interface Integration*
- MDE in CPPS II: *Model Exchange*
- Résumé

MDE: From Software to Systems

Main Motivation: Ubiquitous computing, software, models

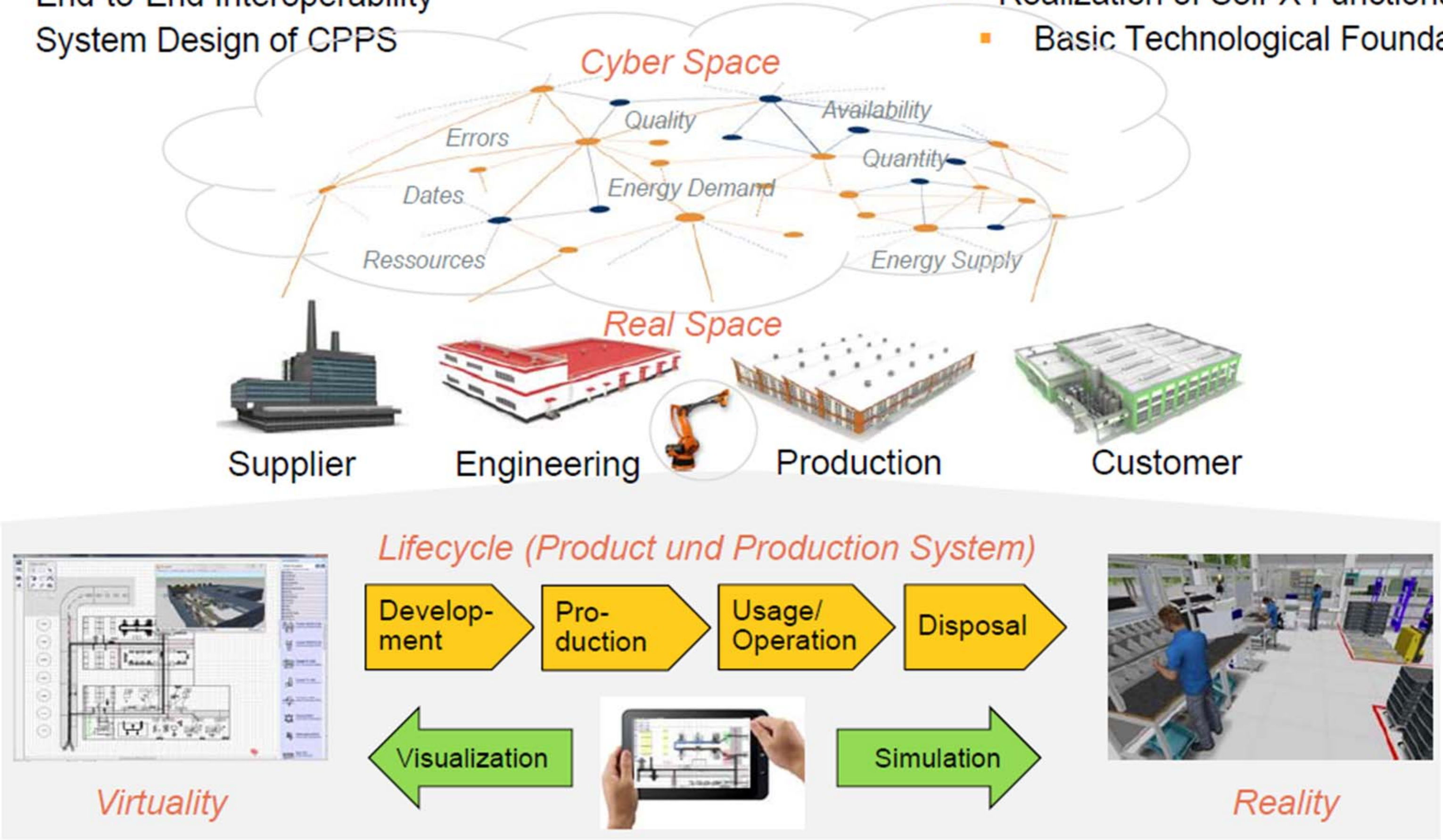


J. Bézivin. ***Software Modeling and the Future of Engineering***. STAF Keynote, 2014.

The CPPS Domain

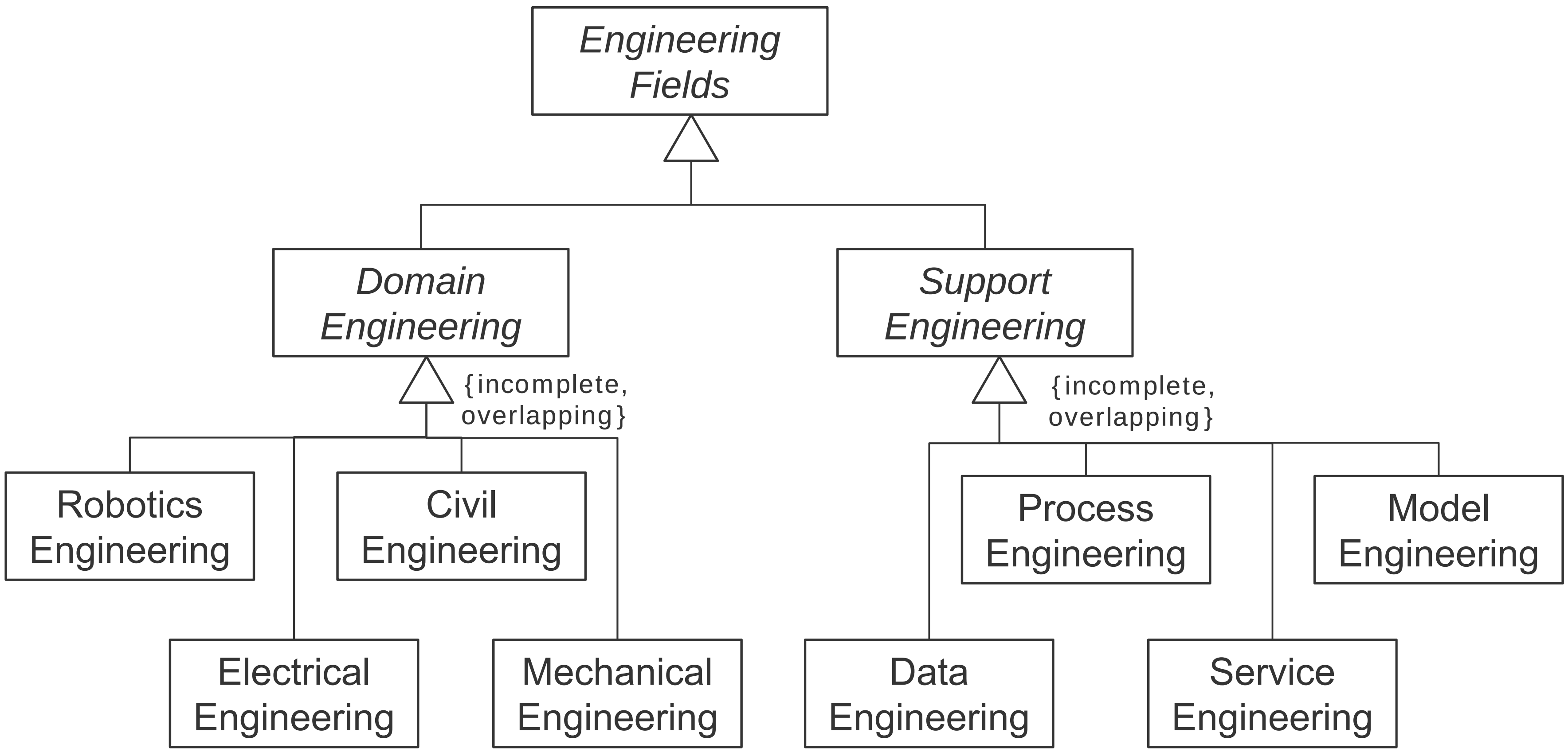
Scope and Scientific Challenges

- End-to-End Interoperability
- System Design of CPPS
- Realization of Self-X Functionalities
- Basic Technological Foundations



Separation of Problem Space and Solution Space

Domain Engineering vs. Support Engineering



J. Bézivin. ***Software Modeling and the Future of Engineering***. STAF Keynote, 2014.

Taking a closer look at CPPS

Problem and Solution Clusters

- **Sub-Domains**
  - Economics
  - Logistics
  - Internet of Things
  - Mechanics
  - Electrical Engineering
  - Mechatronics
  - Control Engineering
  - Enterprise Engineering
  - Robotics
  - …

Main Characteristics
- *Multi-disciplinary field*
- *Socio-Cyber-Physical Systems*

- **Support Engineering**
  - Product Line Engineering
  - Model Engineering
  - Ontology Engineering
  - Requirement Engineering
  - Component Engineering
  - Document Engineering
  - Agent Engineering
  - …

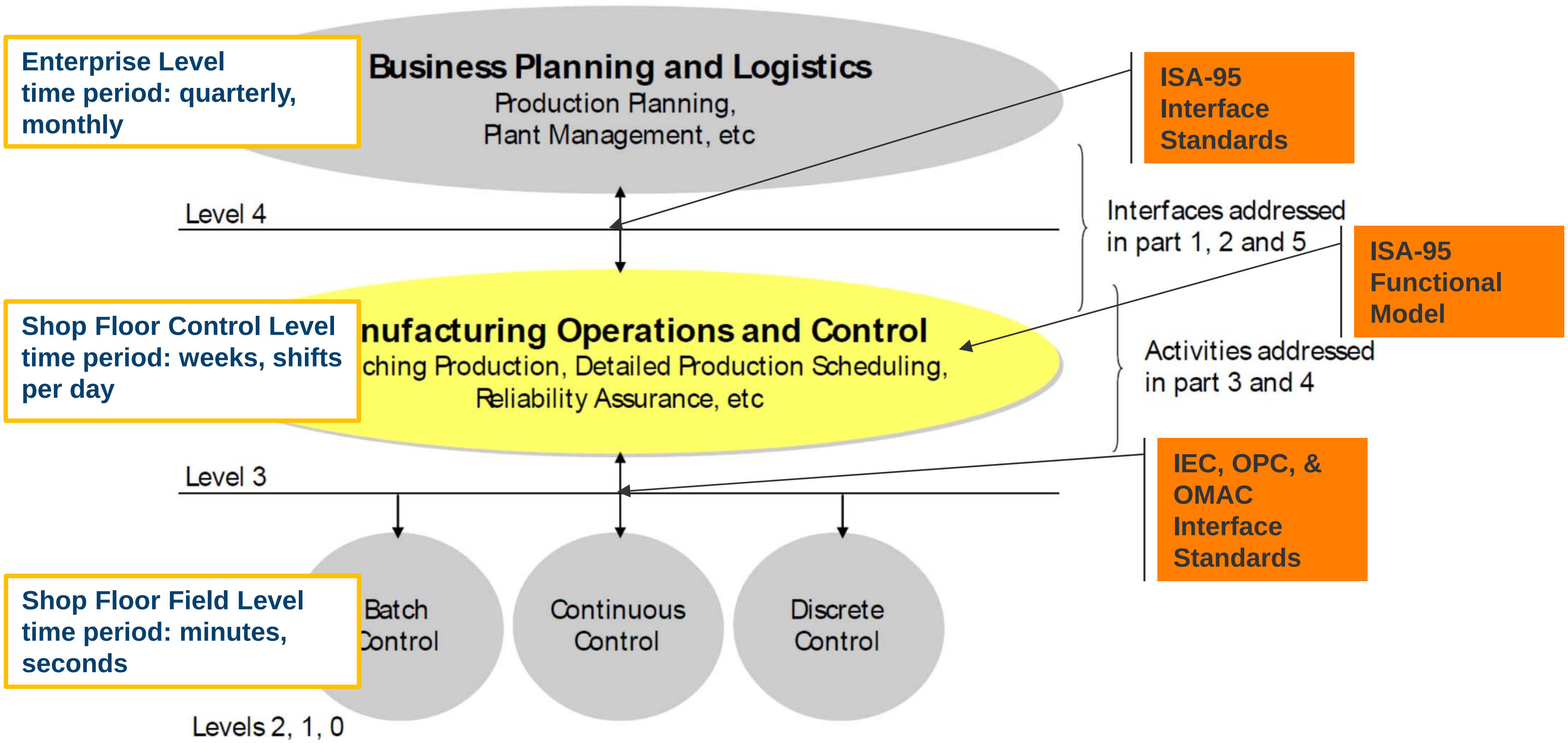
Main Challenges
- *Which solutions are usable by domain engineers?*
- *Which adaptations are necessary for the specific domain?*

Content

- Introduction
  - *Modeling and Model-Driven Engineering in Software Engineering*
  - *Cyber-Physical Production Systems (CPPS)*
- **MDE in CPPS I: *Interface Integration***
- MDE in CPPS II: *Model Exchange*
- Résumé

Interfaces within a manufacturing company

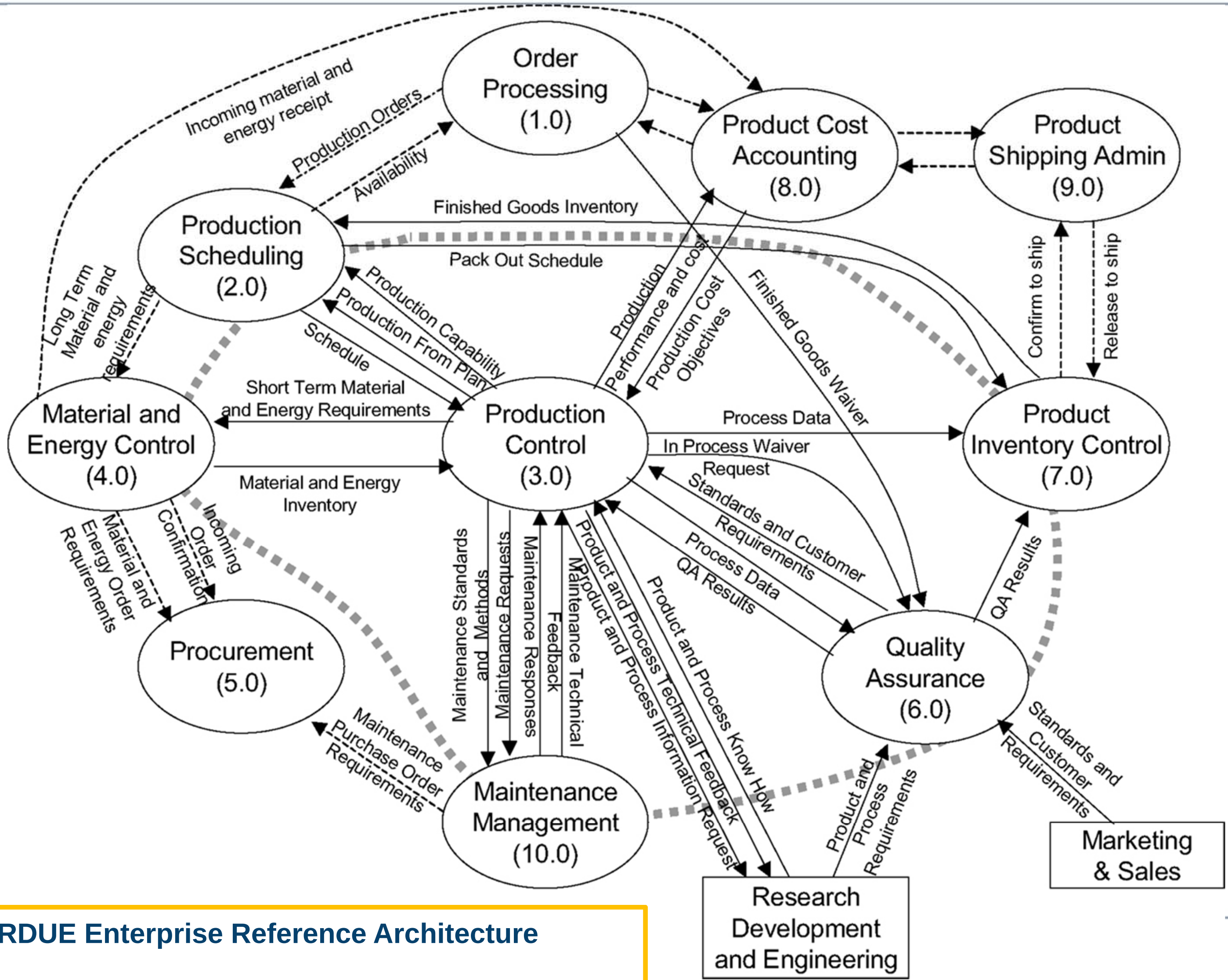
Clear definition of system boundaries (ERP – MES – SCADA/RFID/PLC)



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

Functional Enterprise-control model

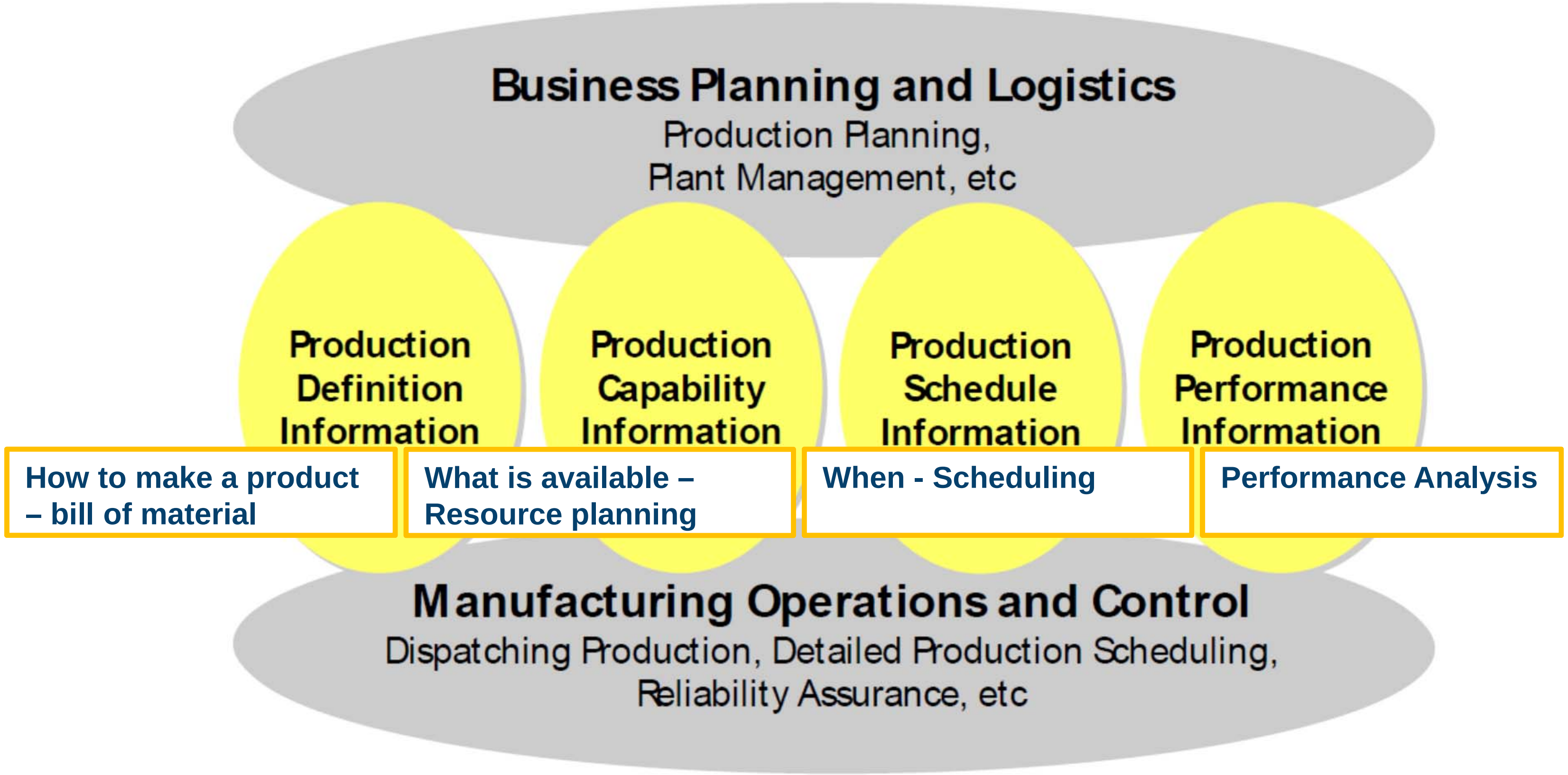
Source: IEC 62264-1
Enterprise-Control System Integration
Part 1: Models and Terminology



Based on the PURDUE Enterprise Reference Architecture

- **The model describes 31 information flows between the enterprise domain (level 4) and the control domain (level 3)**

Types of Information Exchange between Level 4 and Level 3

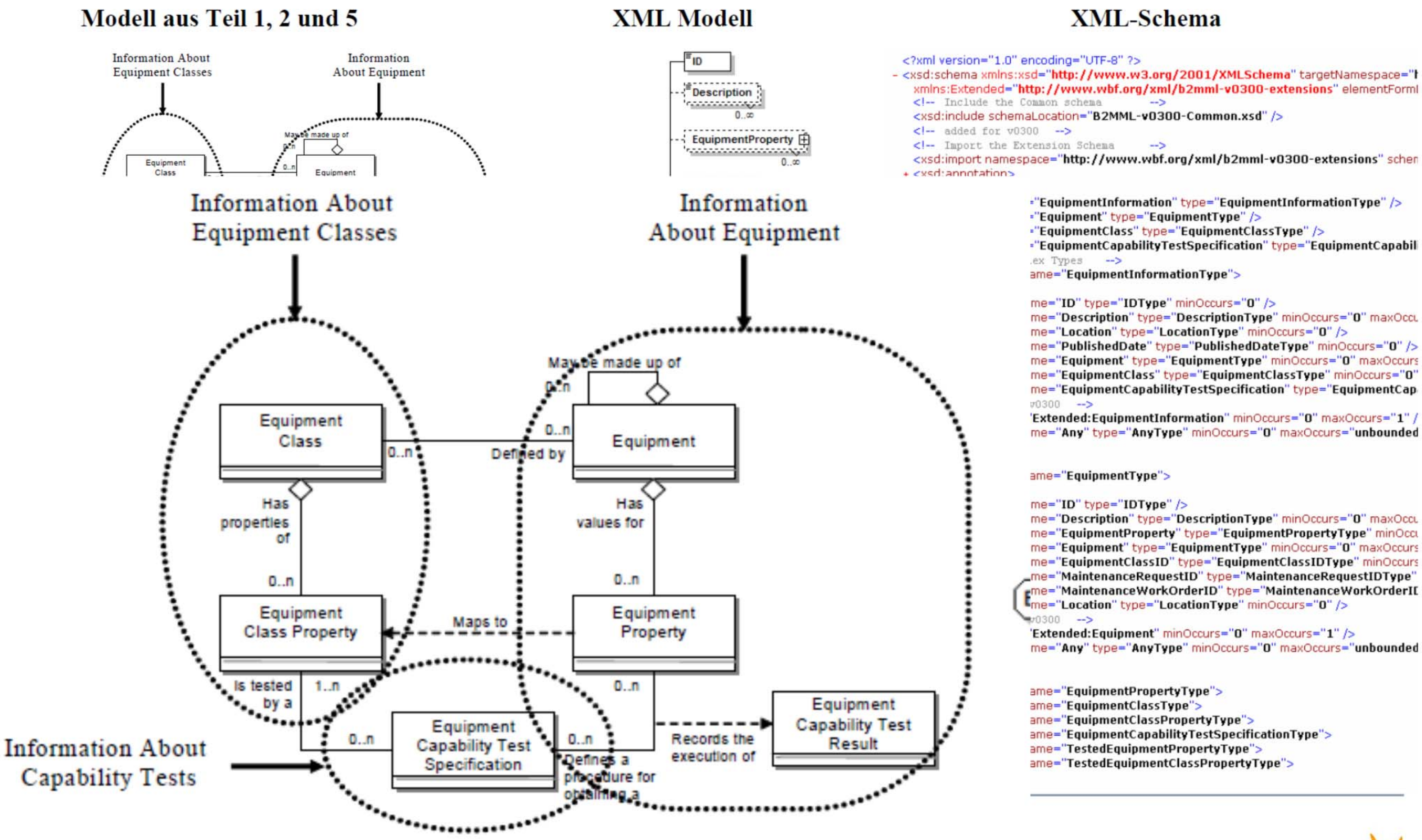


Source: IEC 62264-1
Enterprise-Control System Integration
Part 1: Models and Terminology

ISA-95 Information models

B2MML: XML serialization of the ISA-95 models

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



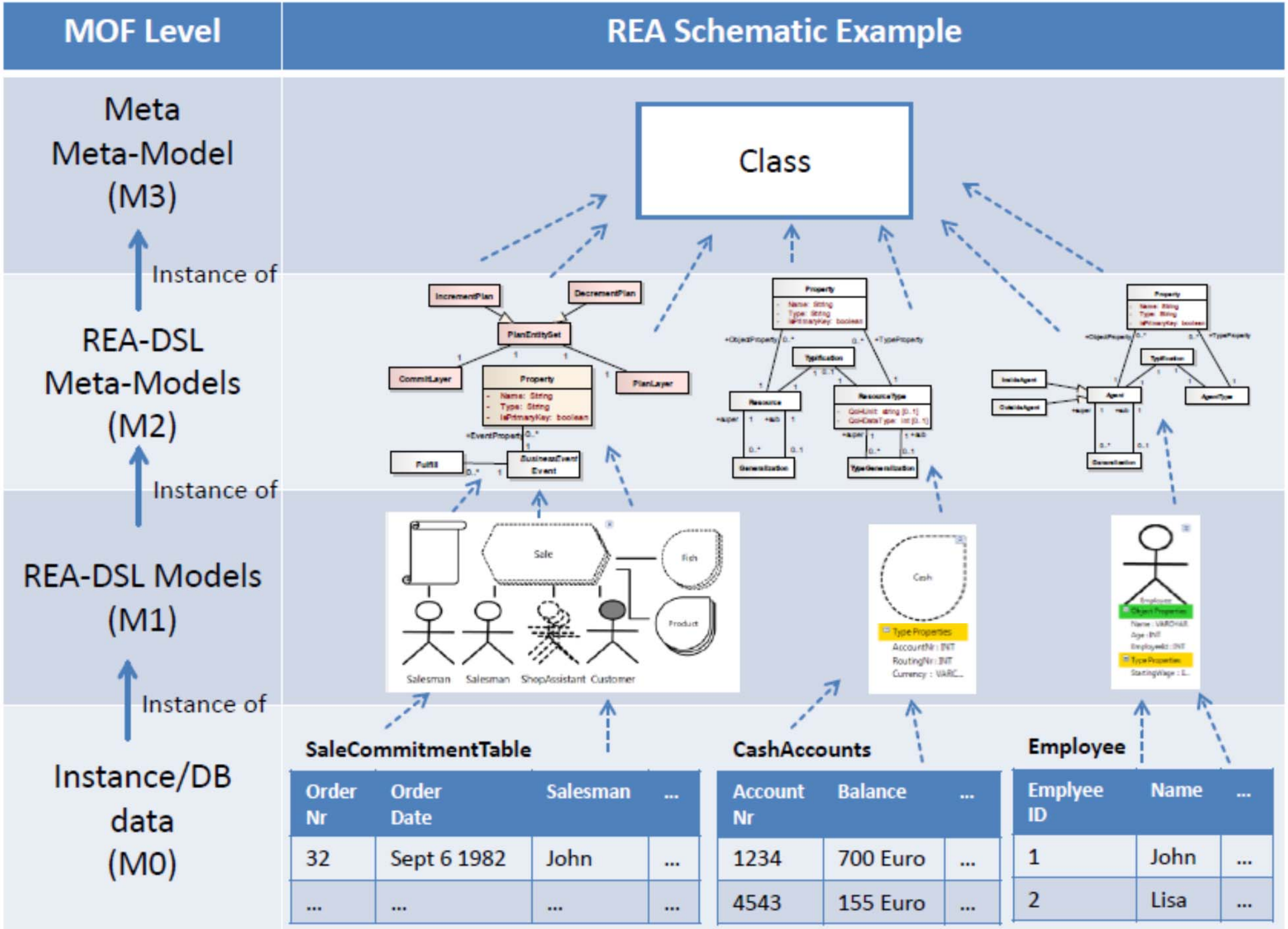
REA Ontology (ISO 15944-4)

Resource Event Agent Business Ontology

- RESOURCES: goods, services, labor, rights – have utility, are scarce and are under the control of a legal or natural person
- EVENTS: are occurrences in time that relate subsequent process states to each other
  - Increment event: gaining control of a resource
  - Decrement event: loosing control of a resource
- AGENTS: enterprises, departments, persons (accountable for, participate in, initiate)
- REA differs two kinds of business activities
  - EXCHANGE (Transfer) exchange of resources between business partners
  - TRANSFORMATION *„production process“* - implicit exchange and conversion (use, consume, produce)

REA Meta Model

Source: FFG BRIDGE-Project REAlist
Mayrhofer, Mazak, Wally, Kratzwald, Huemer, 2014



SaleCommitmentTable

Order Nr	Order Date	Salesman	...
32	Sept 6 1982	John	...
...	...	...	...

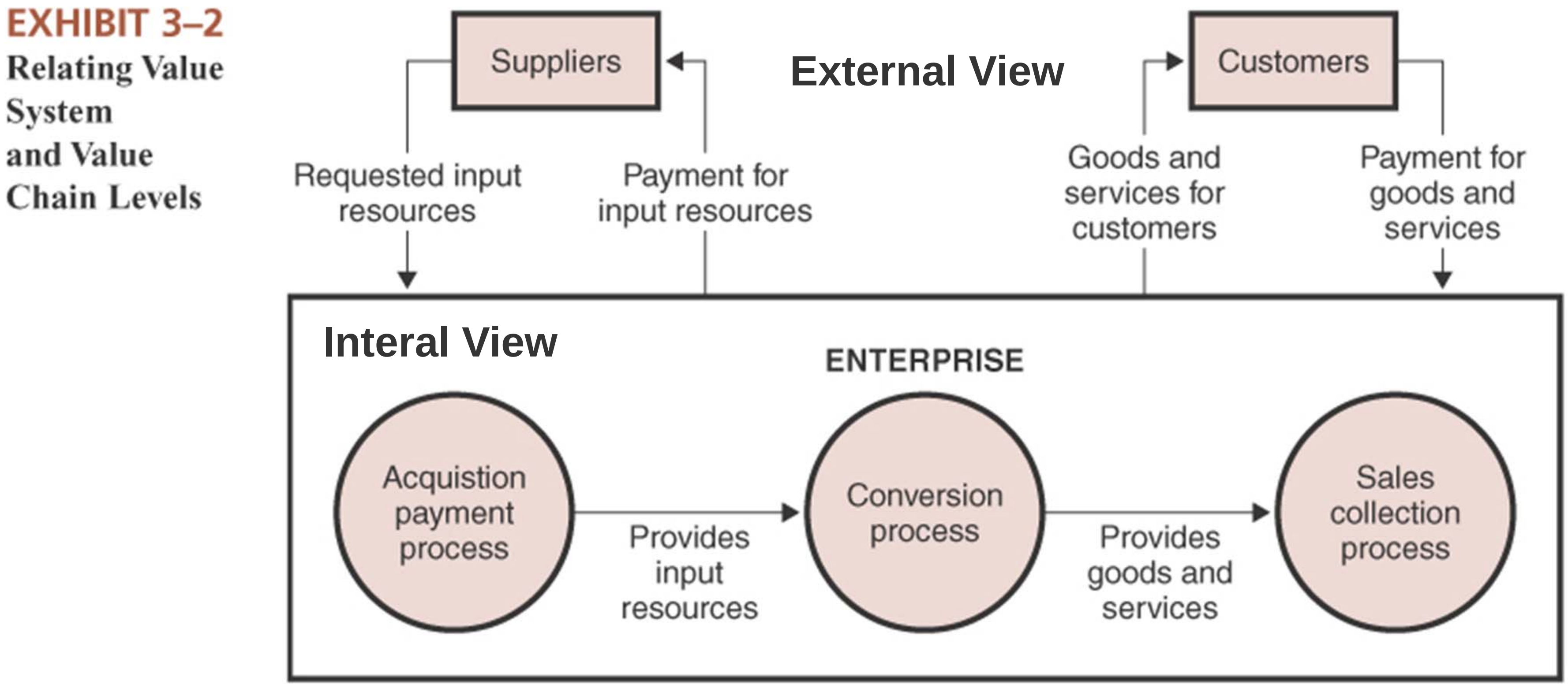
CashAccounts

Account Nr	Balance	...
1234	700 Euro	...
4543	155 Euro	...

Employee

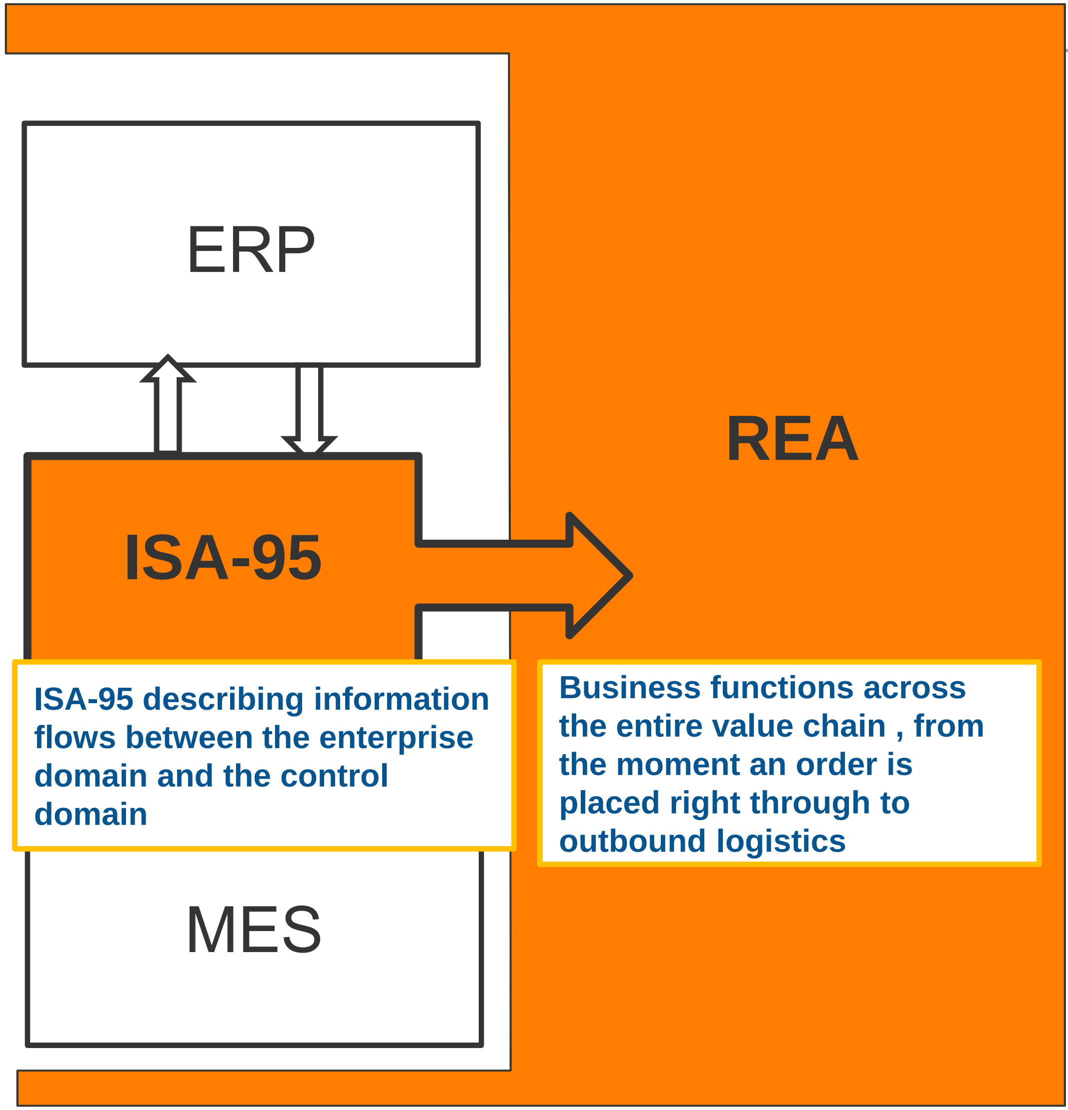
Emplyee ID	Name	...
1	John	...
2	Lisa	...

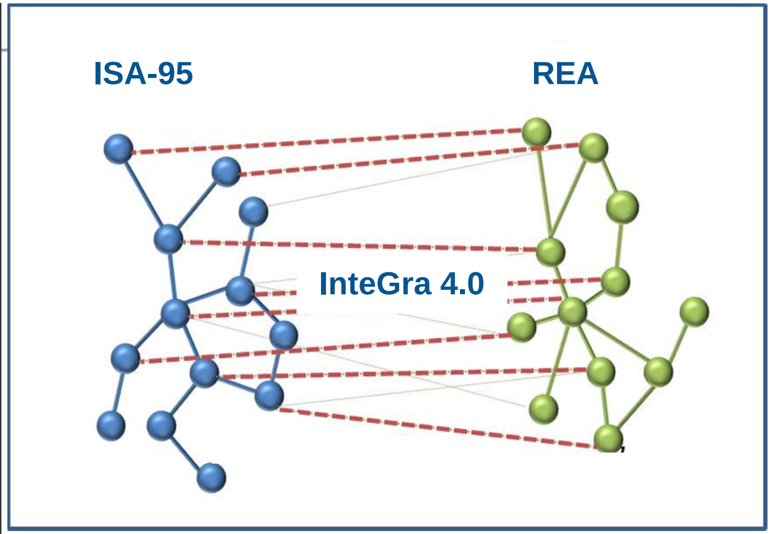
REA: Value Net and Value Chain



Source: Enterprise Information Systems:
A Pattern-Based Approach, Dunn et al., 2004

InteGra 4.0 Approach





Model-driven Smart Engineering: Alignment of the concepts of REA and the models of ISA-95

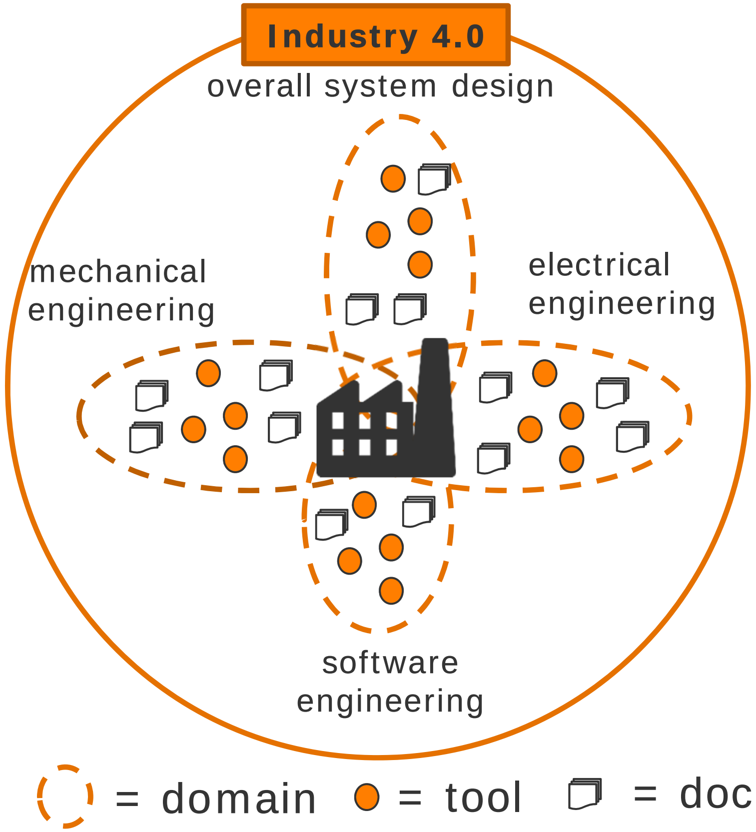
- **Horizontal integration** through value networks
- **Vertical integration** and networked manufacturing systems
- **End-to-end digital integration** of engineering across the entire value chain

Content

- Introduction
  - *Modeling and Model-Driven Engineering in Software Engineering*
  - *Cyber-Physical Production Systems (CPPS)*
- MDE in CPPS I: *Interface Integration*
- **MDE in CPPS II: *Model Exchange***
- Résumé

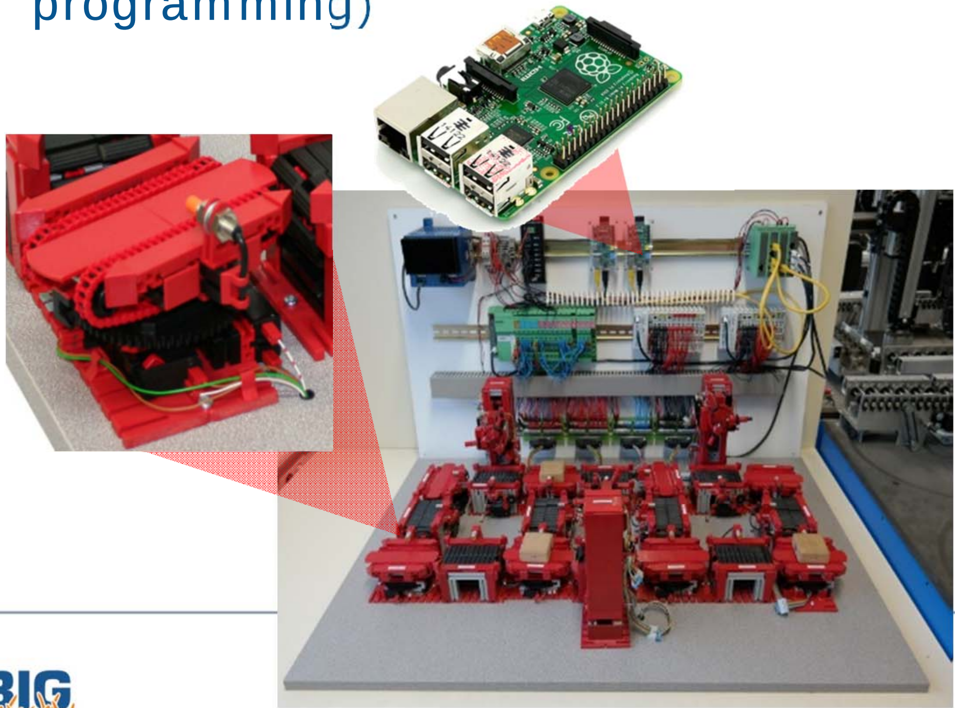
Engineering of CPPS

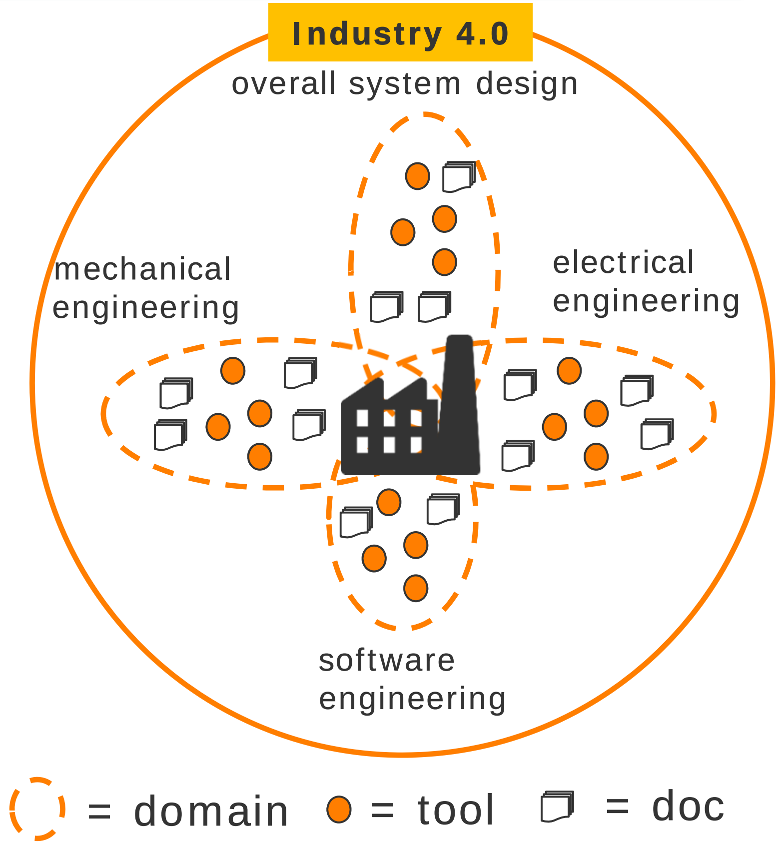
- **Industry 4.0**: **computerization** of manufacturing
- **Principles**
  - **Interoperability**: the ability of CPPS and humans to connect and communicate
  - **Virtualization**: a virtual copy of the factory with sensed data
  - **Decentralization**: the ability of CPPSs to make decisions on their own
  - **Real-time capability**: monitoring, analysis, planning, execution
  - **Modularity**: flexible adaptation of smart factories to changing requirements
  - …
- **Challenges**
  - **Multi-disciplinary** domain
  - **Heterogeneous** document/tool landscape
  - …



Introduction: Engineering of industrial production systems

Lab-sized flexible manufacturing system:
- **hardware parts**: turntables, motor, Raspberry PI, Field I/O modules, electrical wirings.
- **software**: Raspberry Pi programs (IEC 61131-3 standards, PLC programming)

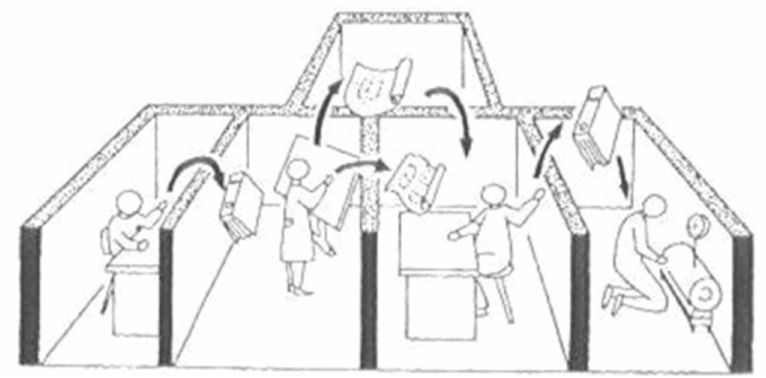


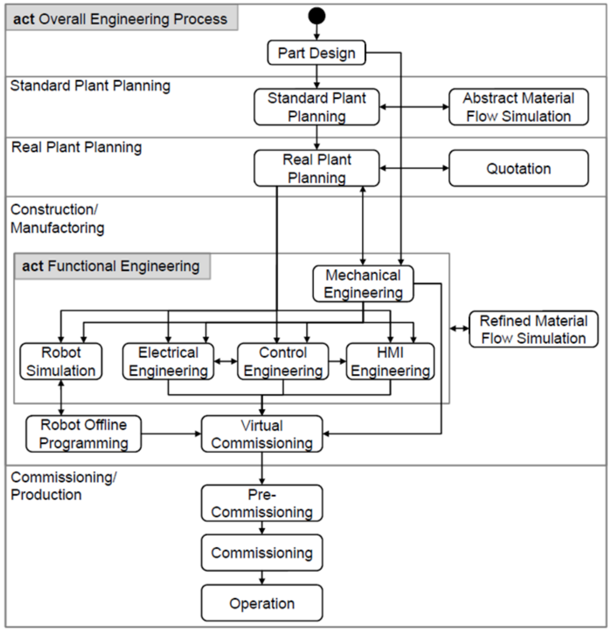


- Equipment Center for Distributed Systems, Institute of Ergonomics, Manufacturing Systems and Automation at Otto-v.-Guericke University Magdeburg.

Problem Description

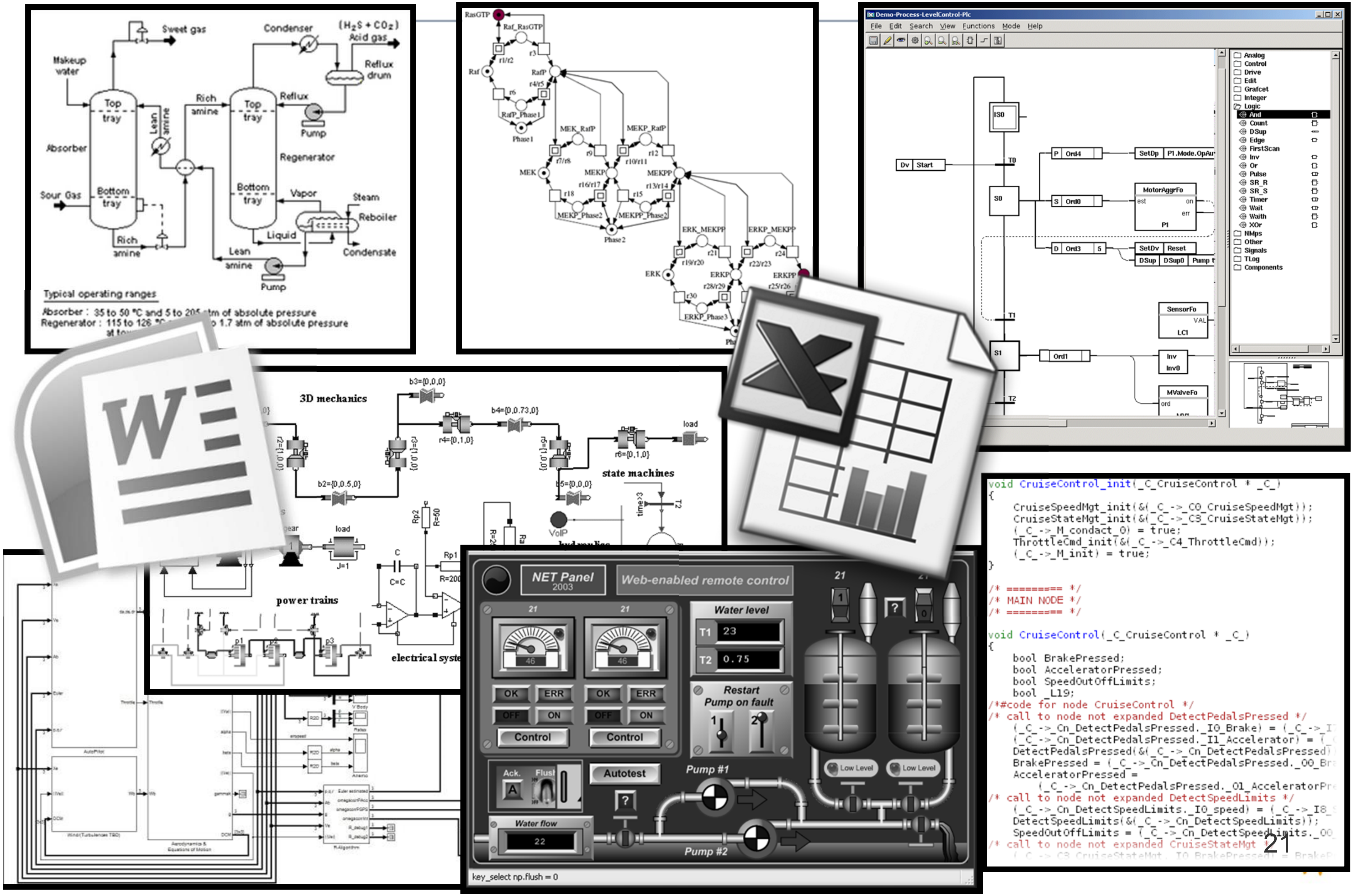
- Different engineering disciplines are involved in the engineering process
- Engineering steps are often done in parallel
- Current solutions often lack support for…
  - Versioning
  - Linking different engineering artefacts





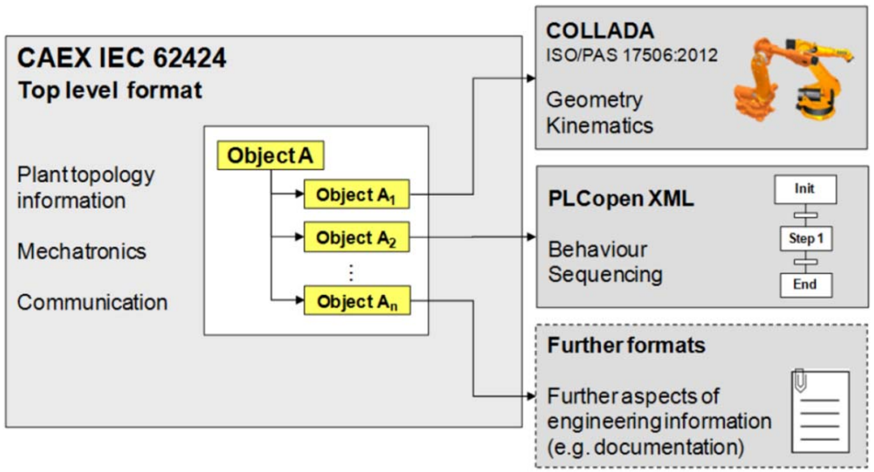
Typical industrial plant engineering process
R. Drath, B. Schröter, and M. Hoernicke, "Datenkonsistenz im Umfeld heterogener Engineering-Werkzeuge", in Automation Conference, 2011, pp. 29-32.

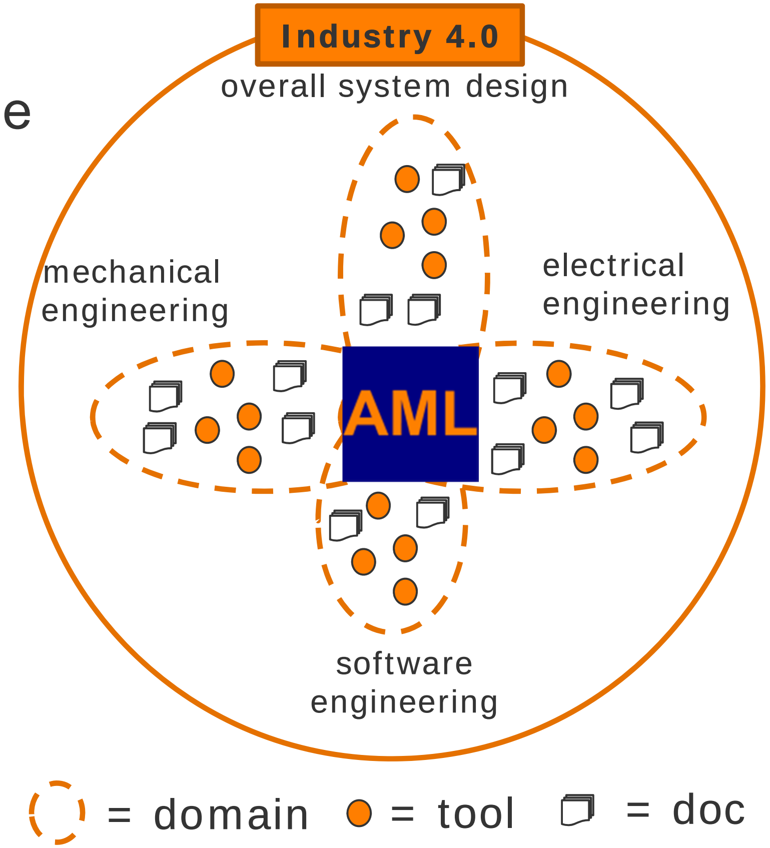
Artefacts found in CPPS Engineering Process



Engineering of CPPS: Common Format?

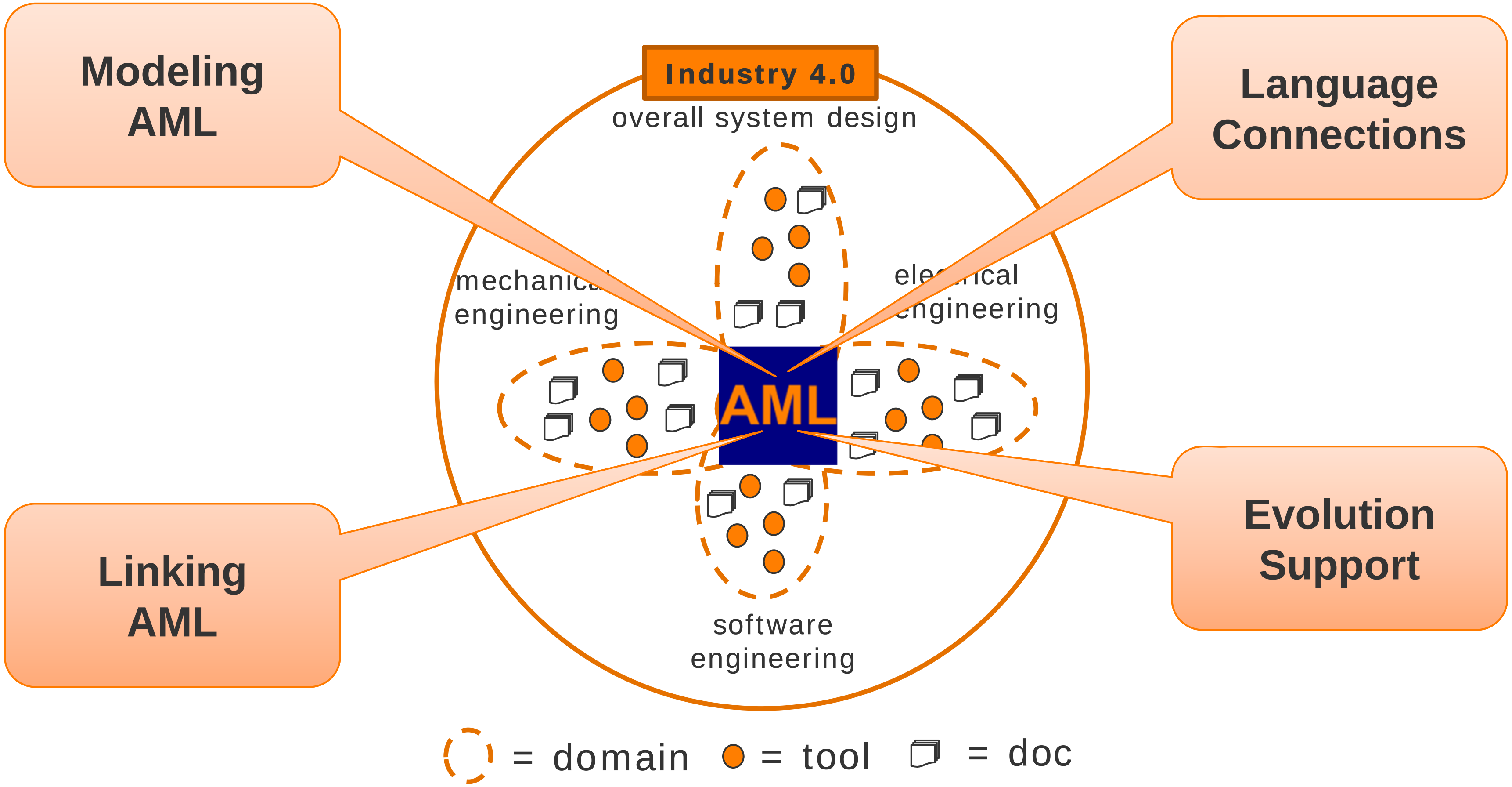
- **AutomationML (AML)**
- Emerging **standard** for tool data exchange
- Foundation for harmonizing engineering data coming from an heterogeneous tool network by means of a **unified format** and **data model**





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

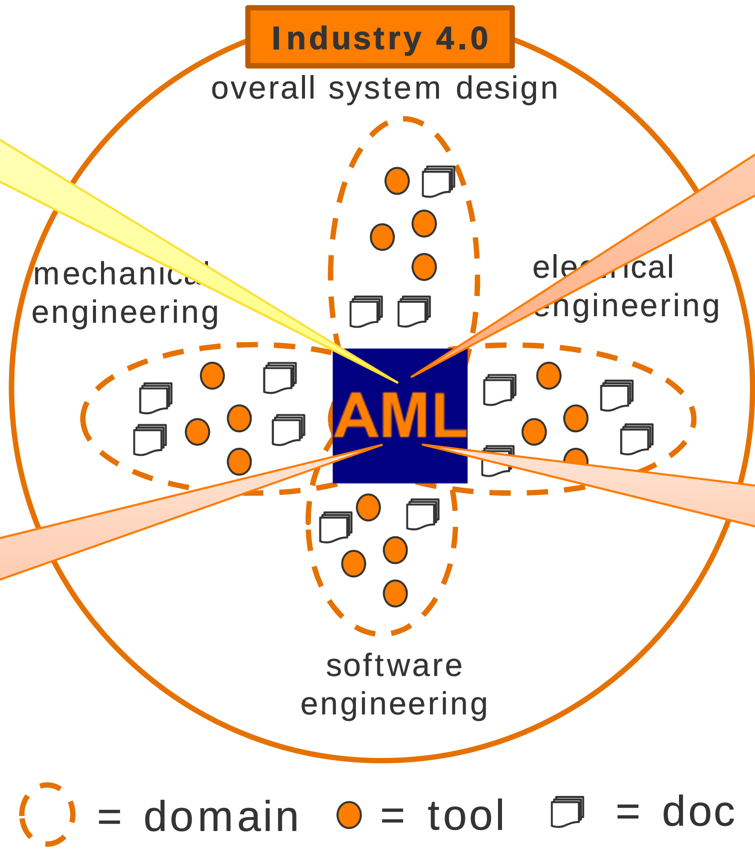
Our AML Research Topics



Our AML Research Topics

Modeling AML

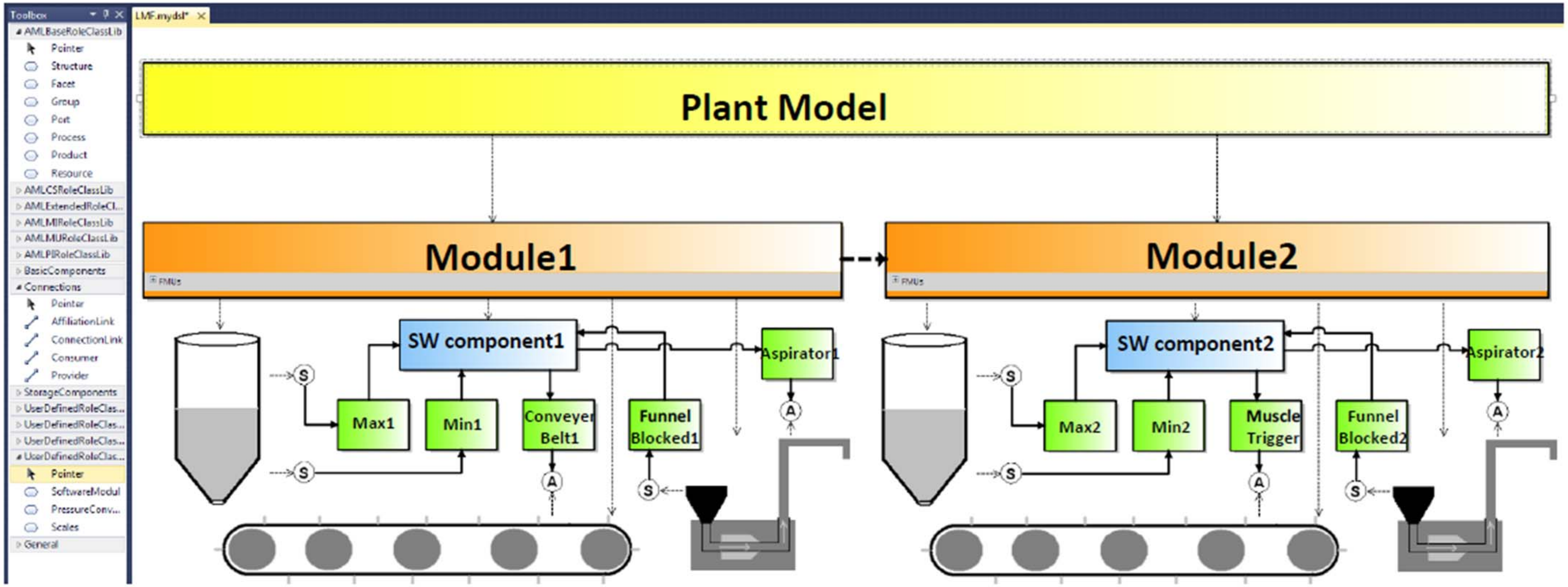
Language Connections



Linking AML

Evolution Support

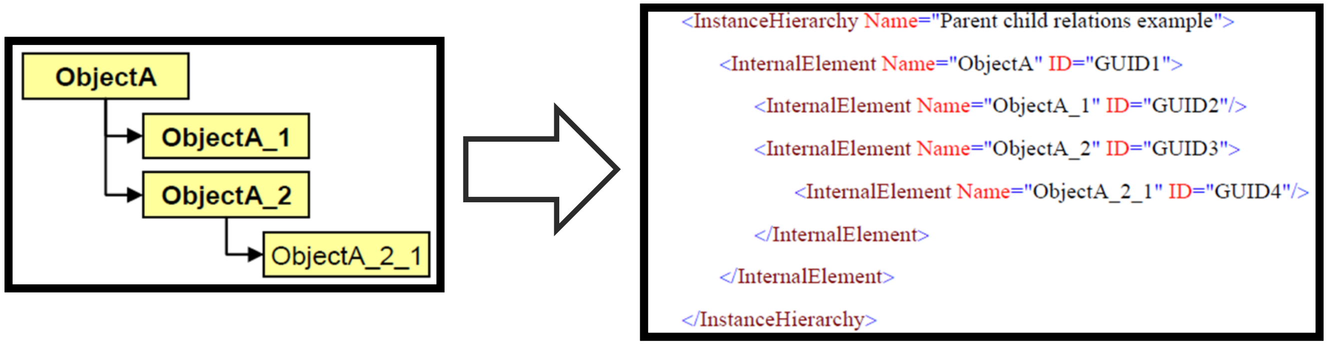
AutomationML =
Automation (Markup | Modeling) Language?



- 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.

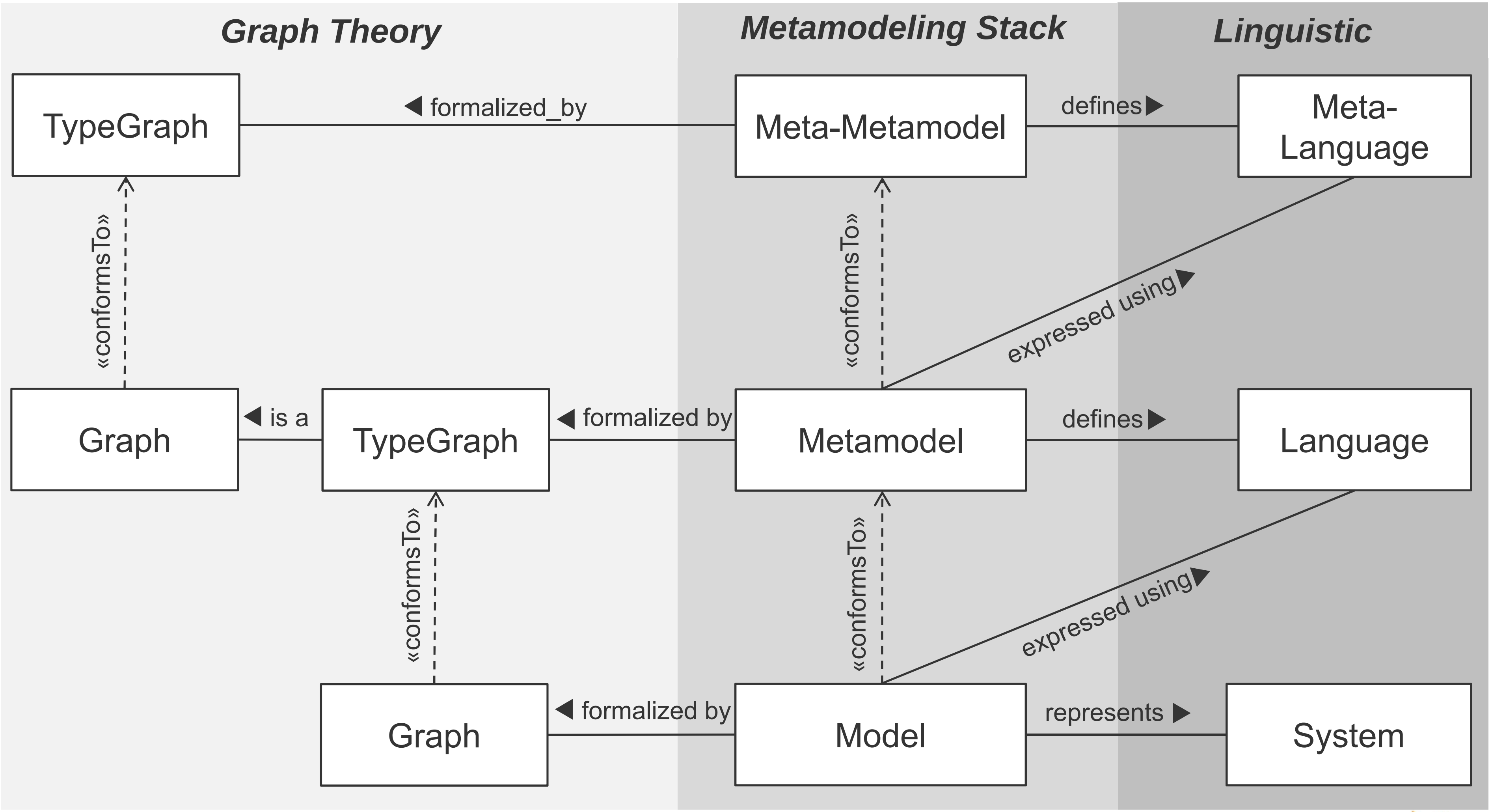
AutomationML = Automation (Markup | Modeling) Language?

- **Object-Oriented Format**
  - *Automation object: physical or logical entity in the automated system*
- **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.*



From Tree-based to Graph-based Representations

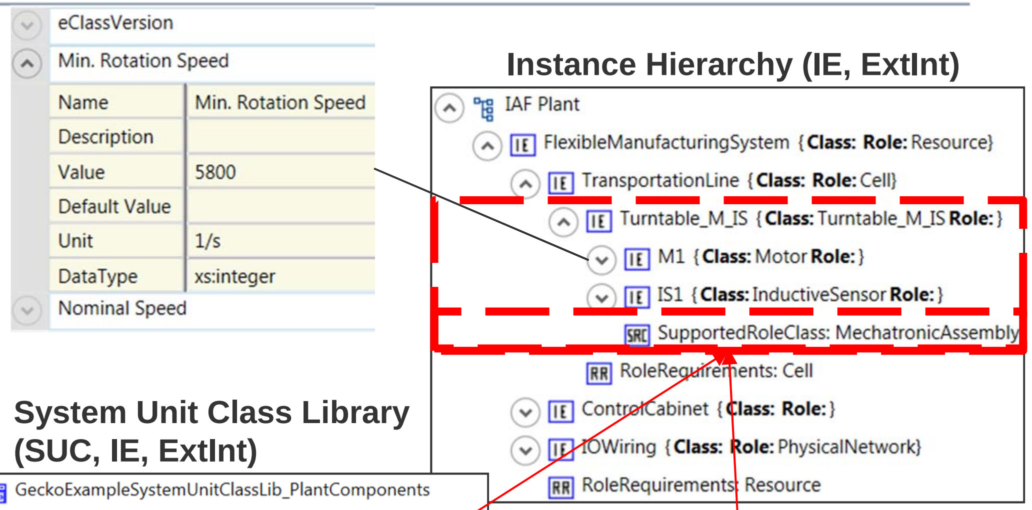
Language Engineering via Metamodeling

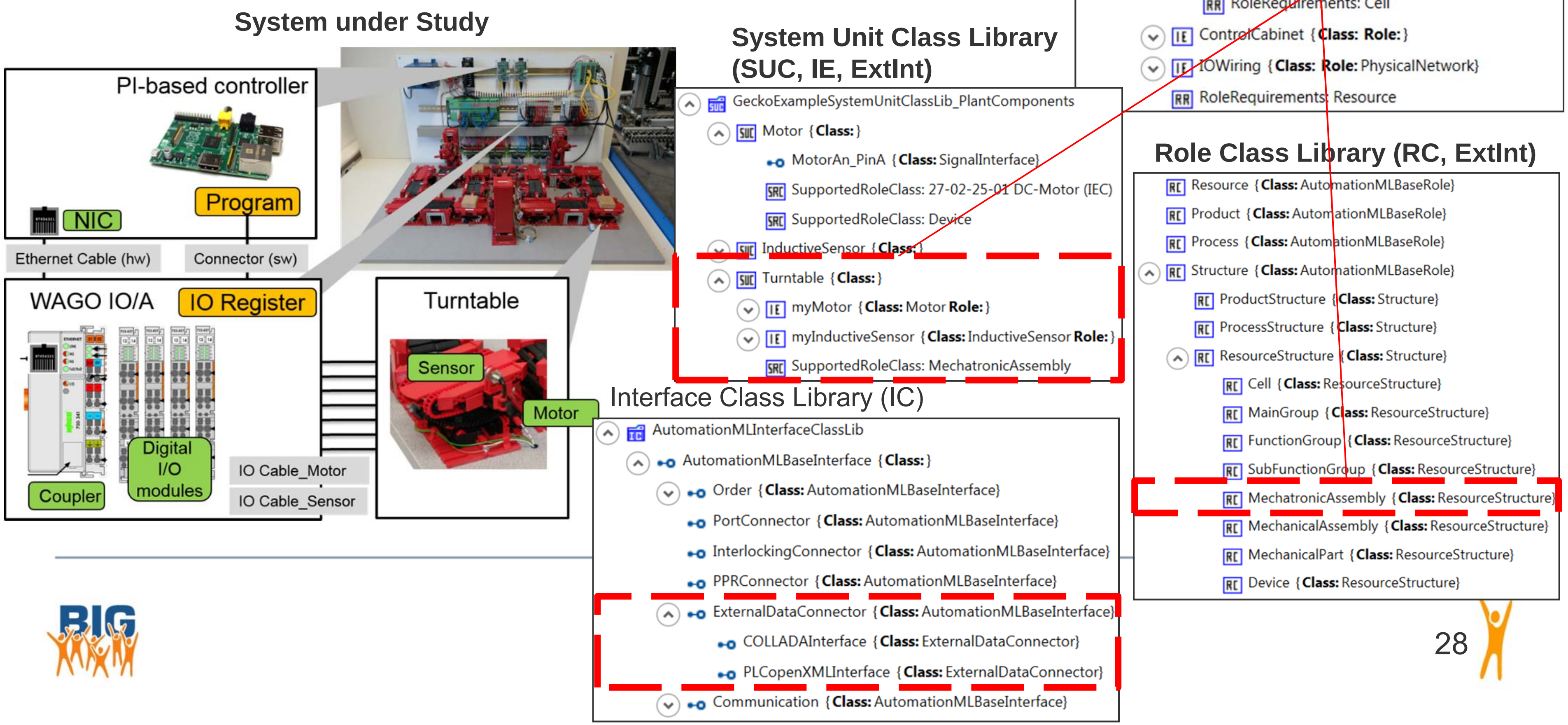


- T. Kühne: ***Matters of (Meta-)Modeling***. Software and System Modeling 5(4), pages 369-385, 2006.
- G. Engels, C. Lewerentz, M. Nagl, W. Schäfer, A. Schürr: ***Building Integrated Software Development Environments Part I: Tool Specification***. ACM Trans. Softw. Eng. Methodol. 1(2):135-167, 1992.

AutomationML by Example

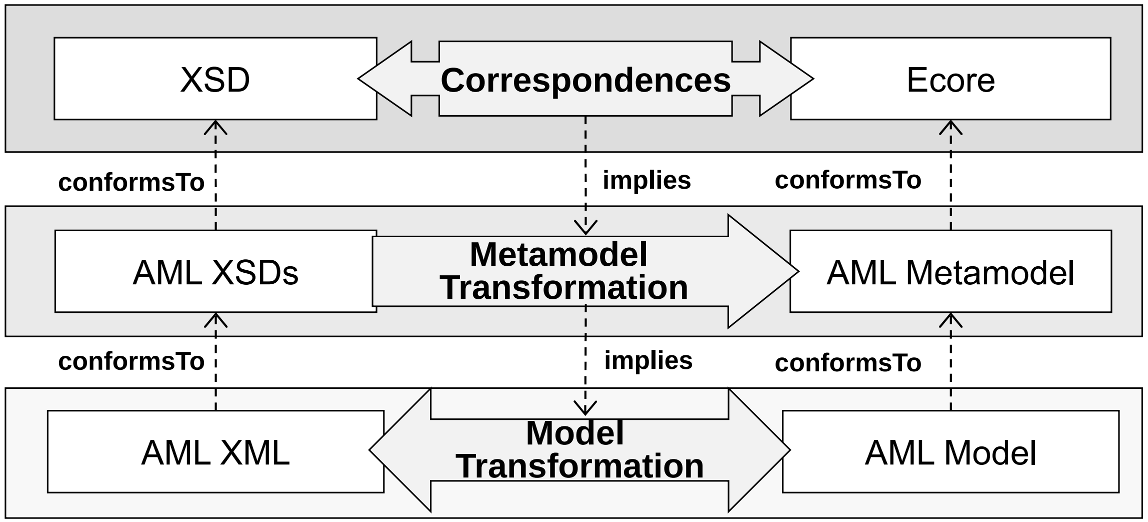
- Equipment Center for Distributed Systems, Institute of Ergonomics, Manufacturing Systems and Automation at Otto-v.-Guericke University Magdeburg.





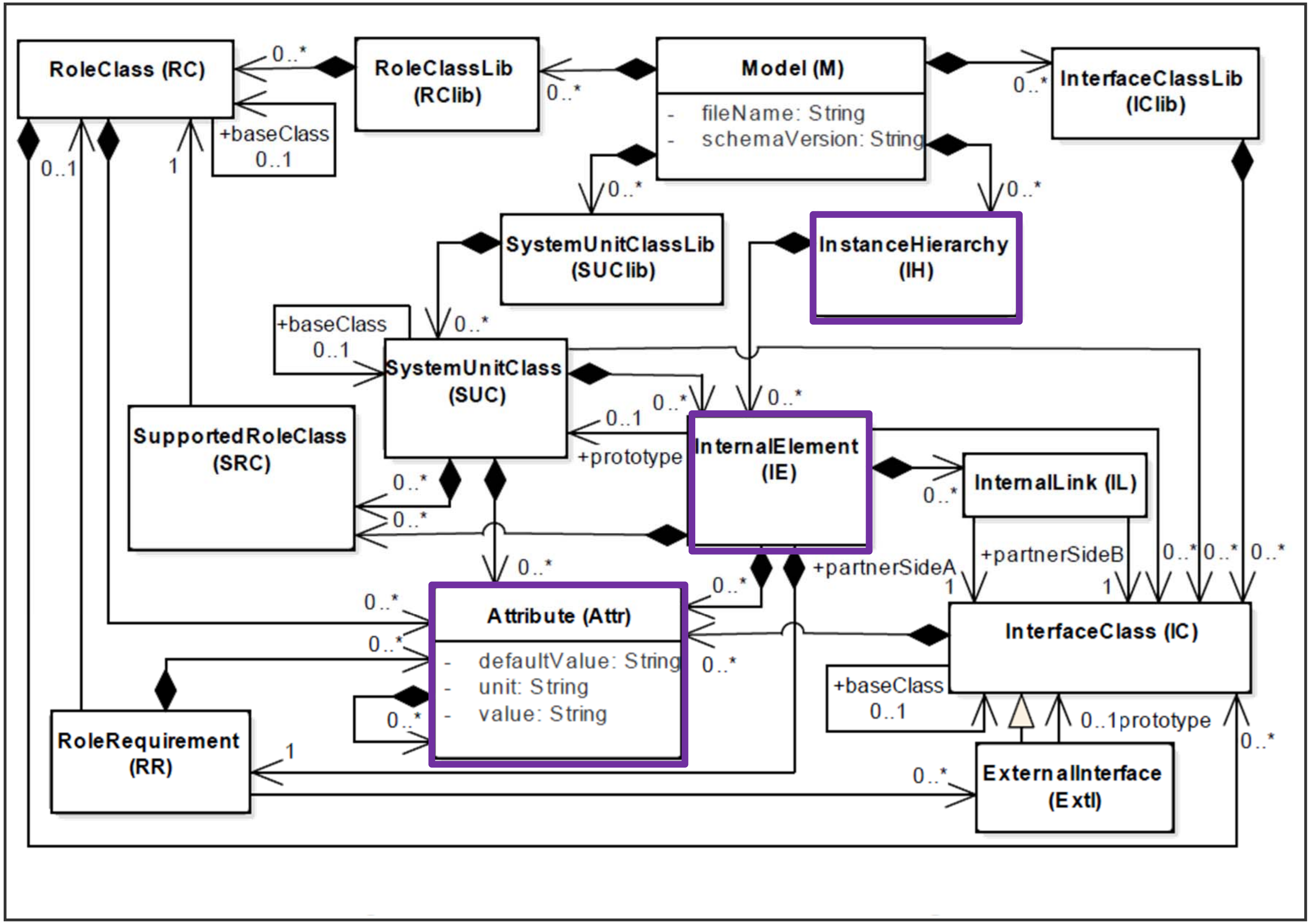
Metamodeling AutomationML

- **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



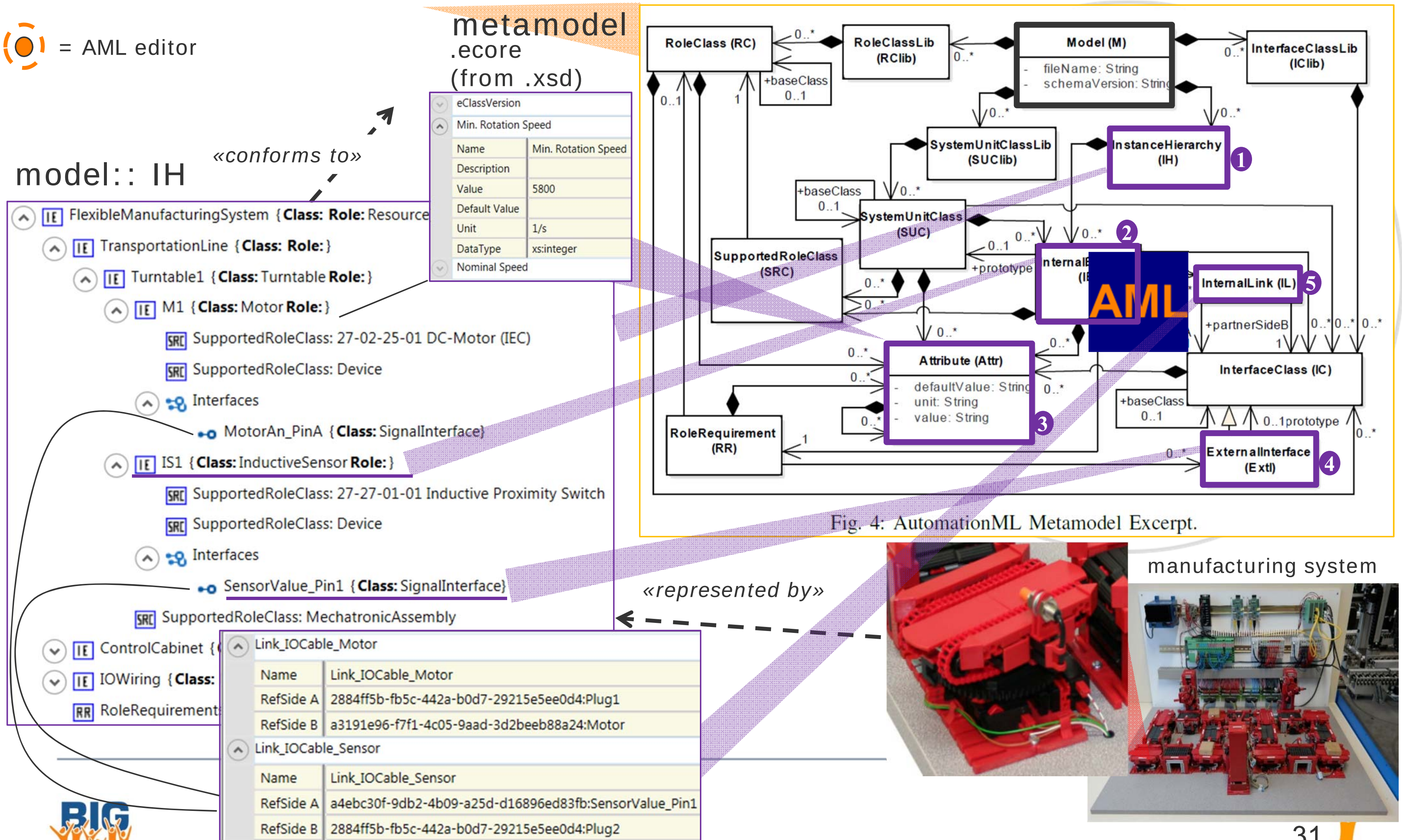
- A. Schauerhuber, M. Wimmer, E. Kapsammer, W. Schwinger, W. Retschitzegger: ***Bridging WebML to Model-Driven Engineering: From DTDs to MOF***. IET Software 1(3), 2007.

AutomationML Metamodel Exceprt



- S. Biffl, A. Lüder, E. Mätzler, N. Schmidt, M. Wimmer: ***Linking and Versioning Support for AutomationML: A Model-Driven Engineering Perspective***, accepted for INDIN, 2015, pp. 1–8.

AutomationML (AML)



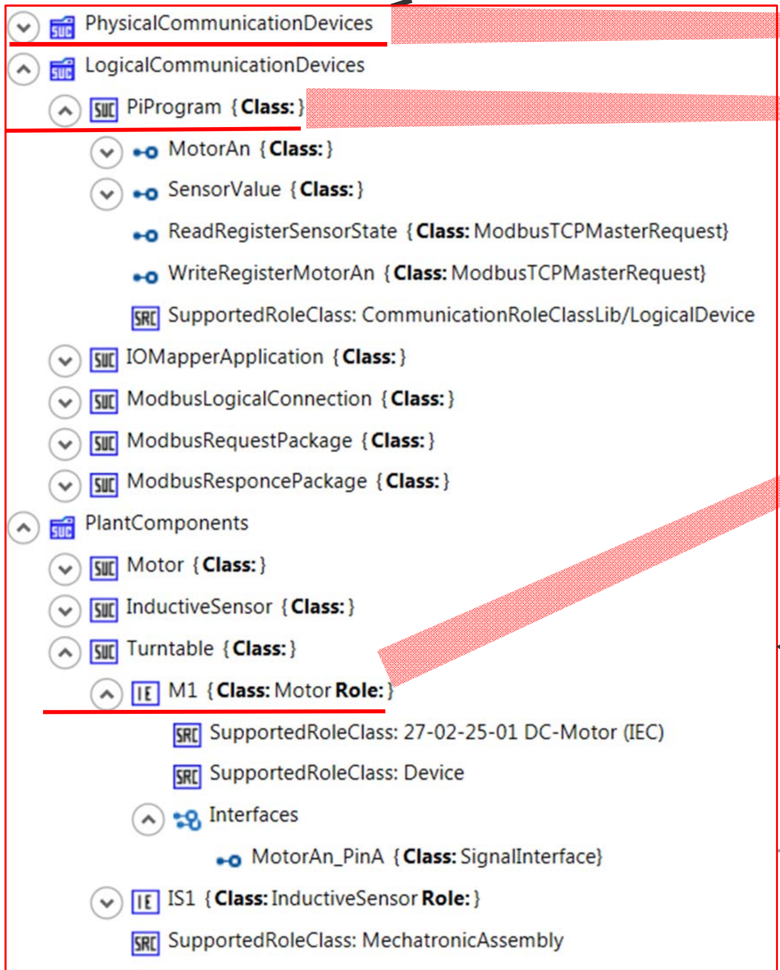
AutomationML (AML)

= AML editor

AML metamodel .ecore

model:: SUClib

«conforms to»



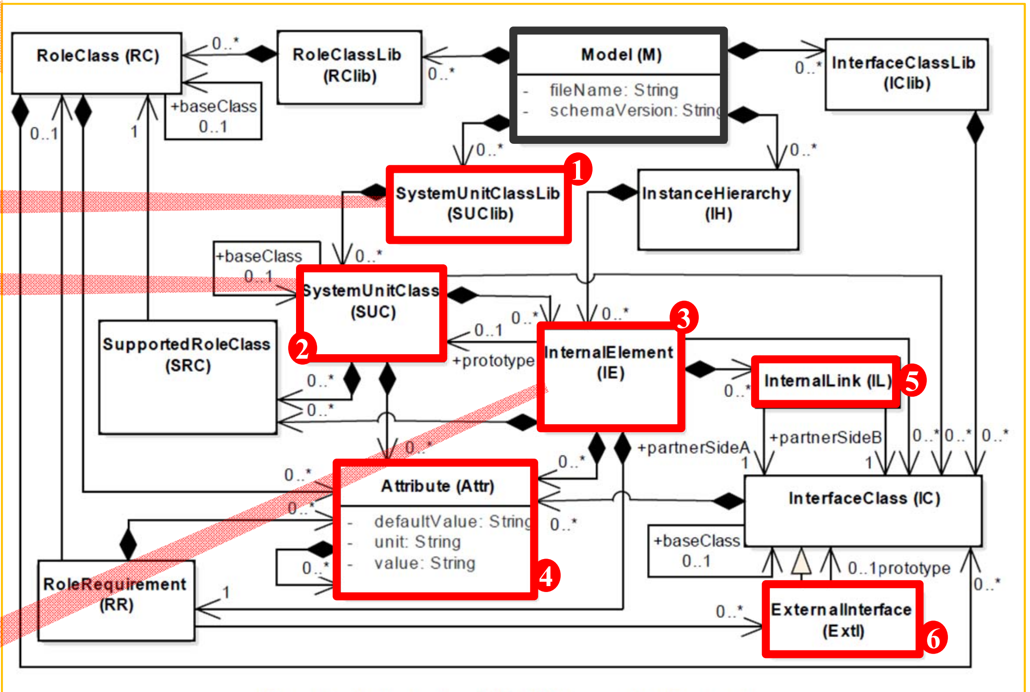
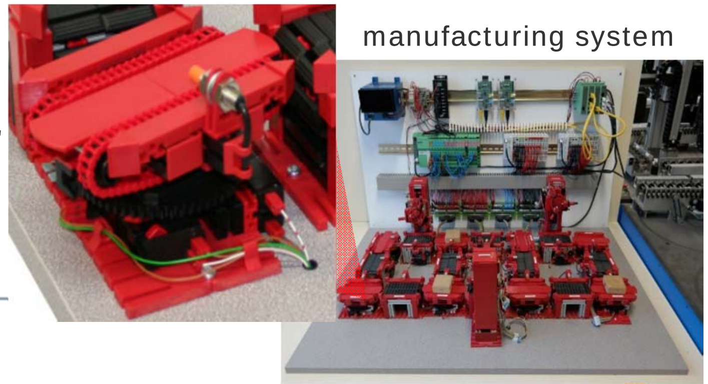


Fig. 4: AutomationML Metamodel Excerpt.

«represented by»

manufacturing system

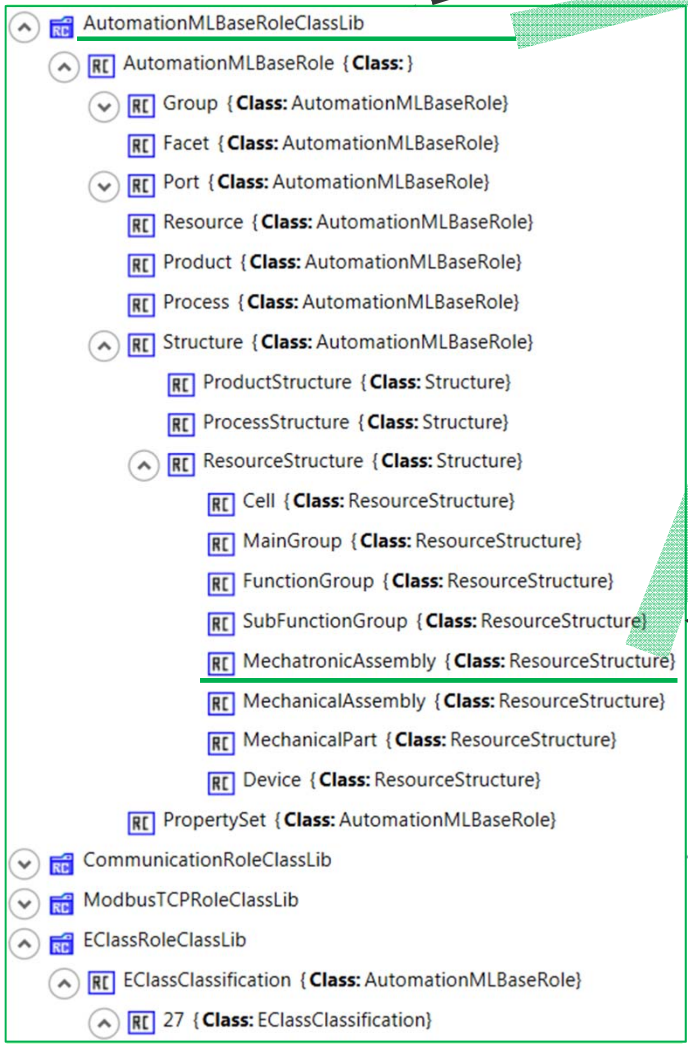


AutomationML (AML)

= AML editor

metamodel .ecore

model:: RClib



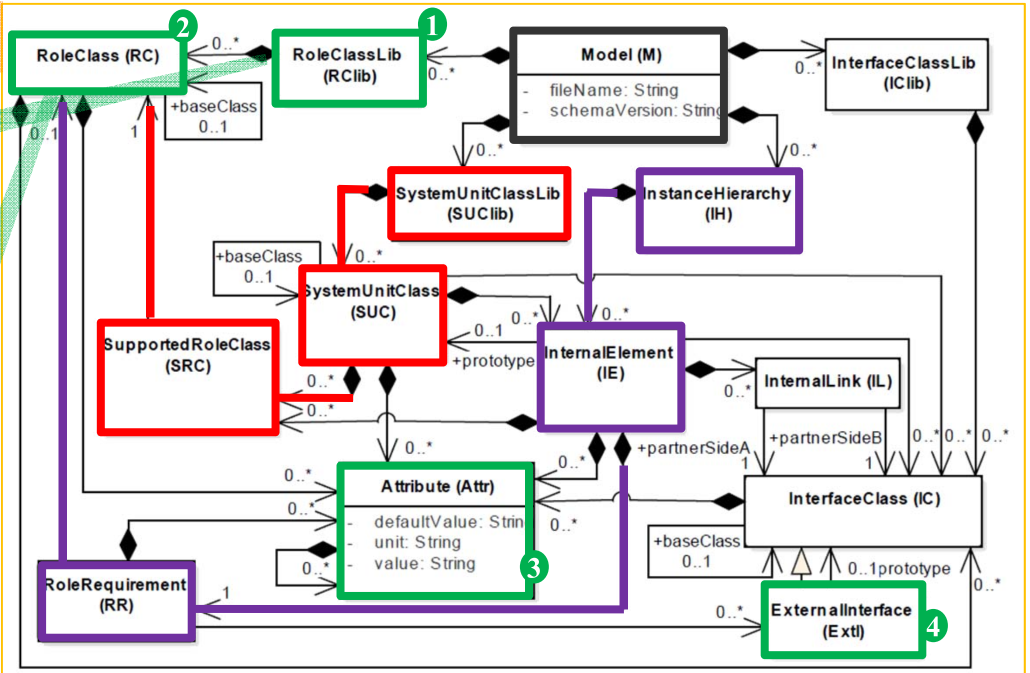
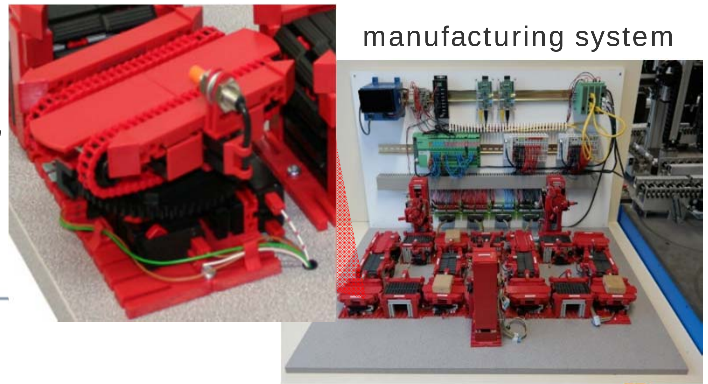


Fig. 4: AutomationML Metamodel Excerpt.

«represented by»

manufacturing system



AutomationML (AML)

= AML editor

metamodel
.ecore

model:: IClib

- CommunicationInterfaceClassLib
  - PhysicalEndPoint {**Class:** Communication}
  - LogicalEndPoint {**Class:** Communication}
  - DatagrammObject {**Class:** Communication}
- ModbusTCPInterfaceClassLib
- AutomationMLInterfaceClassLib
  - AutomationMLBaseInterface {**Class:**}
    - Order {**Class:** AutomationMLBaseInterface}
    - PortConnector {**Class:** AutomationMLBaseInterface}
    - InterlockingConnector {**Class:** AutomationMLBaseInterface}
    - PPRConnector {**Class:** AutomationMLBaseInterface}
    - ExternalDataConnector {**Class:** AutomationMLBaseInterface}
      - COLLADAInterface {**Class:** ExternalDataConnector}
      - PLCopenXMLInterface {**Class:** ExternalDataConnector}
    - Communication {**Class:** AutomationMLBaseInterface}
      - SignalInterface {**Class:** Communication}

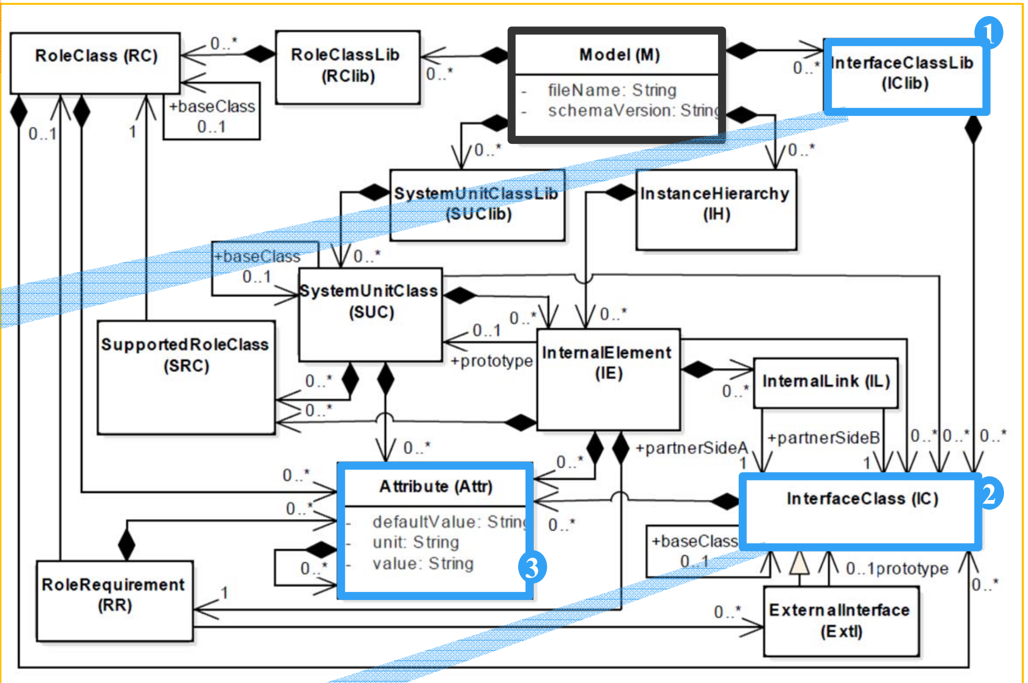
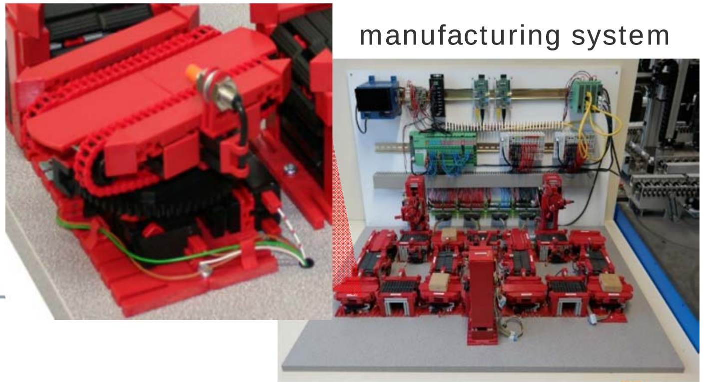


Fig. 4: AutomationML Metamodel Excerpt.

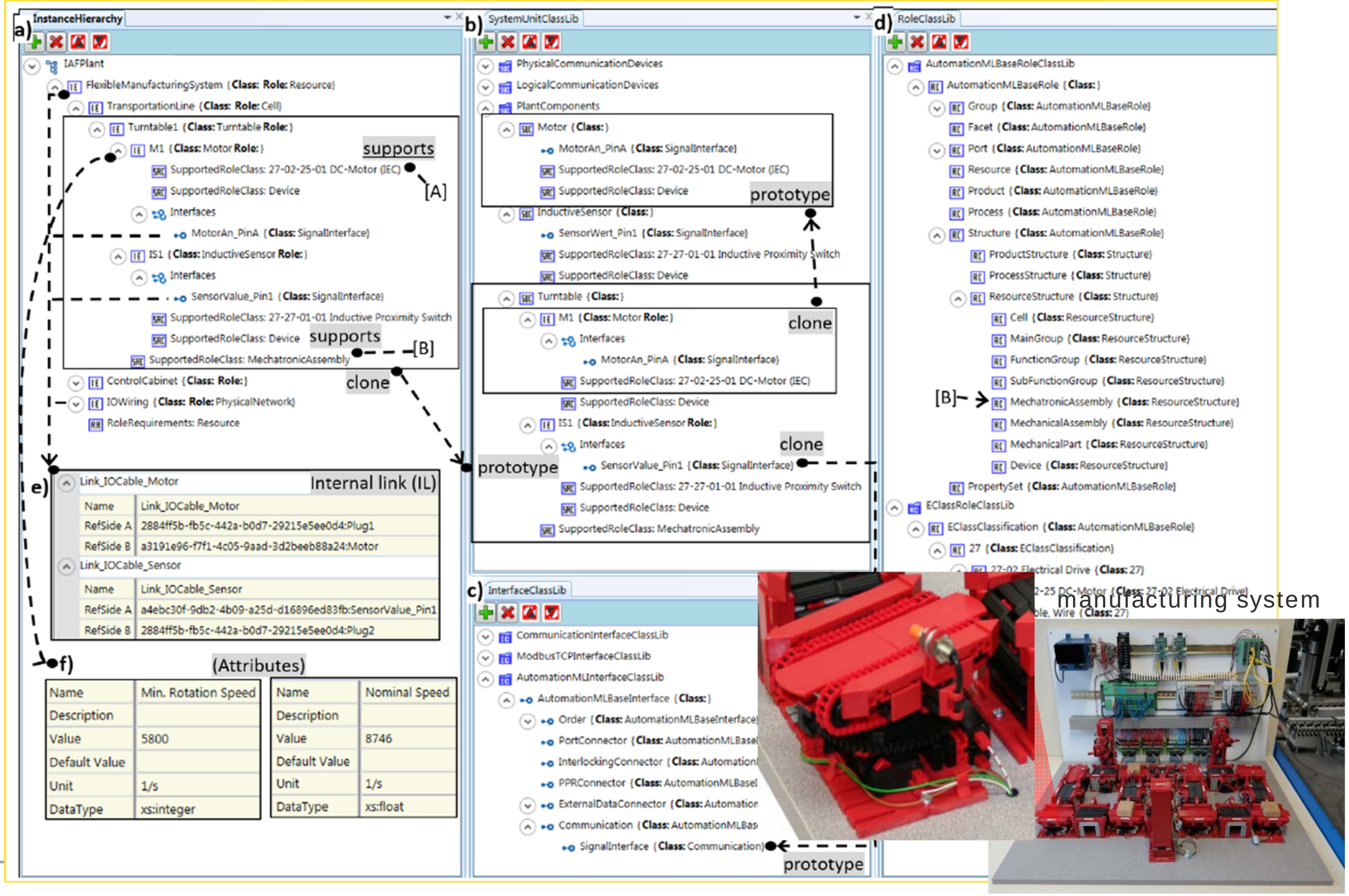
«represented by»

manufacturing system

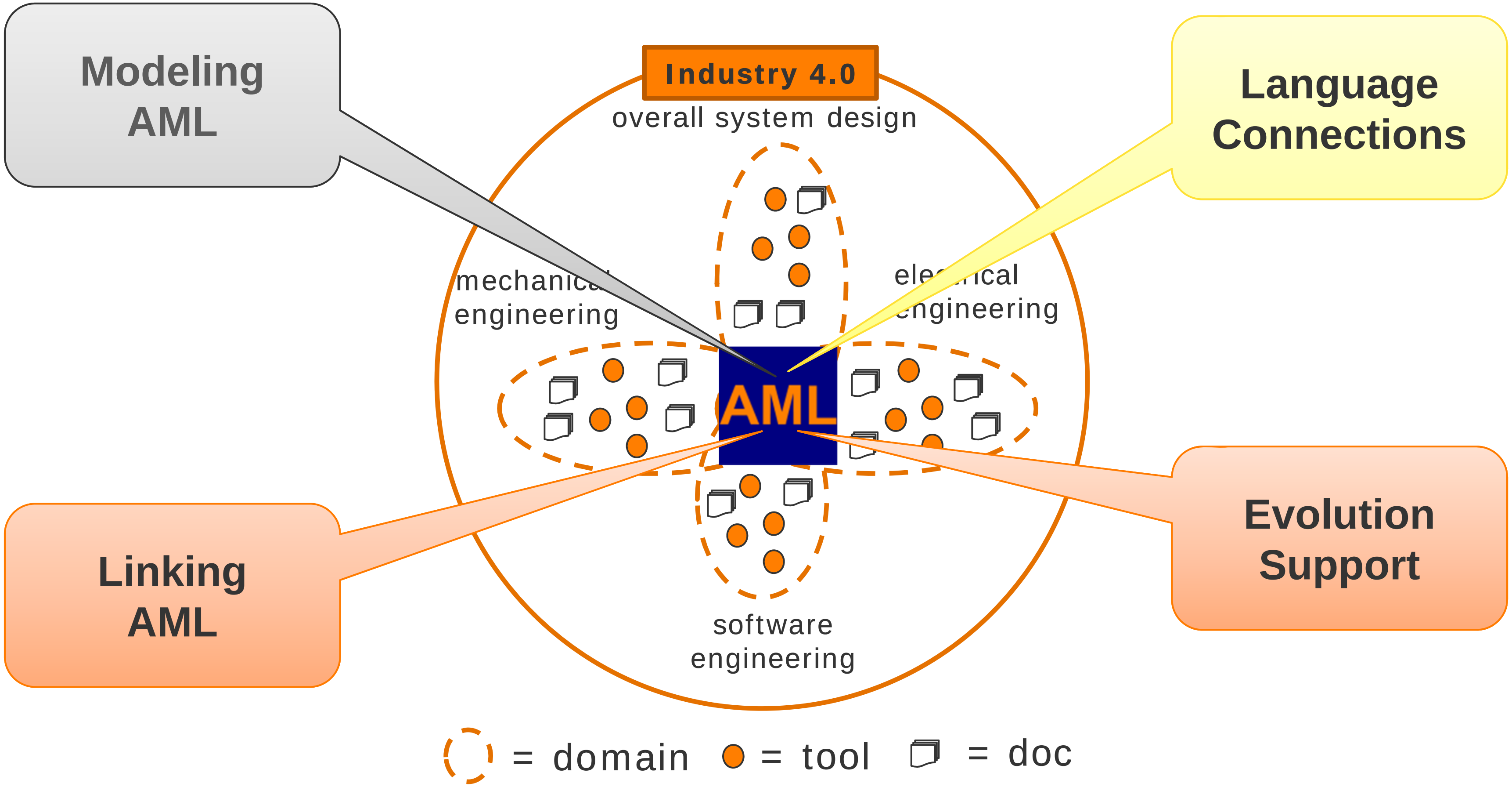


Flexible Manufacturing System in AML Editor

= AML editor

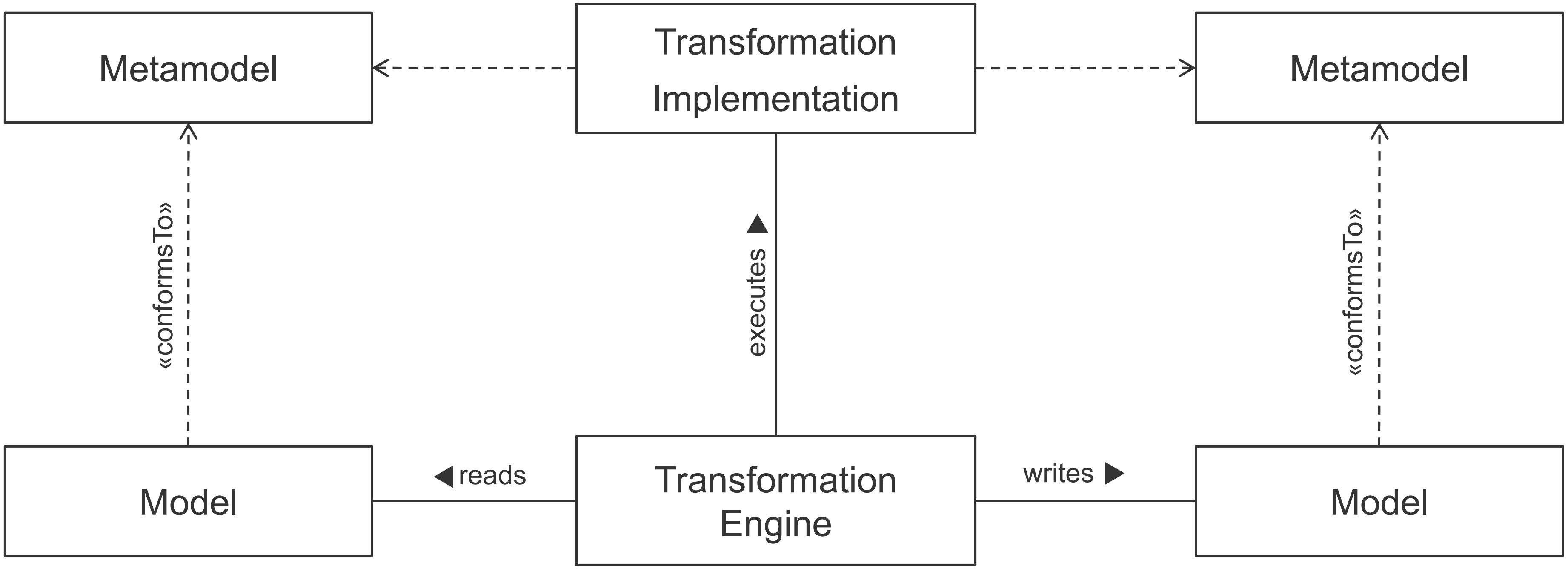


Our AML Research Topics



Further Benefits of Explicit Models

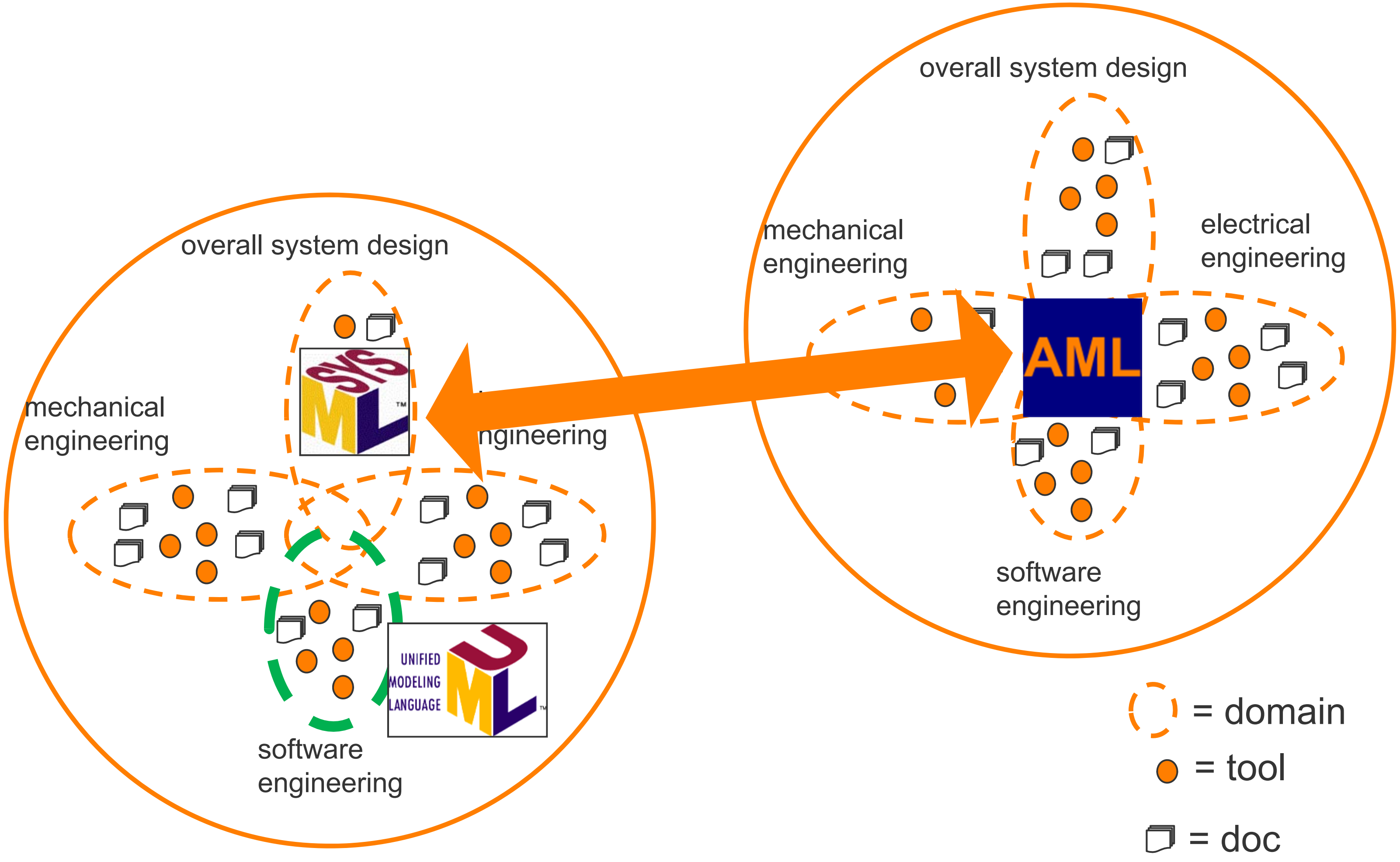
Model Transformation Pattern and Supporting Tools



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

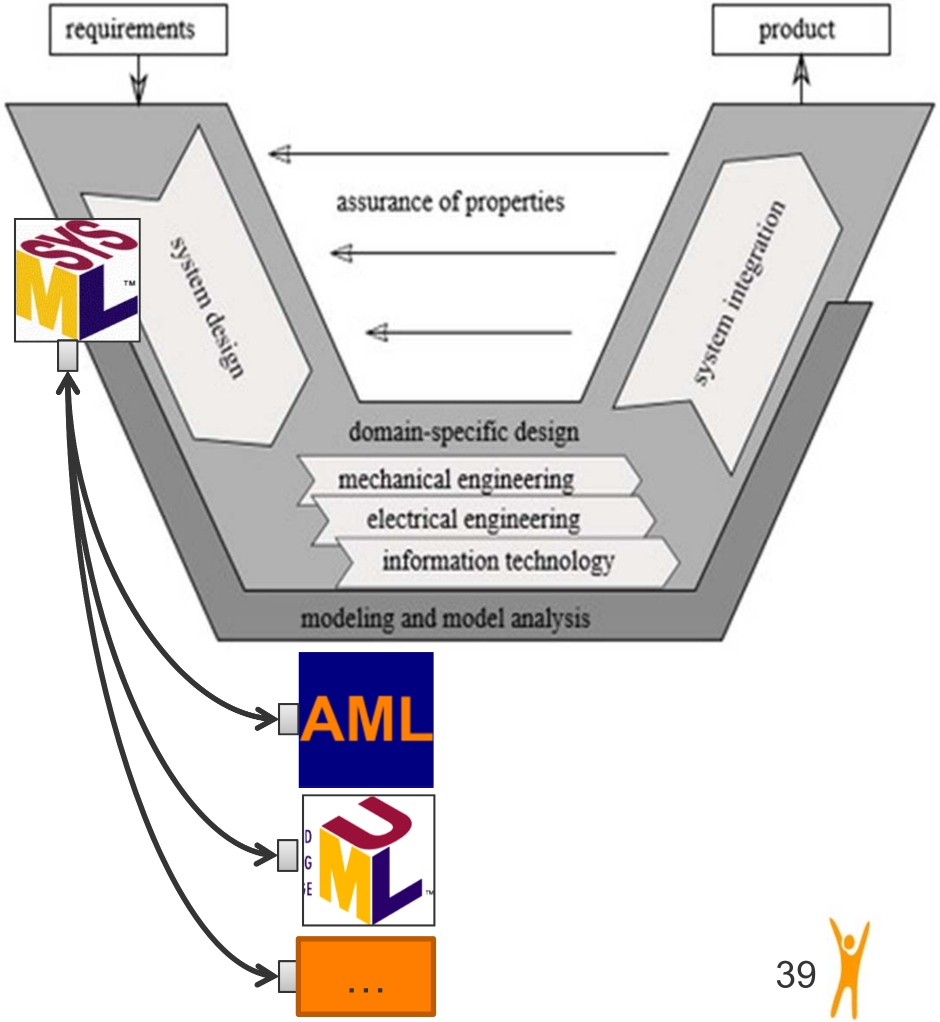
Transformation Scenario Investigated

AML and SysML: Two Unrelated Modeling Standards



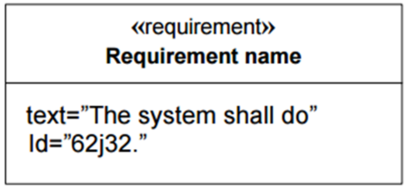
SysML in a Nutshell (1/2)

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

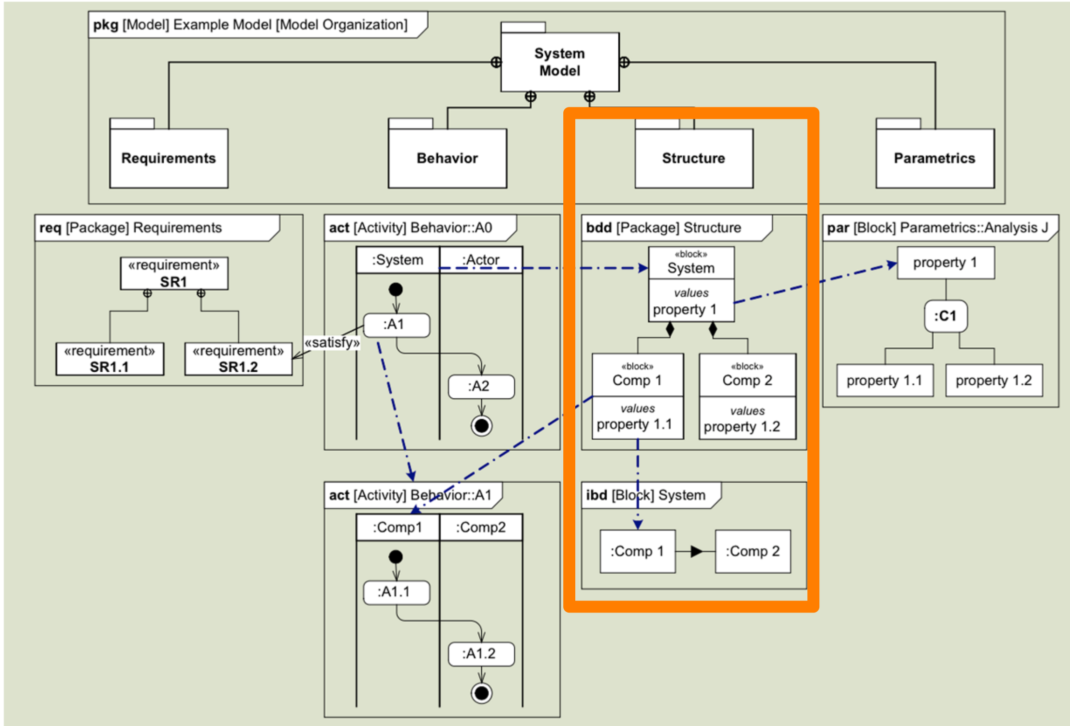


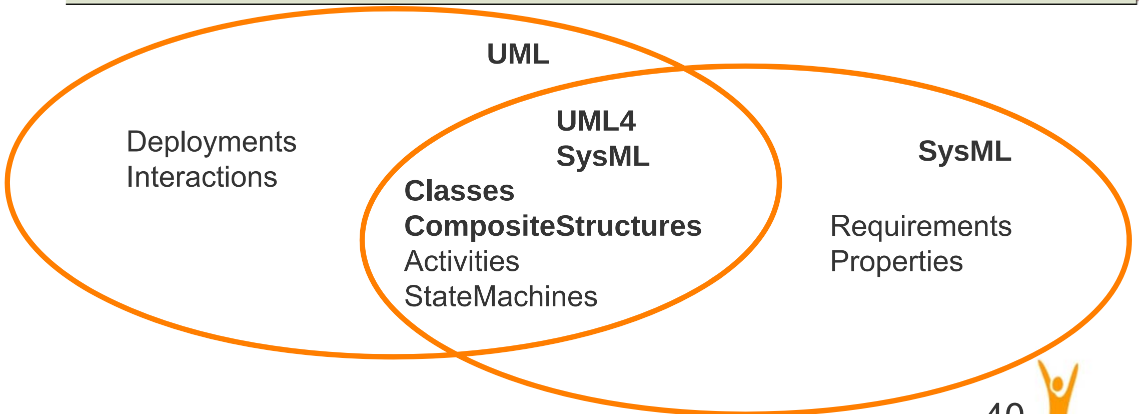
SysML in a Nutshell (2/2)

- **Additions to UML** for Requirements and Properties
  - **Requirement**: SysML provides modeling constructs to represent text-based requirements and relate them to other modeling elements.



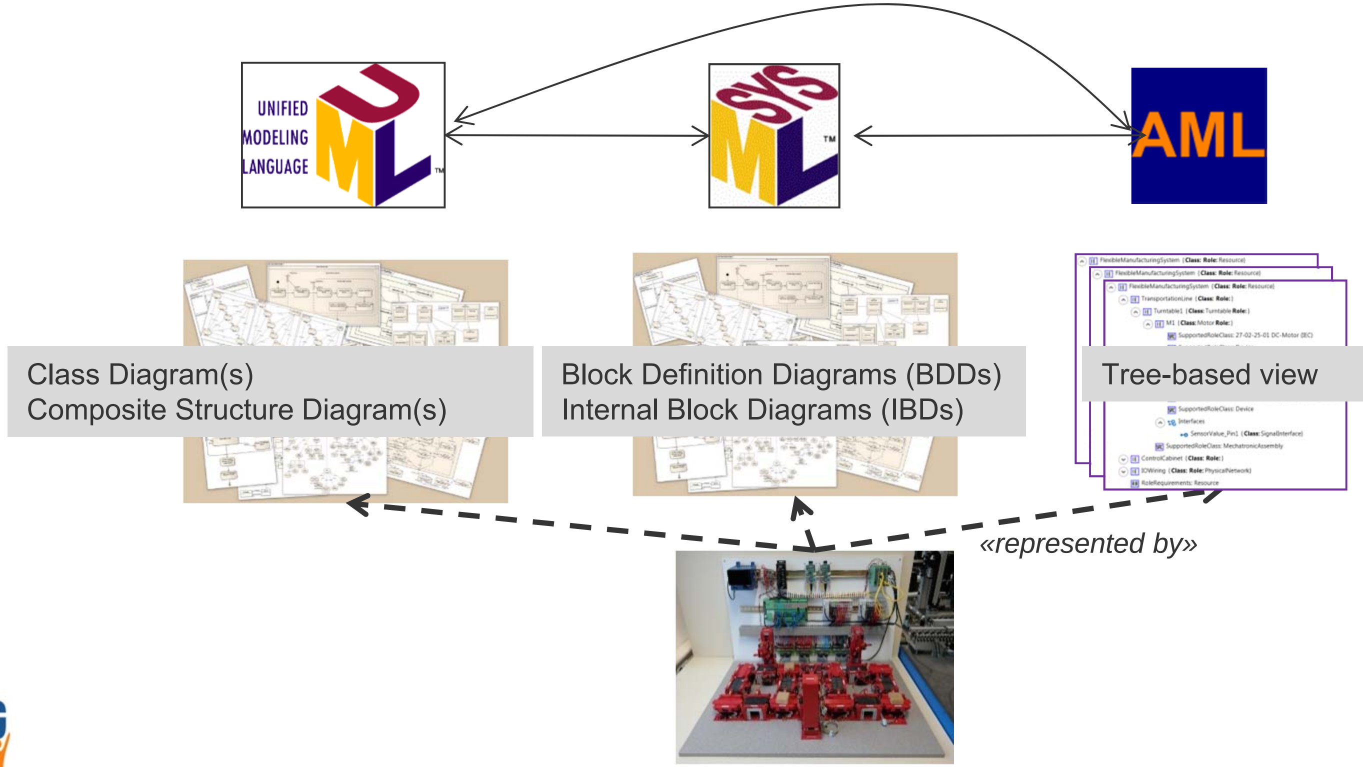
  - Constraints and Parametric Diagram (constraint analysis)
- **Customization of UML** for structural modeling through Classes and Composite Structures
  - **Block** derives from CompositeStructures::Class





From AutomationML to SysML and Back Again

- **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 through language definition)



Comparison of AML and SysML

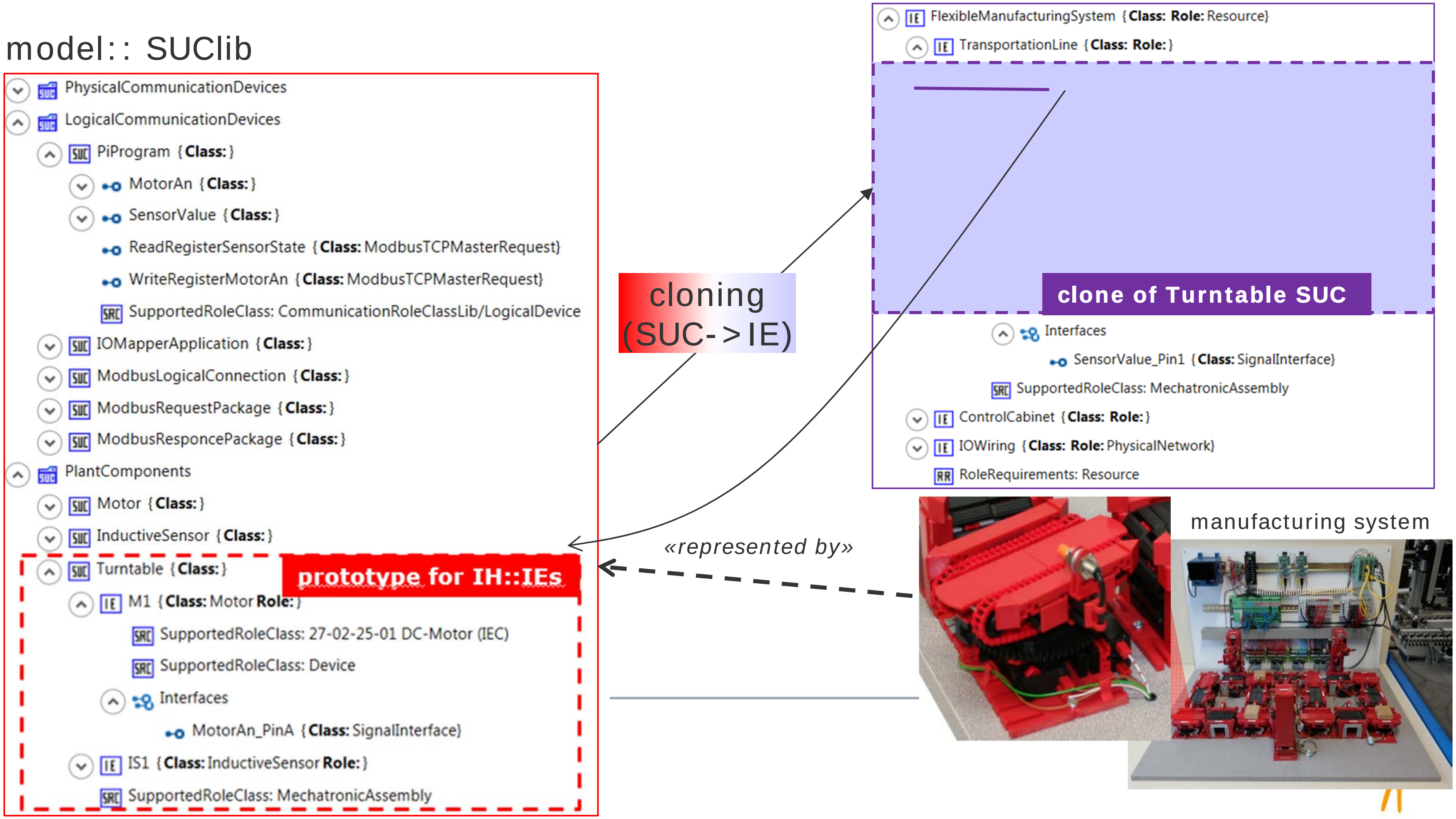
1. AML: Data exchange format vs. SysML: language for systems modeling
   - **AML** serves as a **standardized exchange format** between the diverse discipline-specific tools involved in the development of automation systems.
   - **SysML** is designed as a **language for systems modeling**, i.e., representing the design of a system that builds the basis for planning, implementing, and analyzing it.

2. AML: Tree-based editing vs. SysML: diagram-based editing
   - **Benefits of graphical representation:** visualizing the architecture of a system and the power of building multiple views on a complex system

Comparison of AML and SysML

3. AML: Prototype-based vs. SysML: Class-based

model:: IH



Comparison of AML and SysML

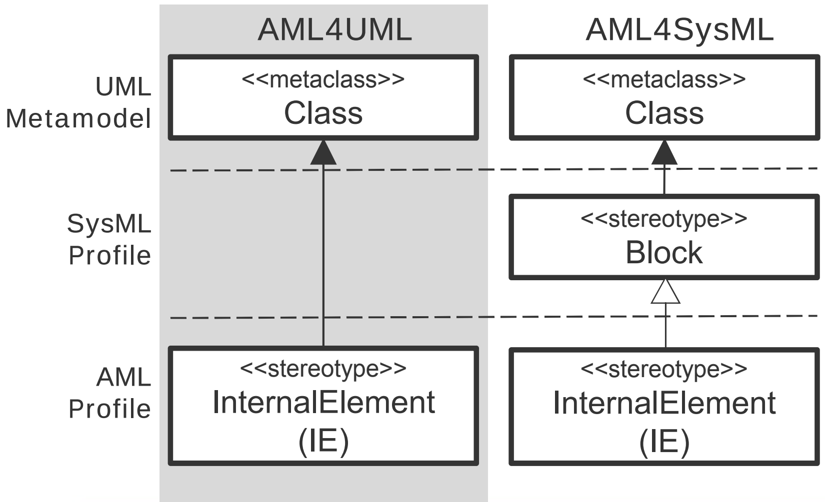
4. Extensibility Mechanisms

- **AML**. **RClibs** can be used for introducing new concepts
  - **Role.** A role is a class that describes an abstract functionality without defining the underlying technical implementation.
  - *A* ***Resource*** is an entity involved in production; they execute processes and handle products. Examples for resources are robots, conveyors or machines. Resources may be hardware components of a production system, but also software.
  - A ***Product*** depicts a produced good. Products are processed by resources.
  - A ***Process*** represents a production process including sub-processes, process parameters and the process chain.

- **SysML**: UML profiling mechanism

Modeling with AML4SysML

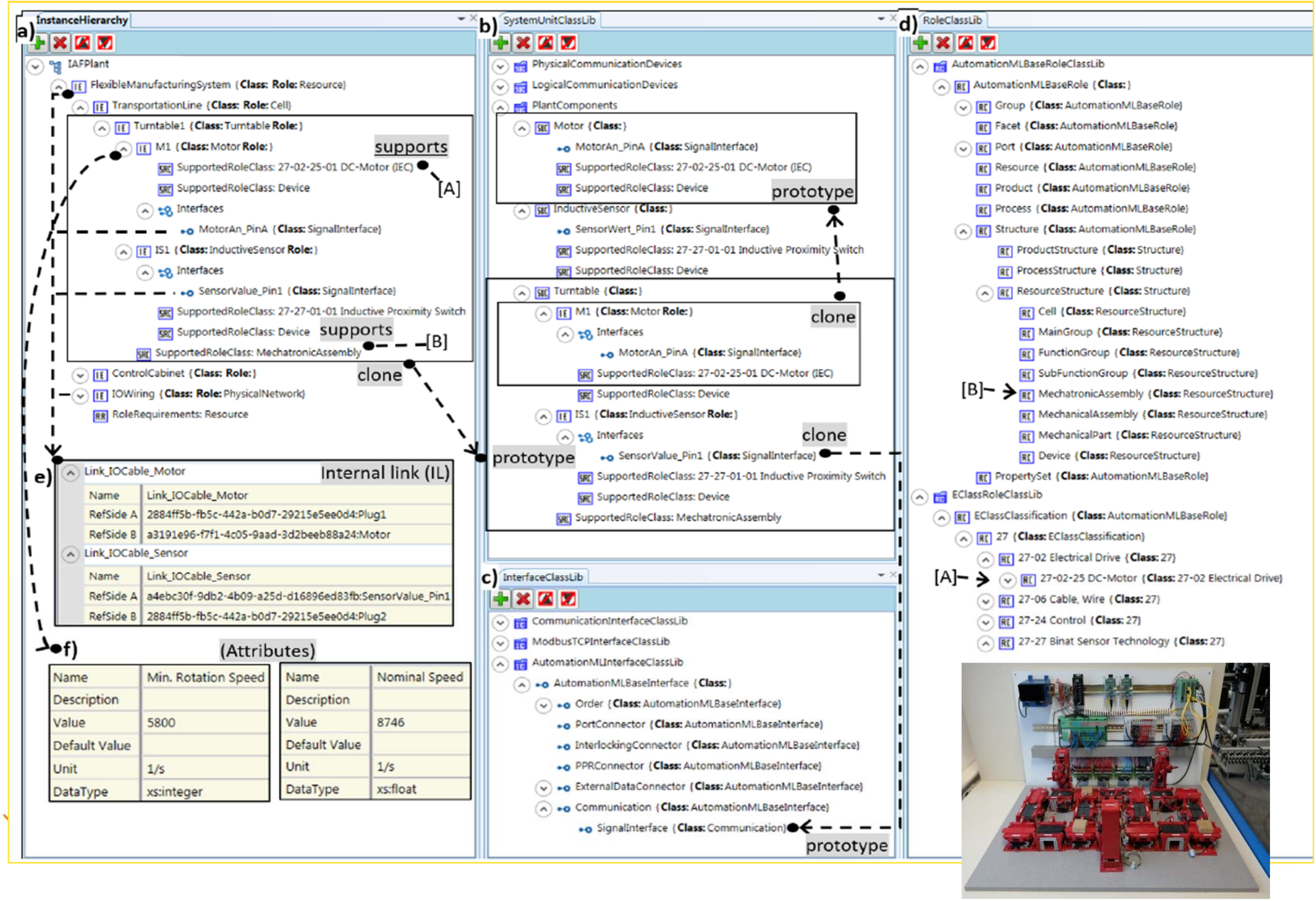
- SysML is an extension of UML
- We reuse and extend the common **UML SysML subset**



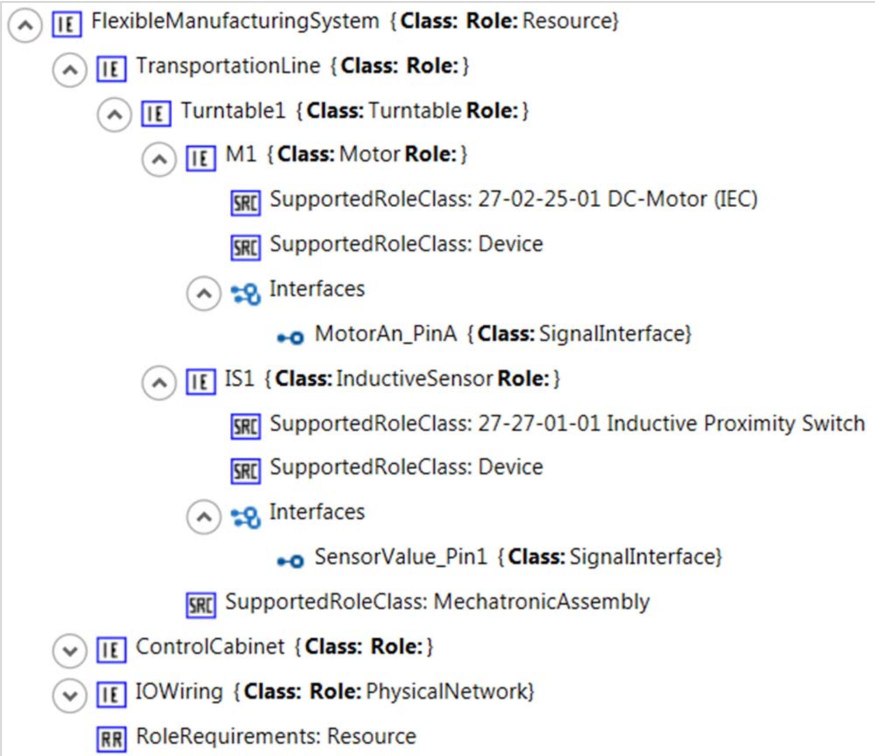
		AML Element	Stereotype	AML4UML Metaclass	AML4SysML Metaclass
		Model	M	Package, Model Library	Package, ModelLibrary
		Attribute	Attr	Property	Property
				DataType	DataType
Libraries		InterfaceClassLib	IClib	Package	Package, ModelLibrary
		RoleClassLib	RClib	Package	Package, ModelLibrary
		SystemUnitClassLib	SUClib	Package	Package, ModelLibrary
AML Classes		InterfaceClass	IC	Class	InterfaceBlock
		RoleClass	RC	Class	Block
		SystemUnitClass	SUC	Class	Block
AML Object		ExternalInterface	ExtI	Port	Port
		InstanceHierarchy	IH	Class	Block
		InternalElement	IE	Class	Block
AML Relationships	*Inheritance*	BaseClass	BaseClass	Generalization	Generalization
	Object Rel.	InternalLink	IL	Connector	Connector
	Object-Class Rel.	RoleRequirement	RR	Dependency	Dependency
				Class	Block
		SupportedRoleClass	SRC	Dependency	Dependency
		Prototype	Prototype	Dependency	Dependency

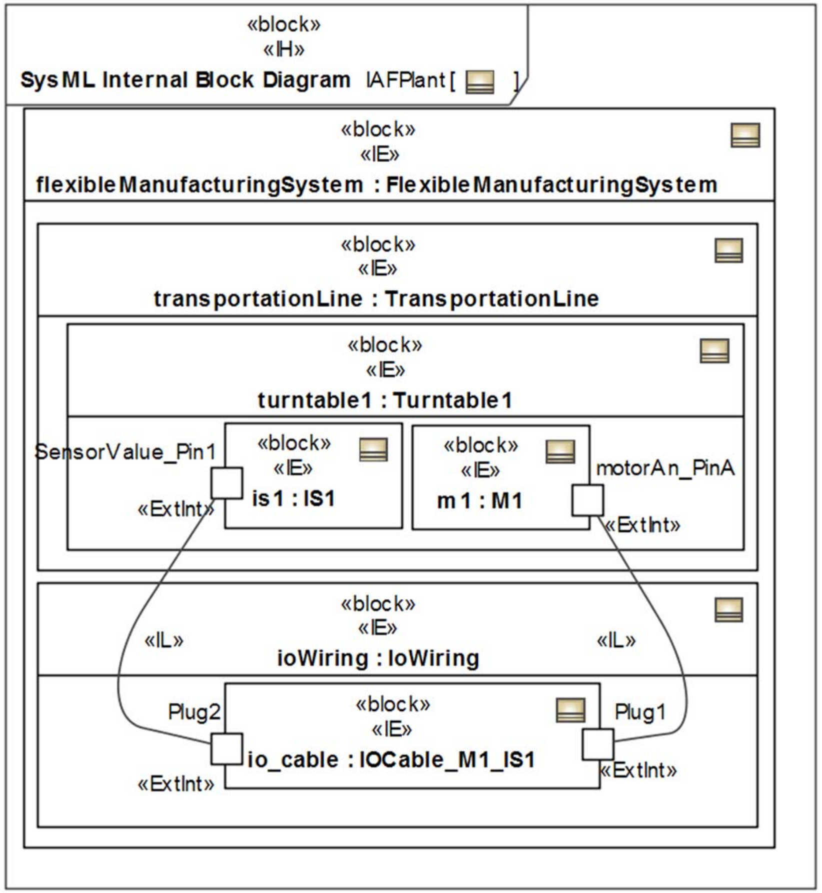
Flexible Manufacturing System in AML Editor

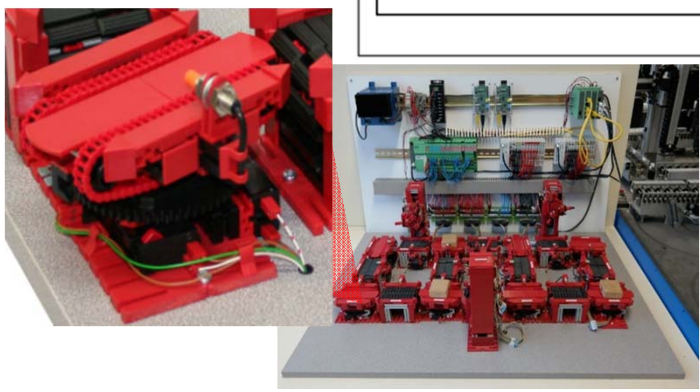
= AML editor



Flexible Manufacturing System in AML and SysML (excerpt)



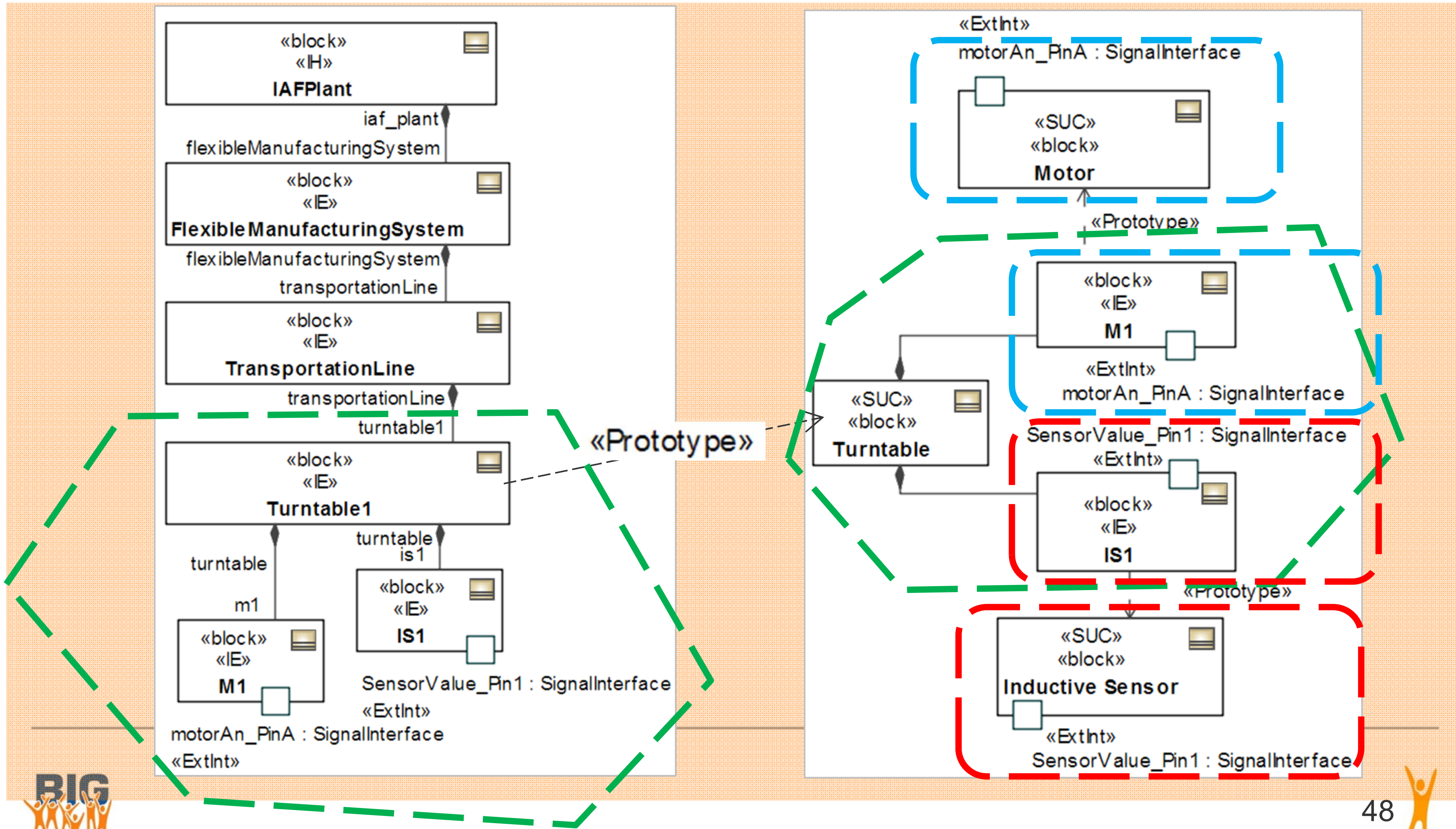




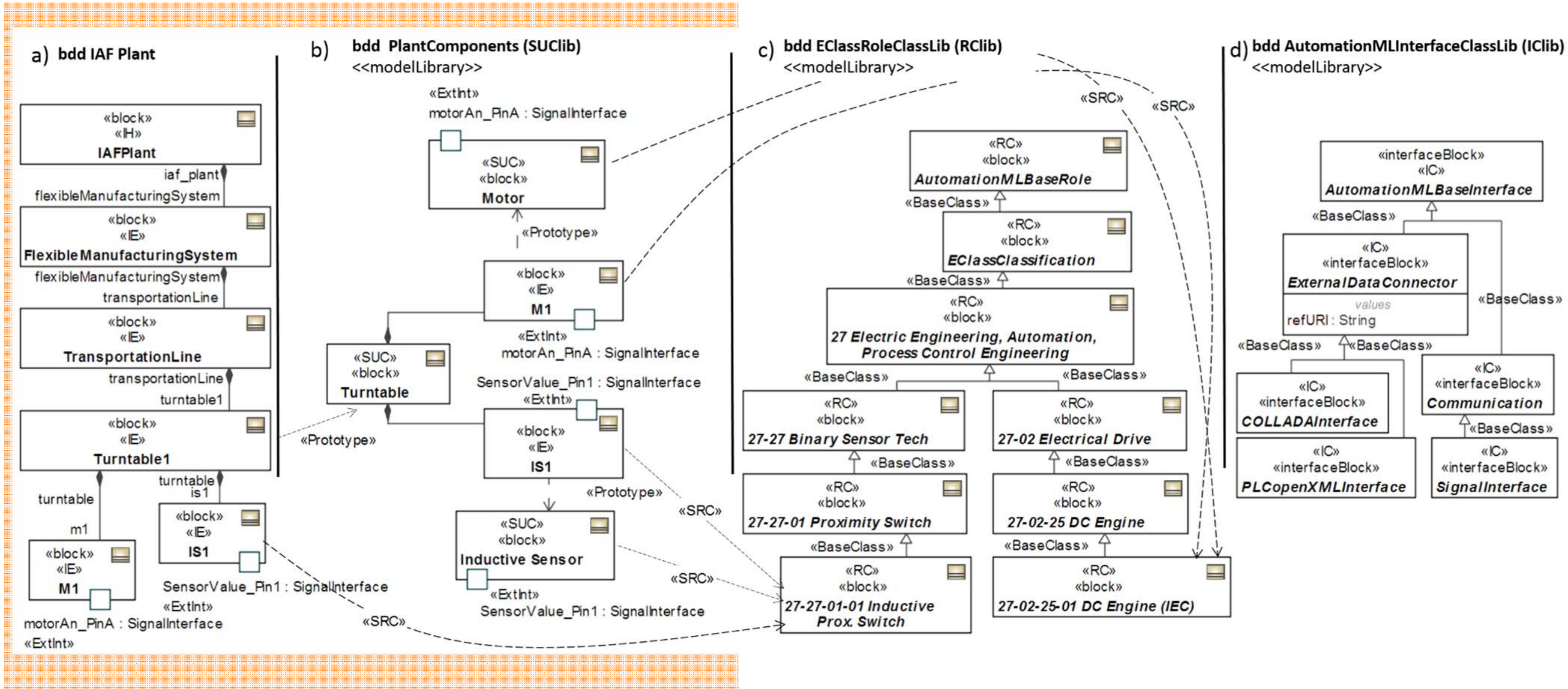
Flexible Manufacturing System in AML and SysML (excerpt)

bdd IAF Plant «IH»

bdd PlantComponents (SUClib) «model library»

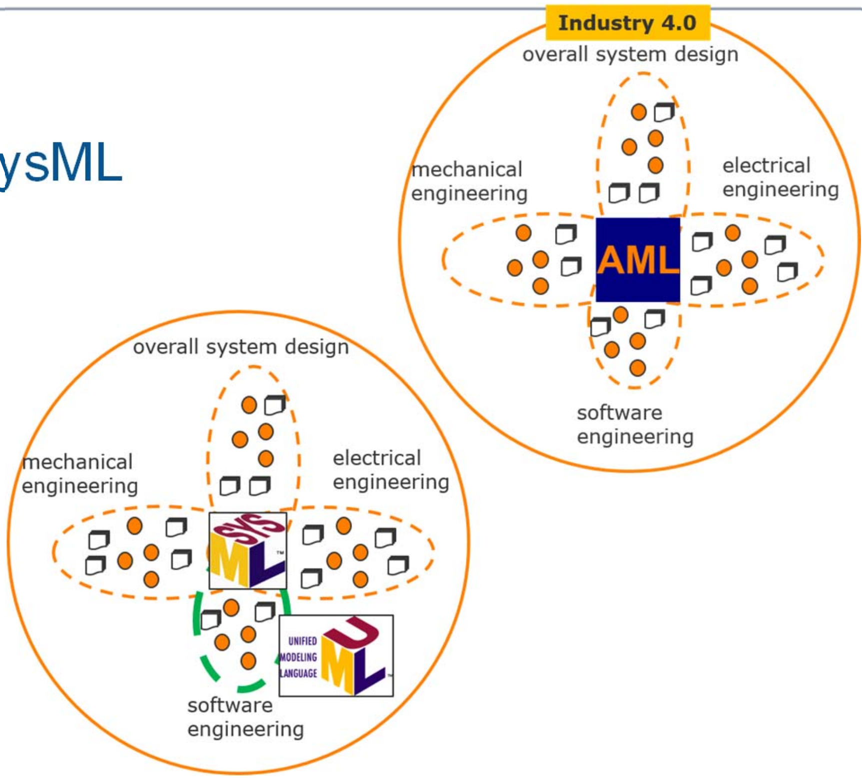


Flexible Manufacturing System in AML and SysML (excerpt)



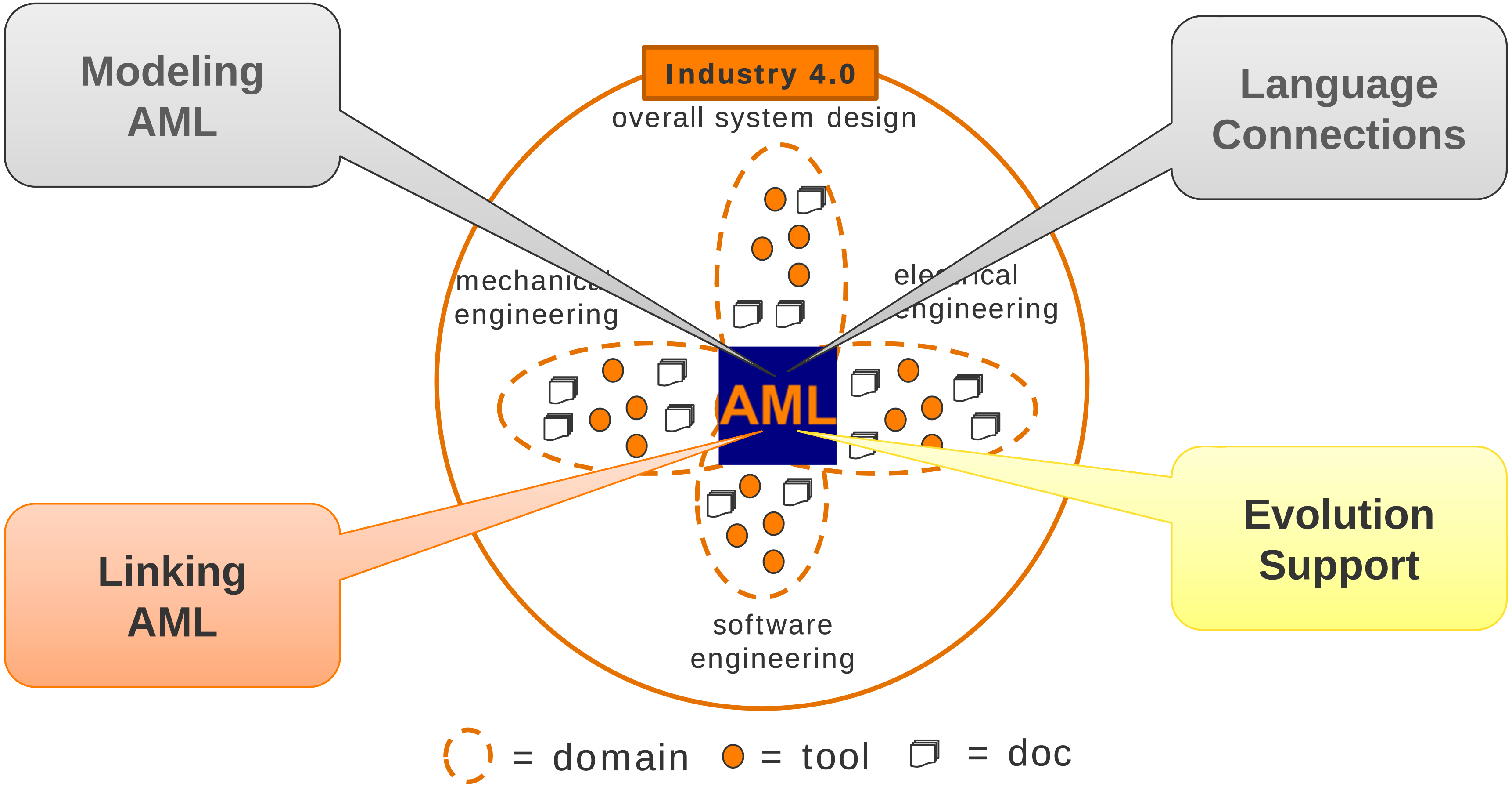
Summary

- Mapping between the structural modeling concepts of AML and SysML
  - Comparison
  - Metamodels
  - UML/SysML profiles
  - Transformations
  - Bridge between IEC and OMG



- Future Work
  - Explore mappings between the behavioral modeling parts of AML PLCopen and SysML Activity Diagrams
  - Code generation, model transformation to formal domains for analysis purposes

Our AML Research Topics



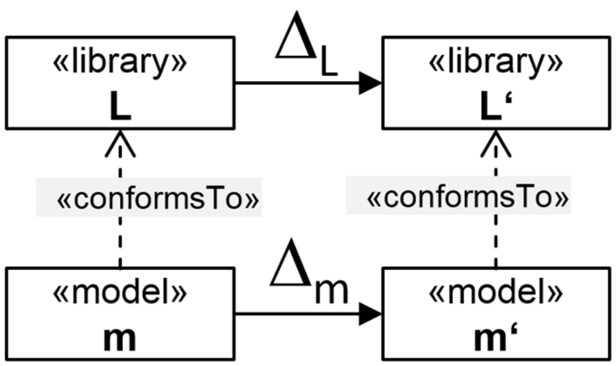
Problem Description

- Engineering industrial production systems is a multidisciplinary activity
  - Engineers from diverse domains are involved
  - Engineers are working in parallel

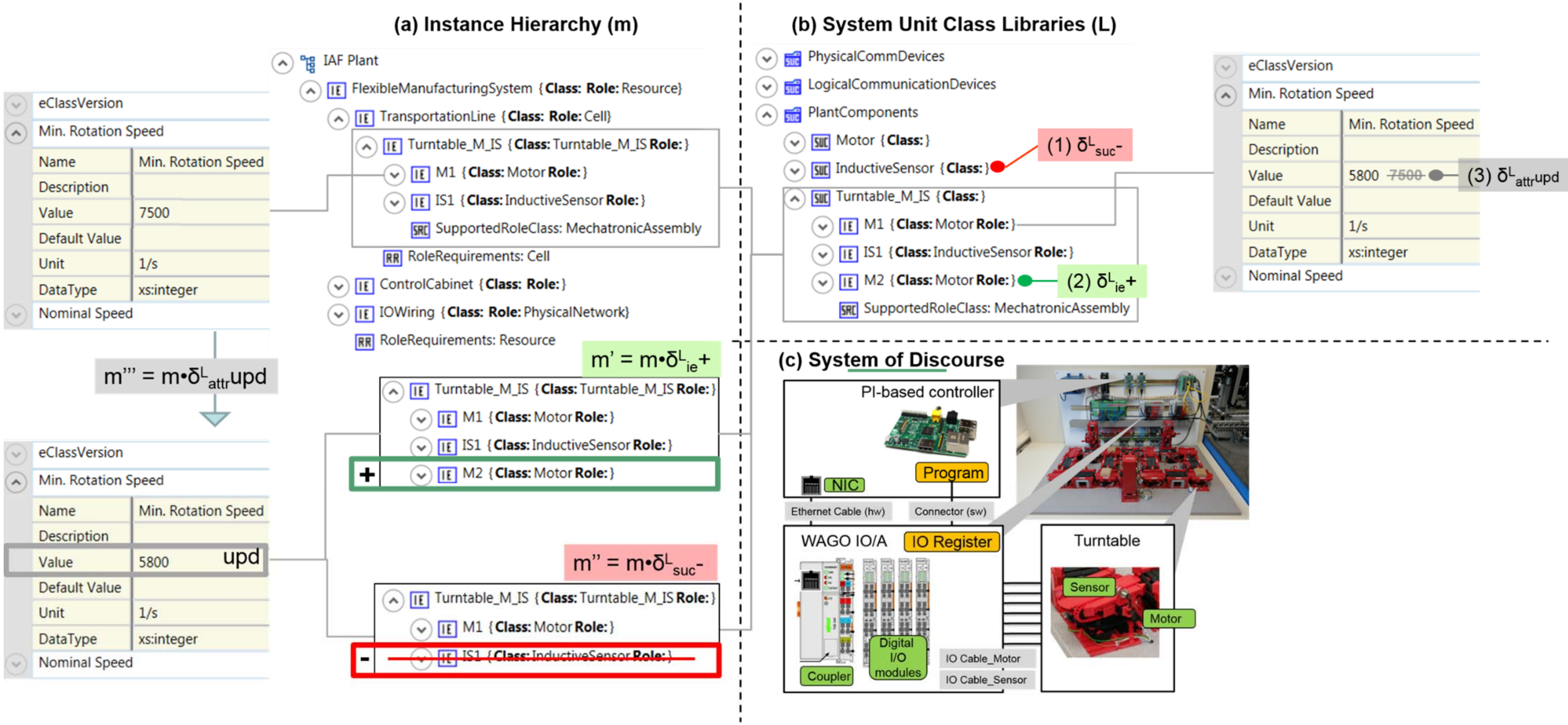
> Challenge: Evolution of engineering data has to be managed

- AutomationML is the predominant standard for representing engineering data of production systems in a model-based way
  - Availability of libraries defining prototypical system elements is an important pragmatics of designing production systems with AutomationML
  - Model of a production system is built by cloning prototypical elements

> Challenge: Co-evolution of prototypes and clones has to be managed



Motivating Example



Contribution

Formal framework to managing prototype/clone co-evolution

1. **Generic metamodel** for prototype-based languages

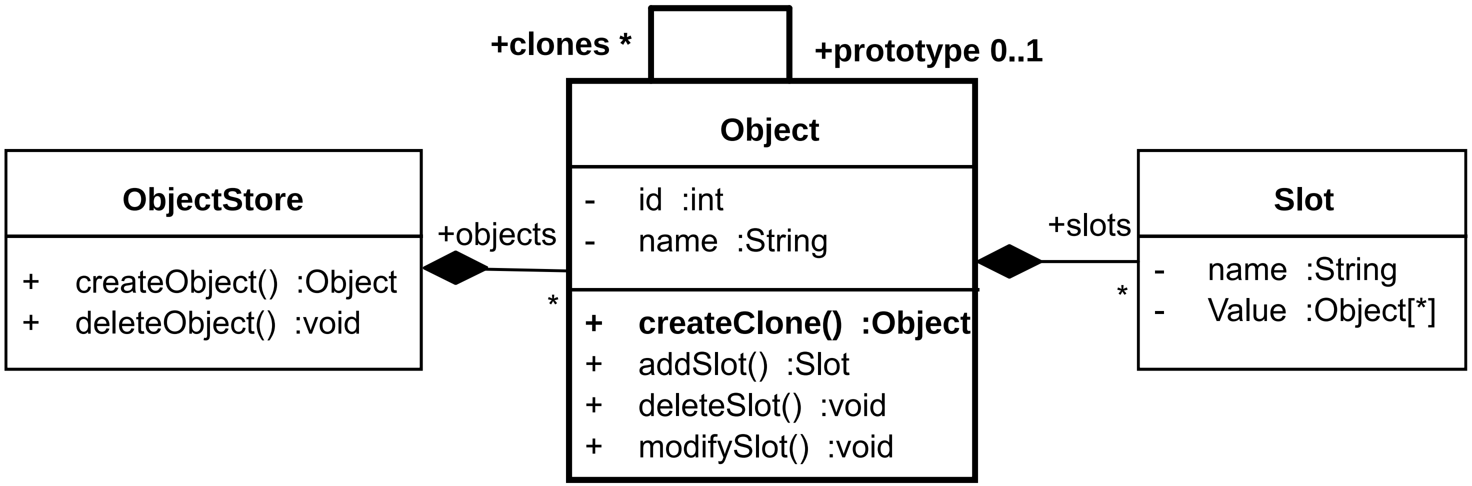
2. **Levels of consistency** rigor between prototypes and clones

3. **Change types** on prototypes and **their impact** on prototype/clone consistency

4. **Repair operations** to re-establish prototype/clone consistency

- L. Berardinelli, S. Biffl, E. Maetzler, T. Mayerhofer, M. Wimmer: **Model-Based Co-Evolution of Production Systems and their Libraries with AutomationML**, 20th IEEE Conf. ETFA, 2015, pp. 1-8.

1. Generic Metamodel for Prototype-Based Languages




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

2. Levels of Consistency Rigor between Prototypes and Clones

- Clones and prototypes may evolve independently
- Different levels of consistency between clones and prototypes may apply

Level 0: Uncontrolled Compliance

- Clones may evolve completely independent from prototypes
- Prototypes are solely used as templates or classification mechanism

Level 1: Substantial Compliance

- Evolution of clones is partially restricted

Level 1a: Extension
Level 1b: Restriction
Level 1c: Redefinition

Level 2: Full Compliance

- Clones may not evolve independently of prototypes

2. Levels of Consistency Rigor between Prototypes and Clones

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

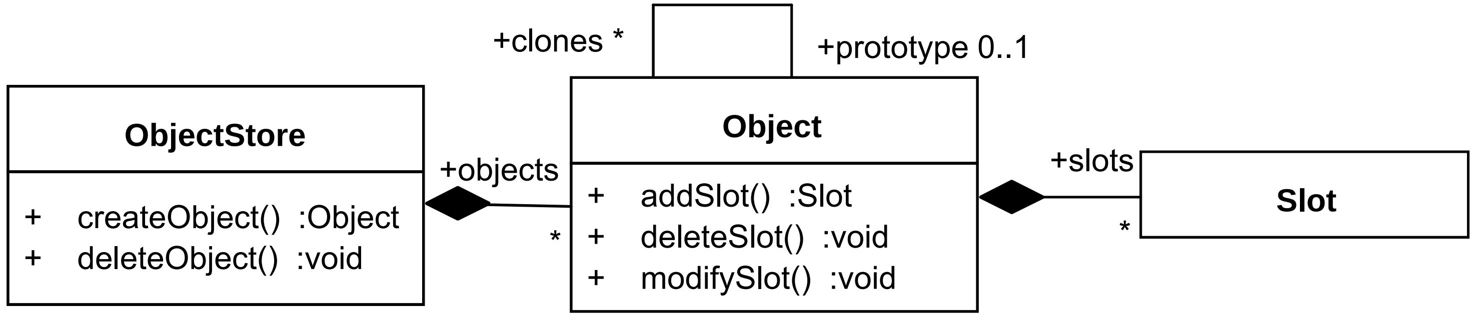
```
-- clone may redefine values of prototype slots
context Object [self.prototype <> OclUndefined]
inv: self.slots -> forAll(cS | self.prototype.slots -> one(pS | cS.name = pS.name))
inv: self.prototype.slots -> forAll(pS | self.slots -> one(cS | pS.name = cS.name))
```

Level 2: Full Compliance

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

3. Change Types on Prototypes and their Impact on Prototype/Clone Consistency

Change Types



Impact on Prototype/Clone Consistency

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

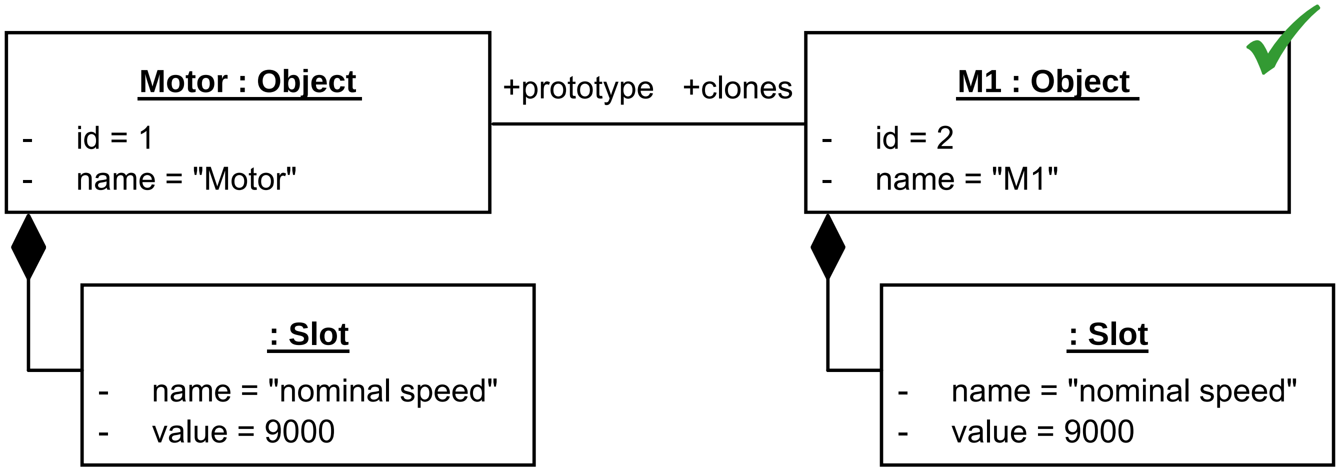
change on prototype — impact on consistency of clones

↑ non-breaking
≠ breaking

3. Change Types on Prototypes and their Impact on Prototype/Clone Consistency

Example

Desired consistency level: L1a Extension

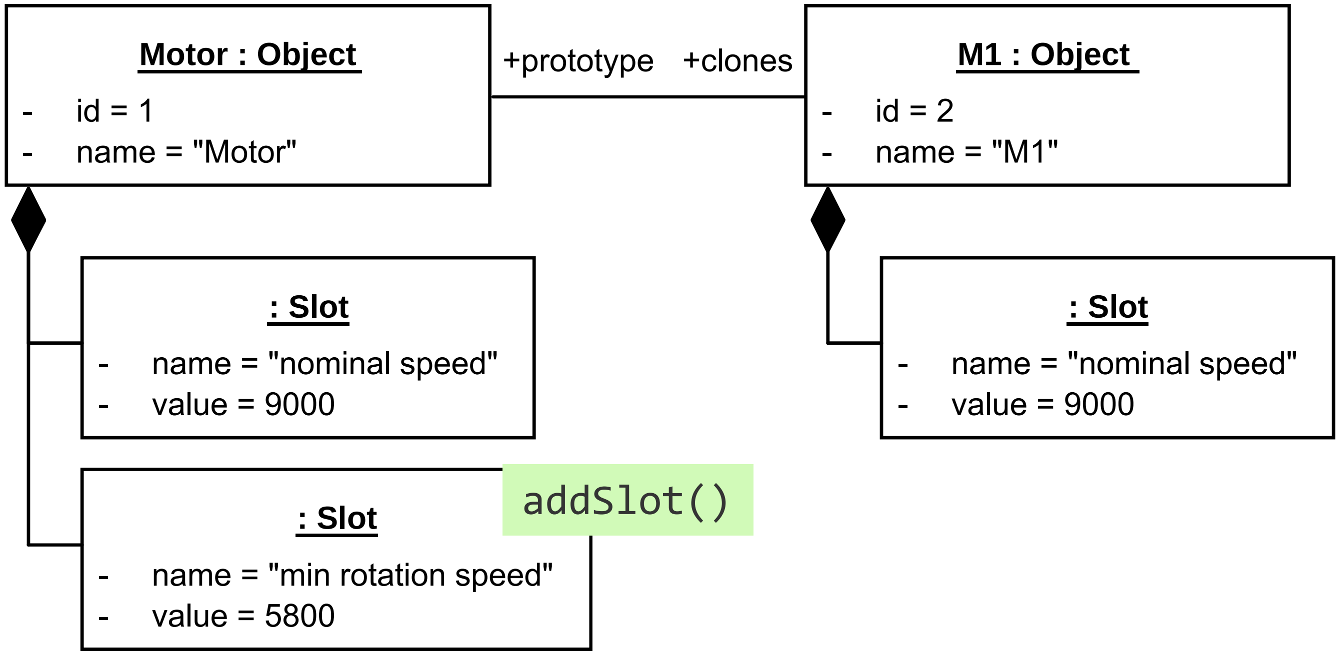



```
-- clone defines all prototype slots but may define additional slots
context Object [self.prototype <> OclUndefined]
inv: self.prototype.slots -> forAll(pS | self.slots -> one(cS | pS.name = cS.name
  and pS.value = cS.value))
```

3. Change Types on Prototypes and Their Impact on Prototype/Clone Consistency

Example

Desired consistency level: L1a Extension

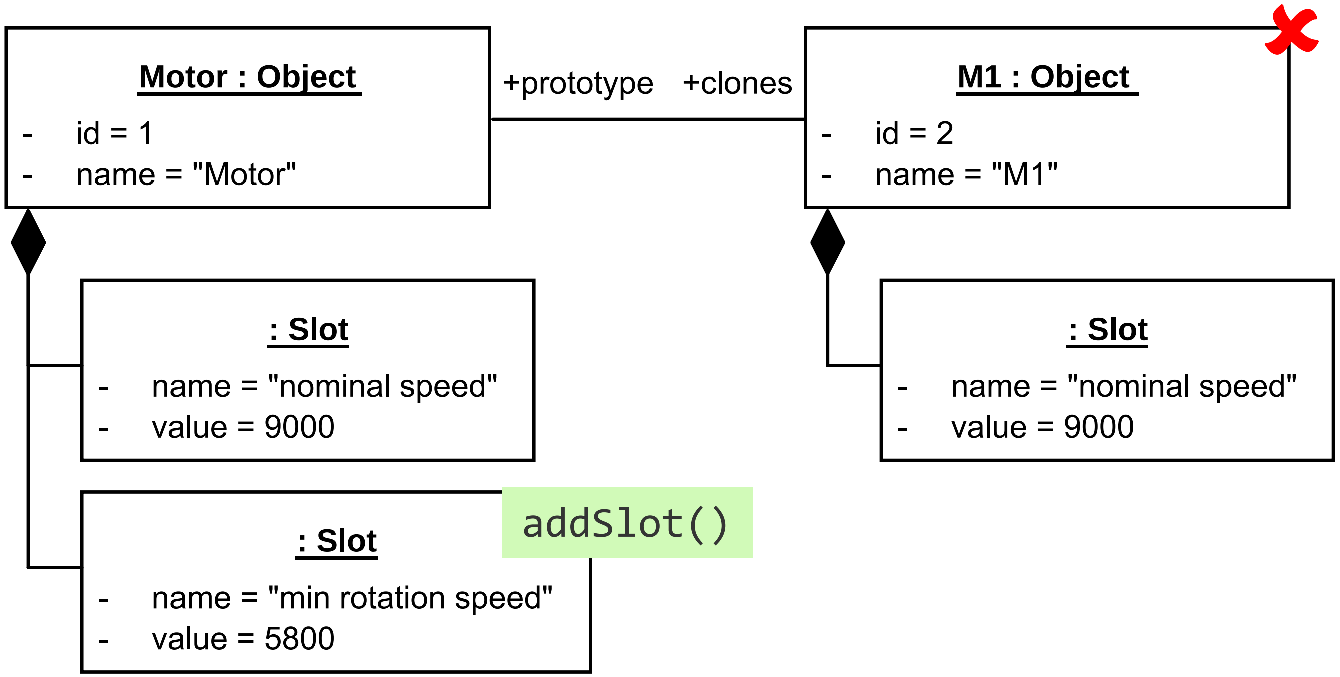



```
-- clone defines all prototype slots but may define additional slots
context Object [self.prototype <> OclUndefined]
inv: self.prototype.slots -> forAll(pS | self.slots -> one(cS | pS.name = cS.name
  and pS.value = cS.value))
```

3. Change Types on Prototypes and Their Impact on 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

- 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

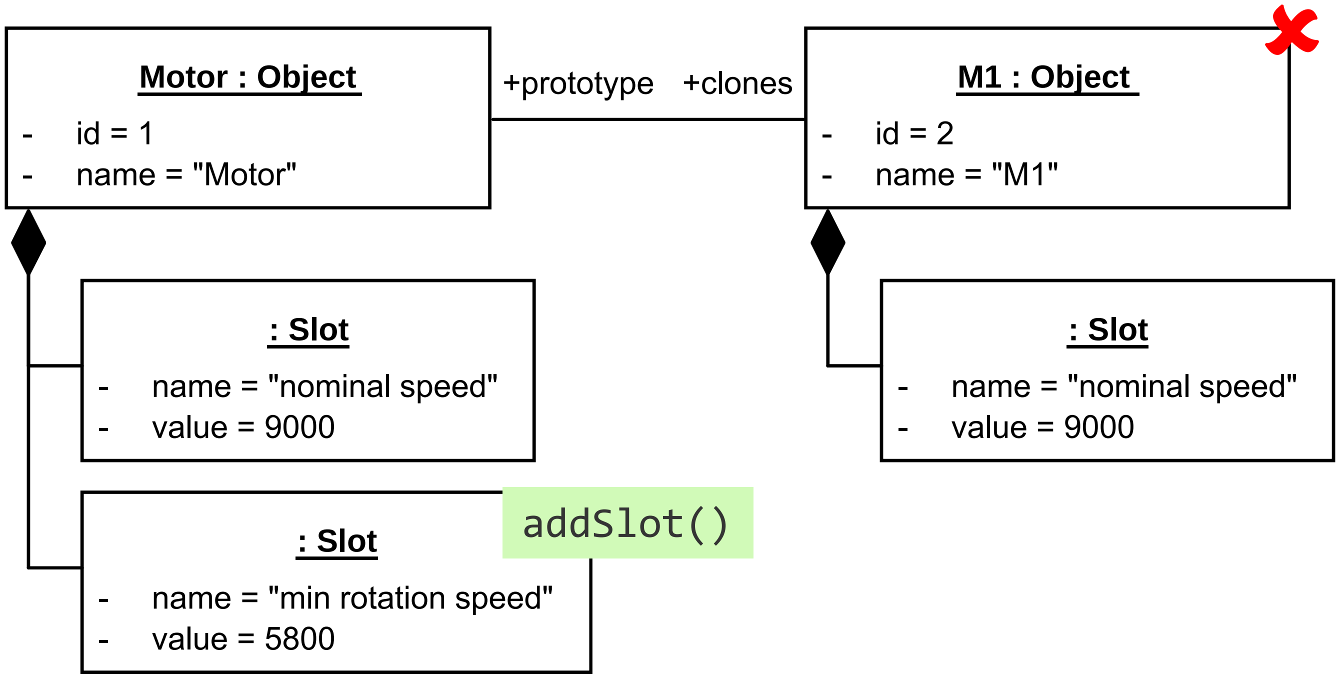
Operation	L0	L1a	L1b	L1c	L2	
ObjectStore::createObject	↑	↑	↑	↑	↑	
ObjectStore::deleteObject	≠	≠	≠	≠	≠	
Object::addSlot	↑	≠	↑	≠	≠	> Add slot to clones
Object::deleteSlot	↑	↑	≠	≠	≠	> Remove slot from clones
Object::modifySlot	↑	≠	≠	↑	≠	> Update slot value in clones

≠ manual resolution needed ≠ automated resolution possible

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

Example

Desired consistency level: L1a Extension

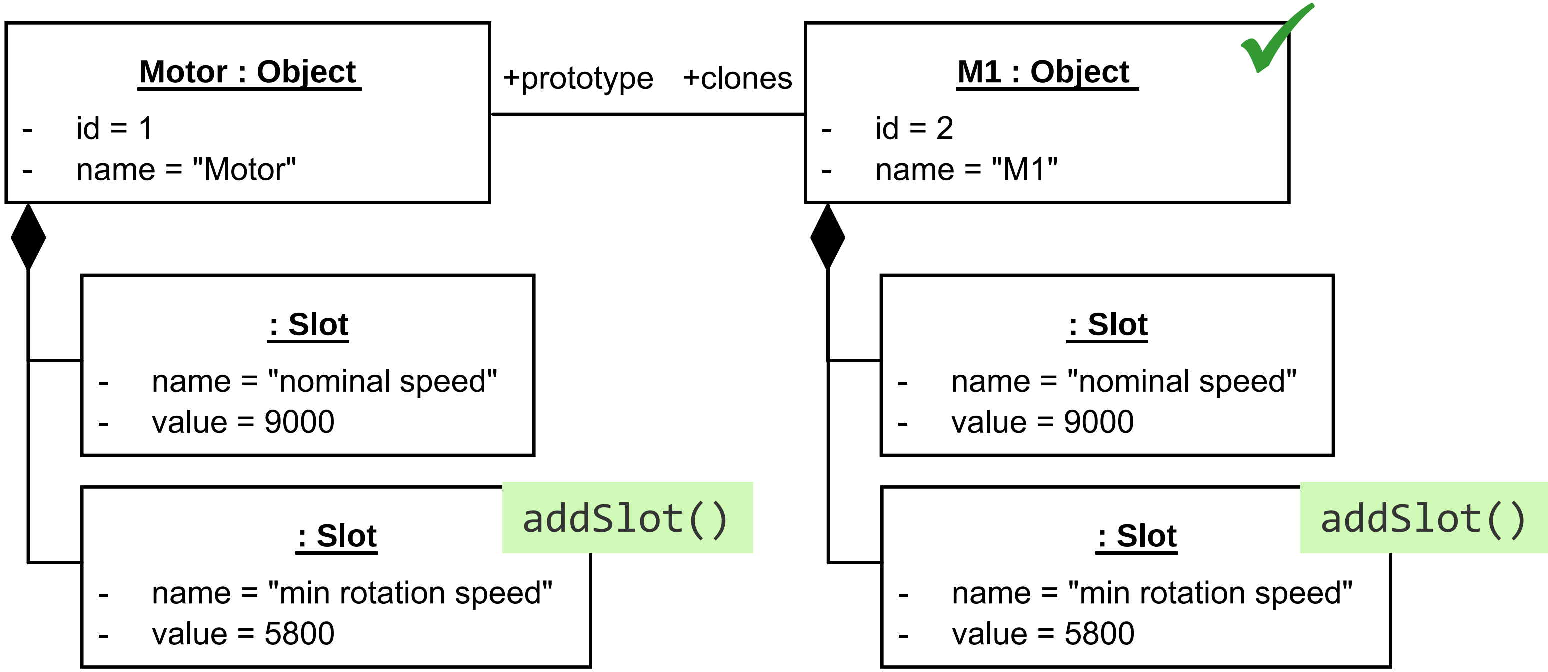



```
-- clone defines all prototype slots but may define additional slots
context Object [self.prototype <> OclUndefined]
inv: self.prototype.slots -> forAll(pS | self.slots -> one(cS | pS.name = cS.name
  and pS.value = cS.value))
```

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

Example

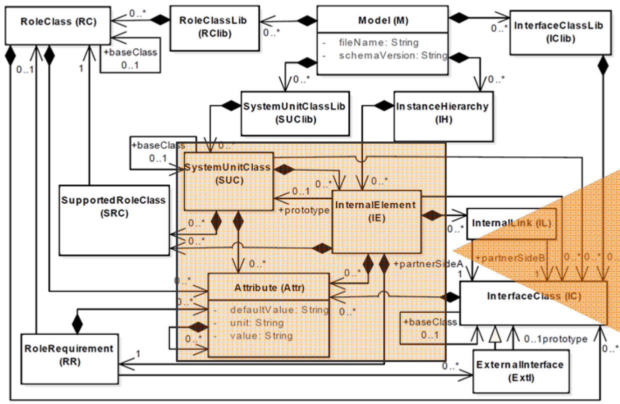
Desired consistency level: L1a Extension



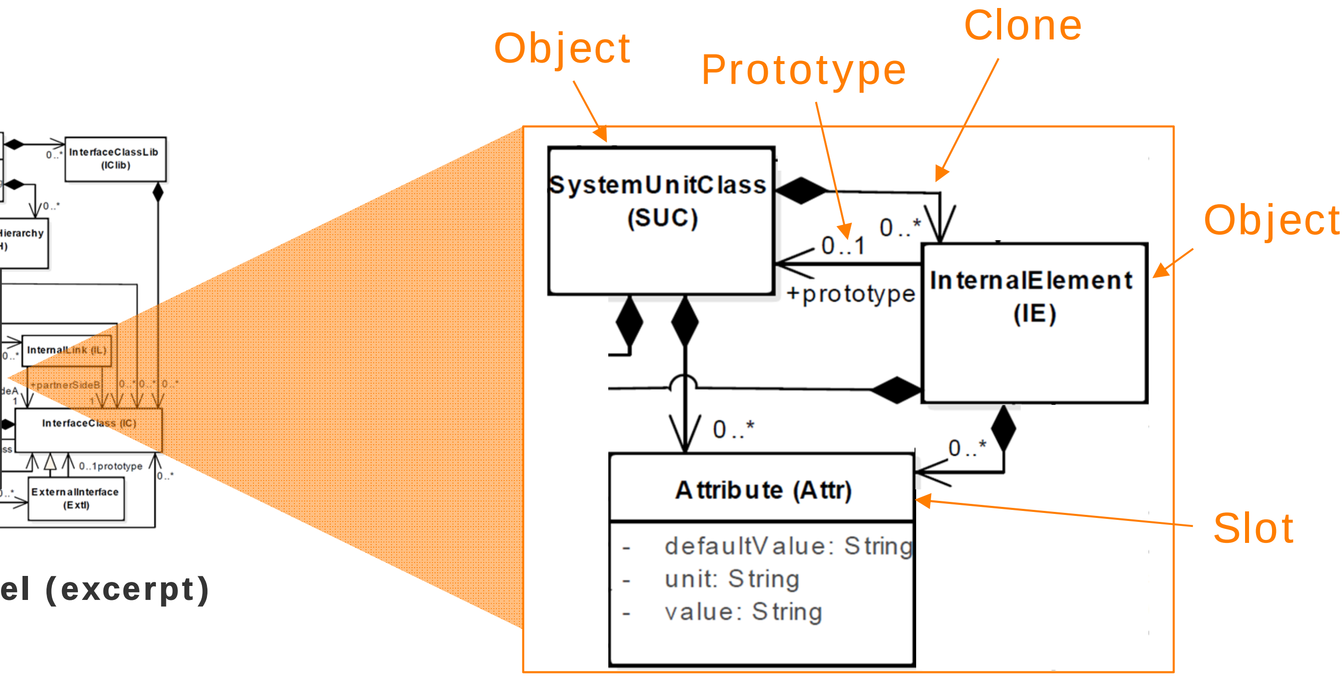

```
fix title : "Add missing slots from prototype" do
  for (pS : self.prototype.slots)
    if (not self.slots -> exists(cS | cS.name = pS.name))
      self.addSlot(pSlot.copy())
```

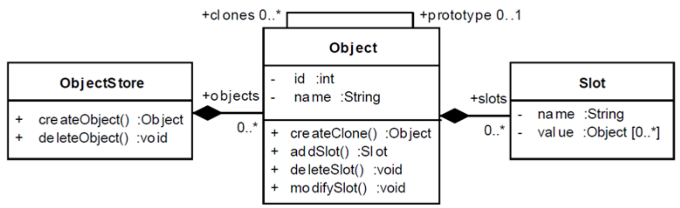
Case Study: AutomationML

- Mapping of generic framework to AutomationML



AutomationML Metamodel (excerpt)

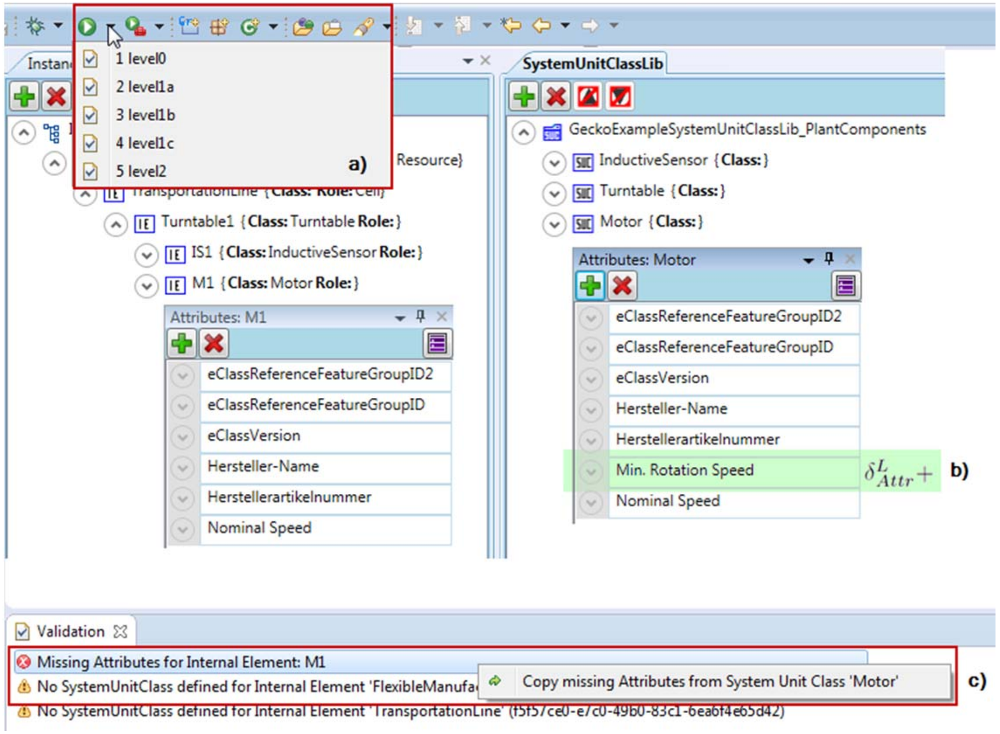




Generic Metamodel for Prototype-Based Languages

Case Study: AutomationML

- Tool support



a) Run configurations for different Consistency Level Rigors

b) Evolution: Attribute added to SystemUnitClass (=Prototype)

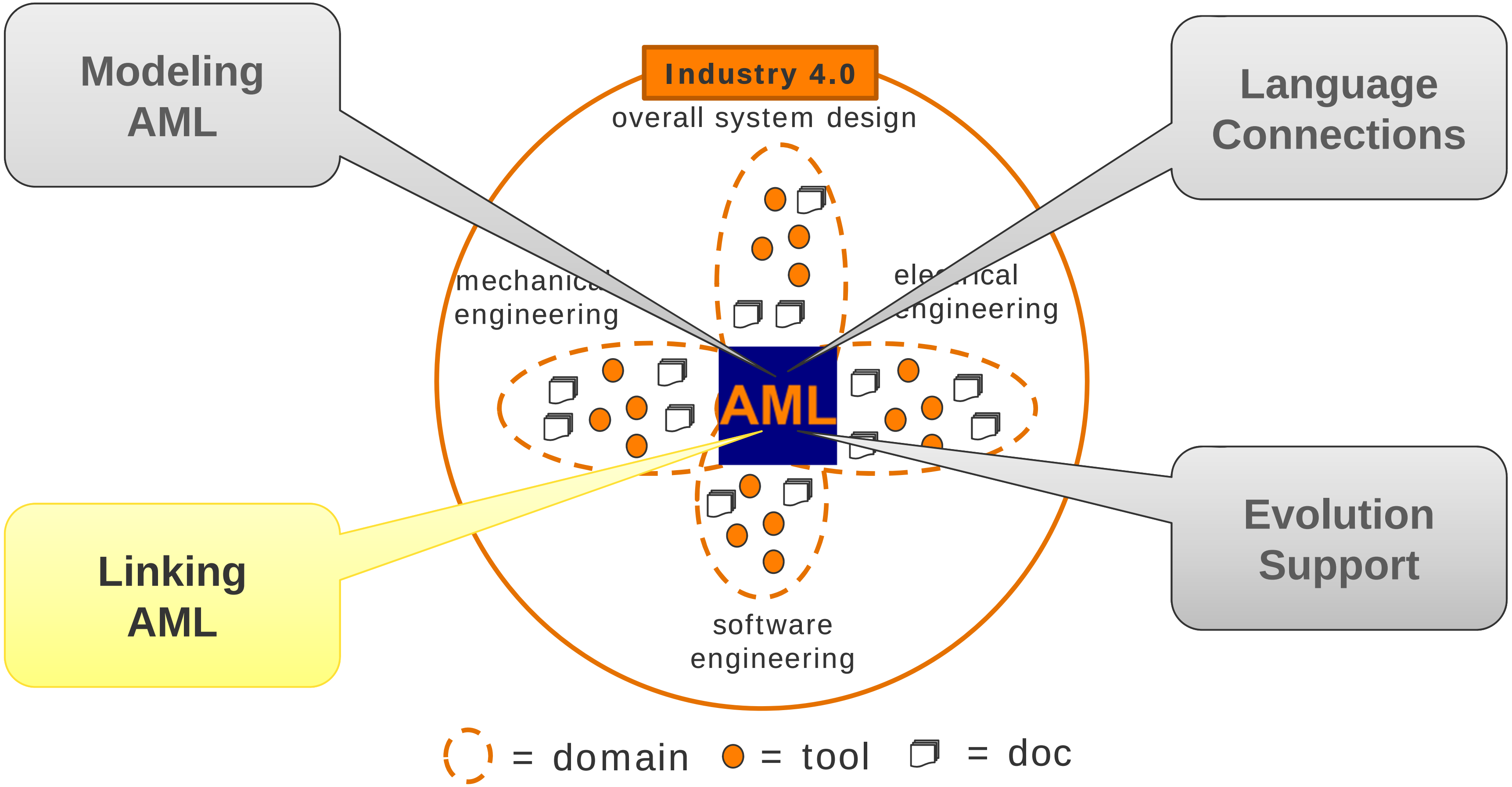
c) Validation Results with proposed fixes

Summary

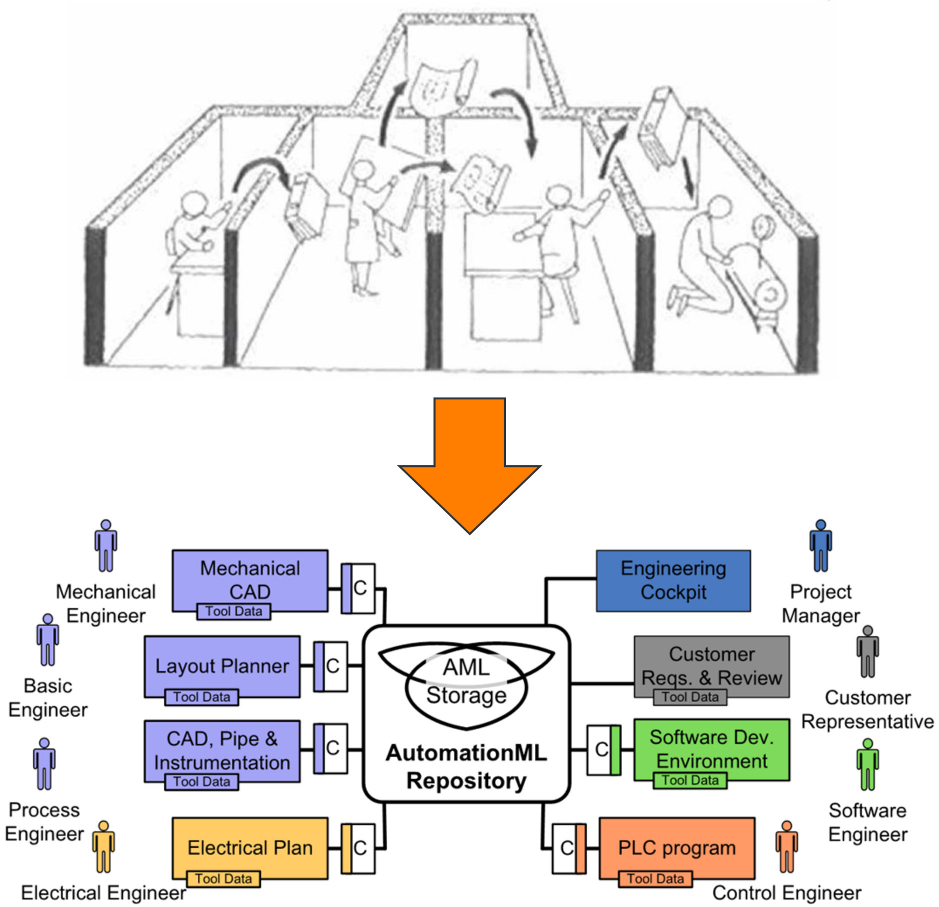
- Evolving libraries and co-evolving system models
- General model to characterize prototype-based languages
- Minimal change model and classified changes based on different consistency levels
- Adopted the general model to AutomationML
- Tool support for consistency checks and (semi-)automated fixing

- Future work
  - Define nesting and inheritance for prototypes in the general model
  - Consider all concepts of AutomationML (e.g., interfaces and roles resulting in multi-level prototype/clone relationships)

Our AML Research Topics

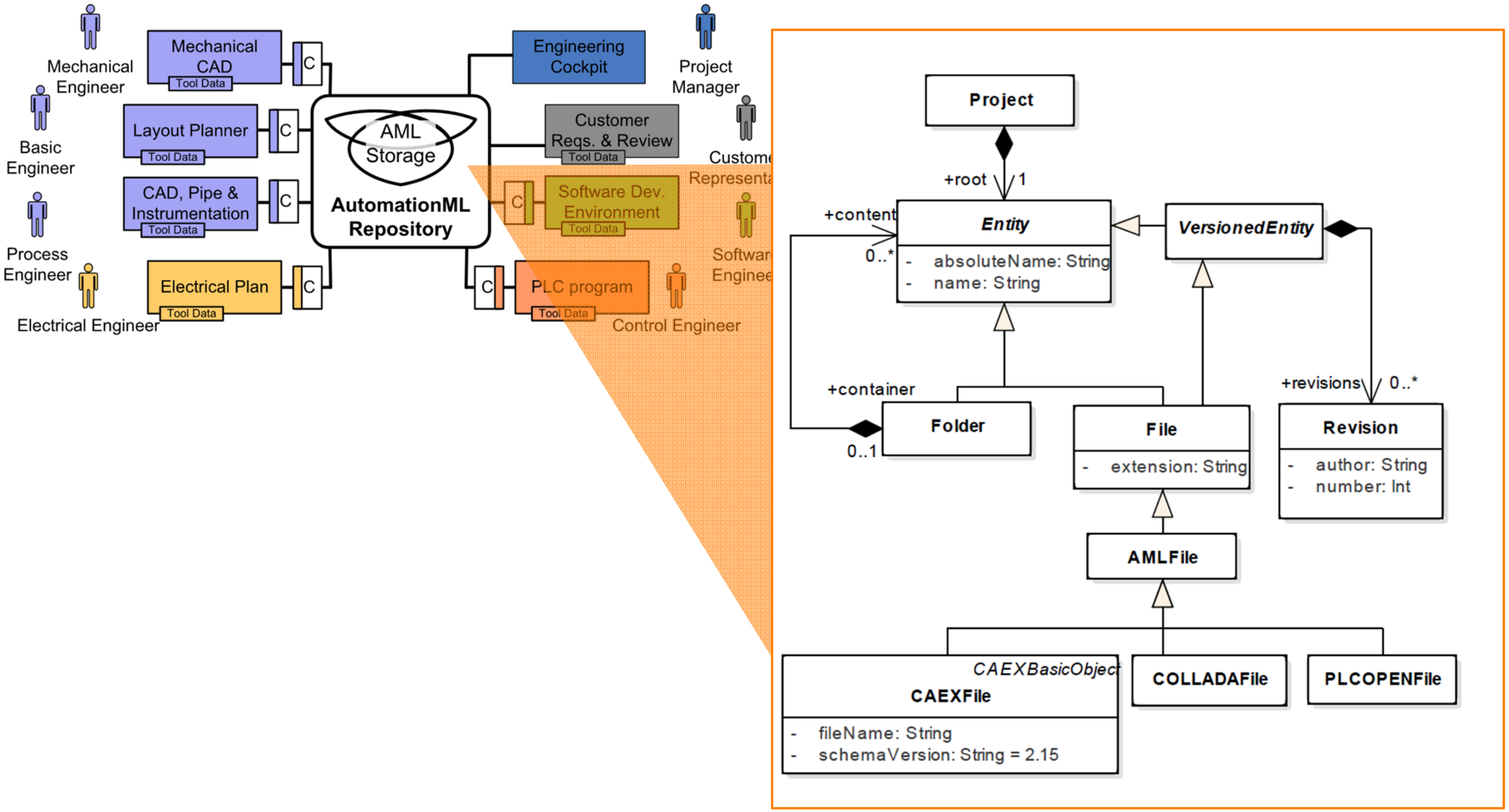


AML Data Integration and Version Management

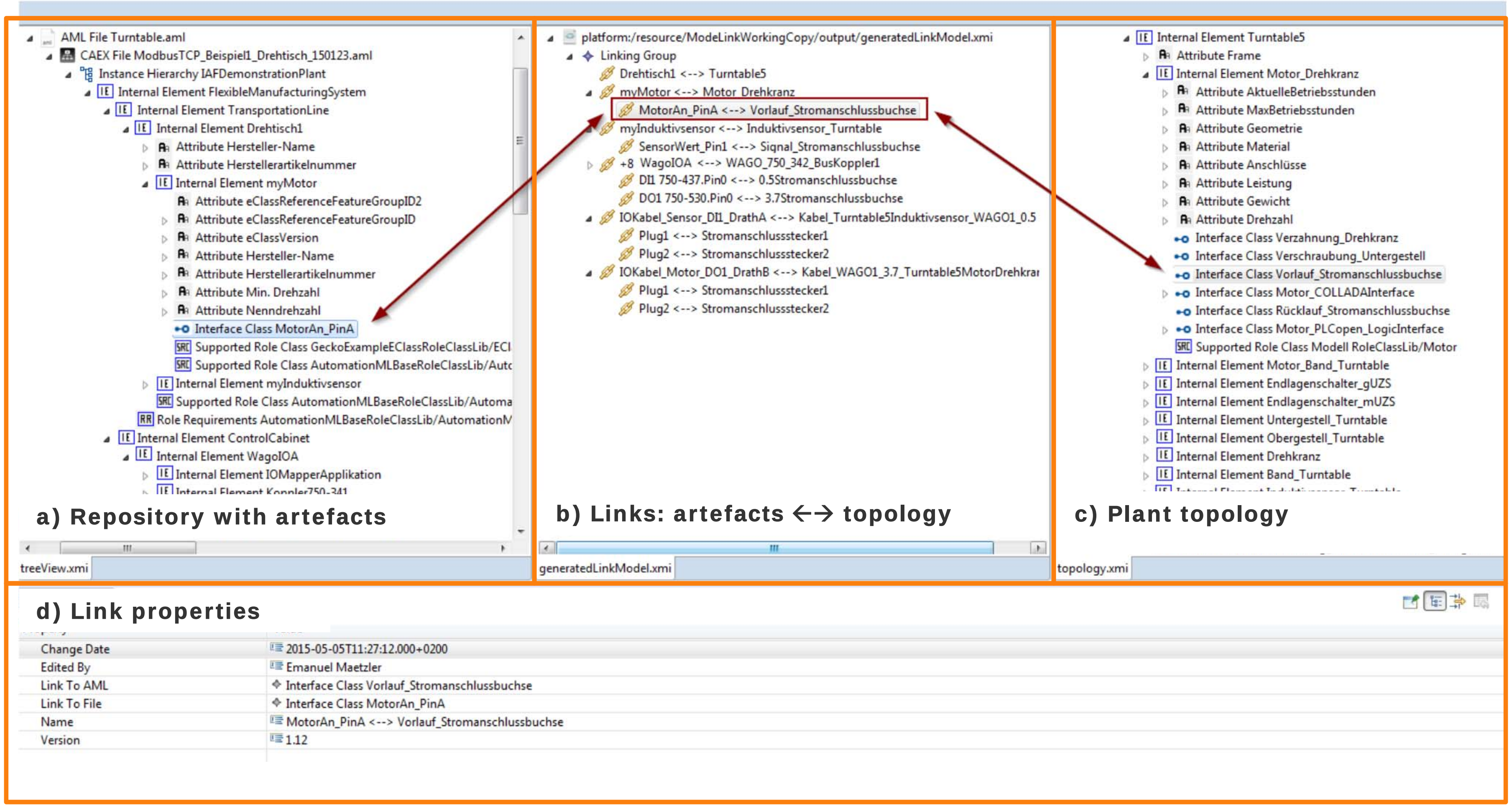


- **Process for AML *Data Integration* and *Version Management***
  - ***Versioning* of Data Elements**
  - ***Linking* the versioned engineering Results**
- **Model-driven tool support**

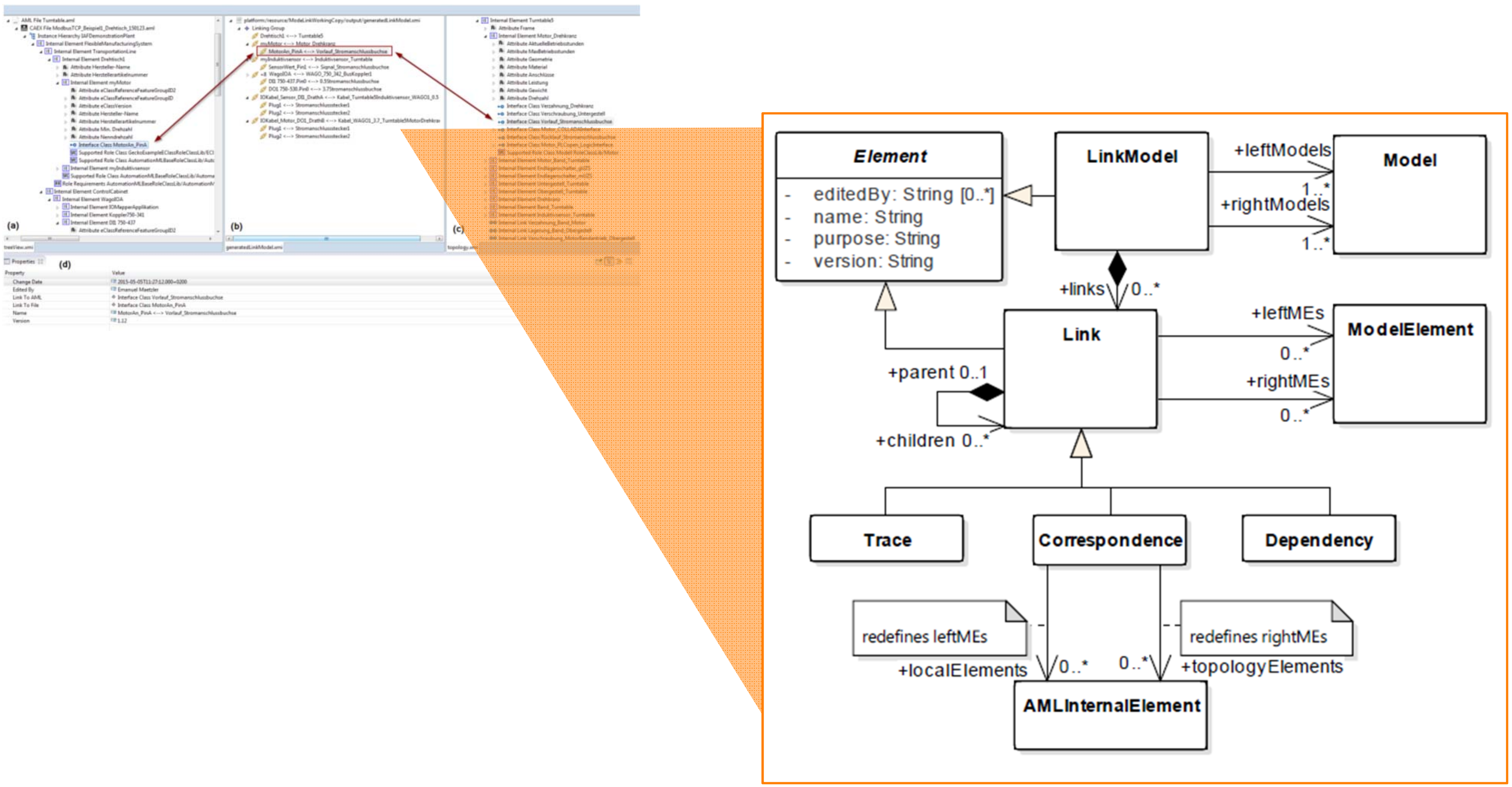
The AutomationML Repository



Linking Engineering Artefacts (view of a plant planner)



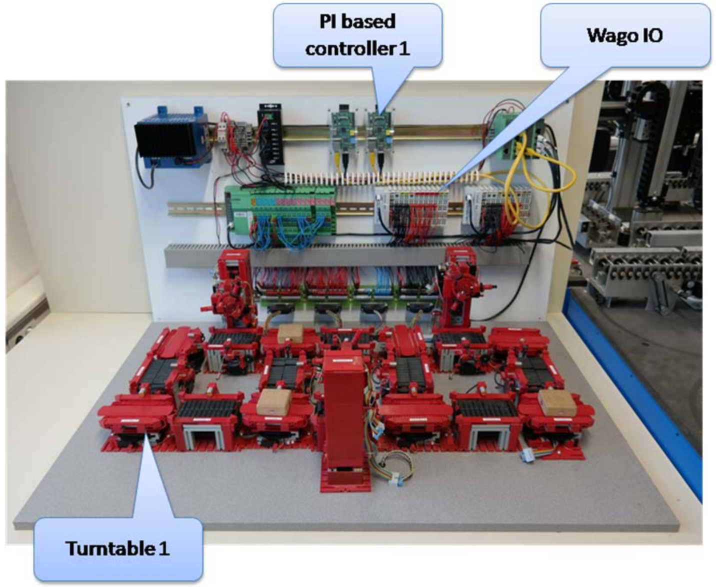
Linking Metamodel



Evaluation

- RQ1 Roundtrip capabilites
  - *Are transformations between AML XML and AML models possible without loss of information?*
  - **Result:** All reference examples and real world examples could be transformed to AML models and back to AML XML without loss of information

- RQ2 Integration capabilites
  - *Is the linking language expressive enough for practical settings?*
  - **Result:** All mappings of a lab-sized production system (picture to the right) could be modeled



Lab-sized Production System
"Equipment Center for Distributed Systems," http://www.iafbg.ovgu.de/en/technische ausstattung cvs.html, Institute of Ergonomics, Manufacturing Systems and Automation at Otto-v.-Guericke University Magdeburg.

Content

- Introduction
  - *Modeling and Model-Driven Engineering in Software Engineering*
  - *Cyber-Physical Production Systems (CPPS)*
- MDE in CPPS I: *Interface Integration*
- MDE in CPPS II: *Model Exchange*
- **Résumé**

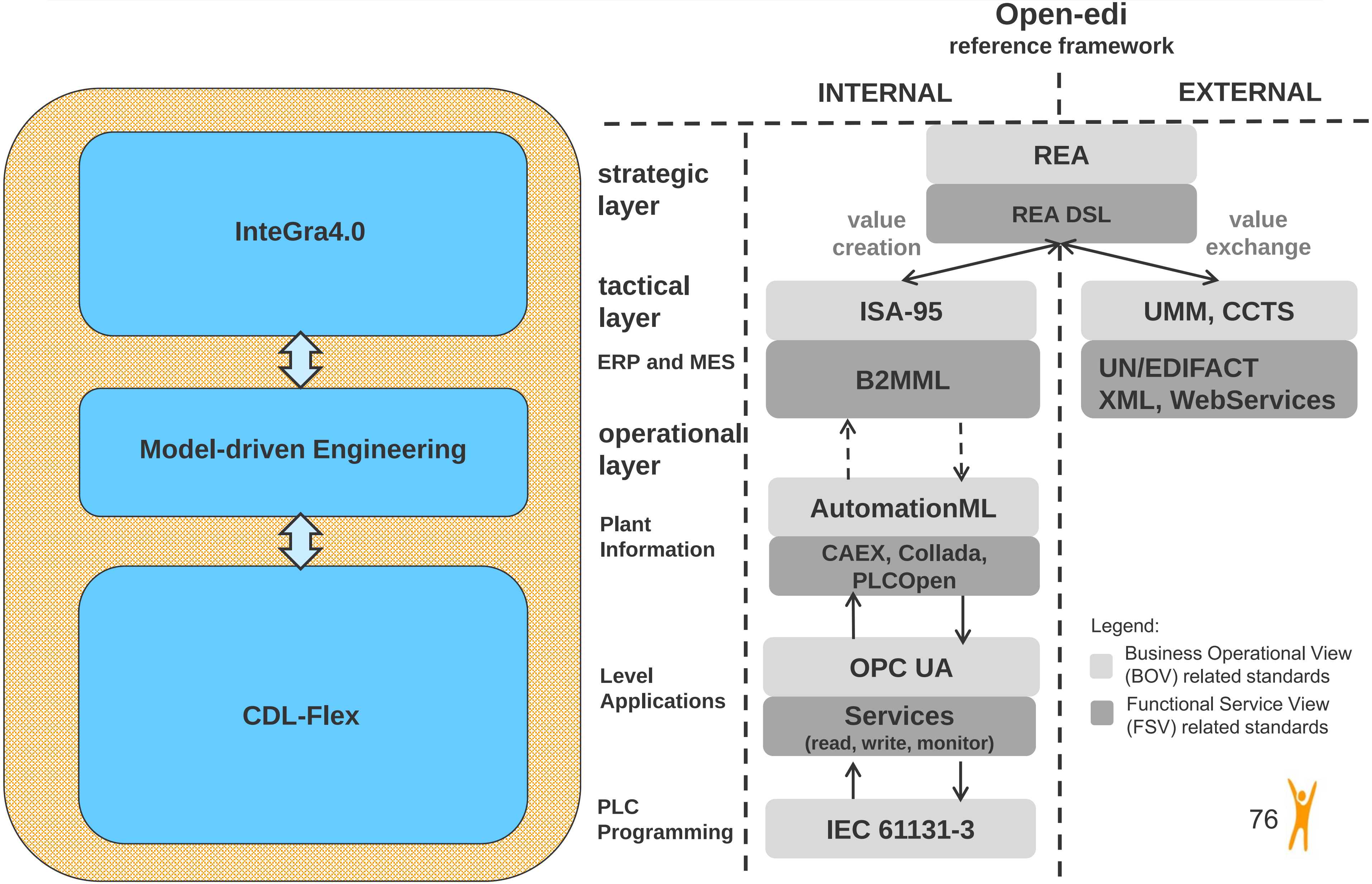
Résumé

Our Lessons Learned

- **Model-Driven Engineering** is beneficial to
  - Represent modeling languages
  - Derive tool support
  - Bridging different languages

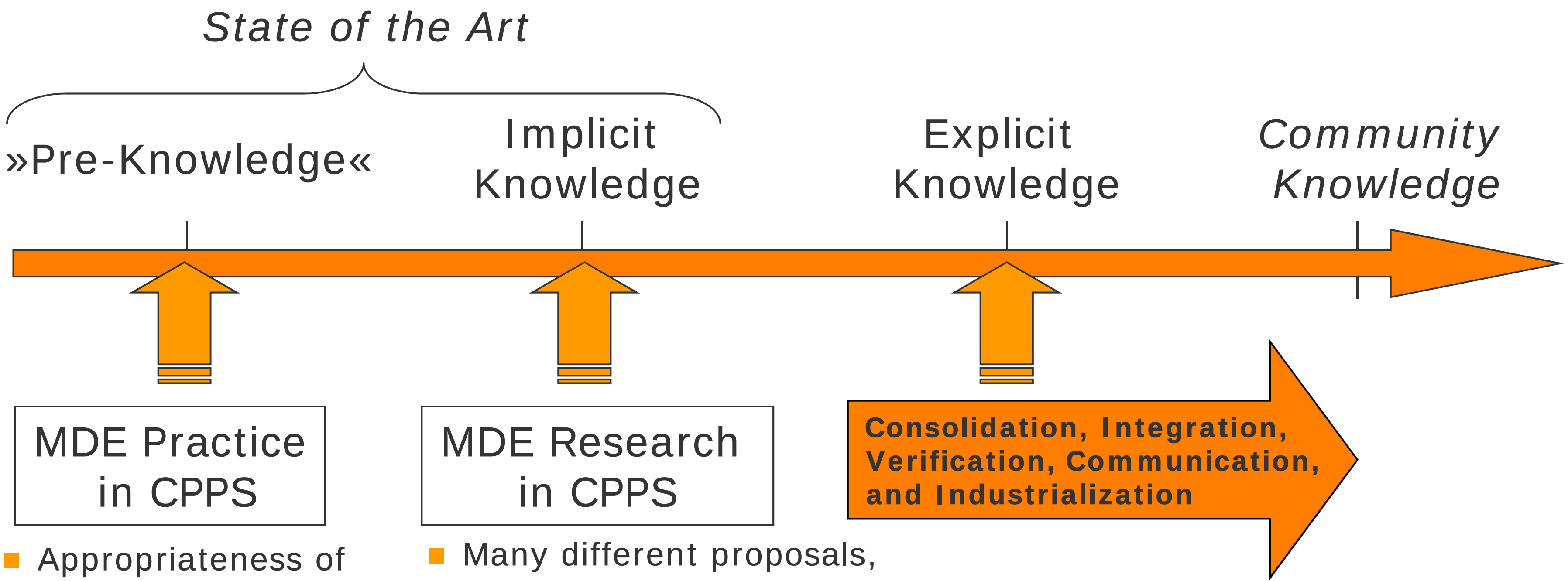
- Resulting **modeling tools** are
  - **Open** and **extensible**
  - Usable in **combination** based on model exchange
  - Allow for a **mixture** of modeling languages leading to **multi-paradigm modeling** approaches
  - **Model management** support is available **out-of-the-box** based on common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

Next Step: Models, Standards, and Technology for Digital Transformation



Résumé

Model-Driven Engineering in CPPS – Still enough to do :-!



MDE Practice in CPPS

- Appropriateness of some standards questionable (SysML) – not yet adopted
- CASE-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, ...
- Emerging standards and initiatives
  - E.g., ManTIS DTF @ OMG

Résumé

- Are existing **standards mature enough** to represent a proper basis for engineering CPPS ...

  *... or are they just a more or less useful **patchwork of interests** of different parties?*

- Are existing **MDE-tools** capable to **manage increasing systems complexity** ...

  *... or doesn't they contribute even **more** to the **complexity** of systems engineering?*

- Do we already **understand** the „**modeling phenomenon**" enough in order to build appropriate MDE techniques ...

  *... or are we still in the „crafts(wo)menship" phase, recalling **just another CASE-tool area**?*

Thanks to…


Alexandra Mazak

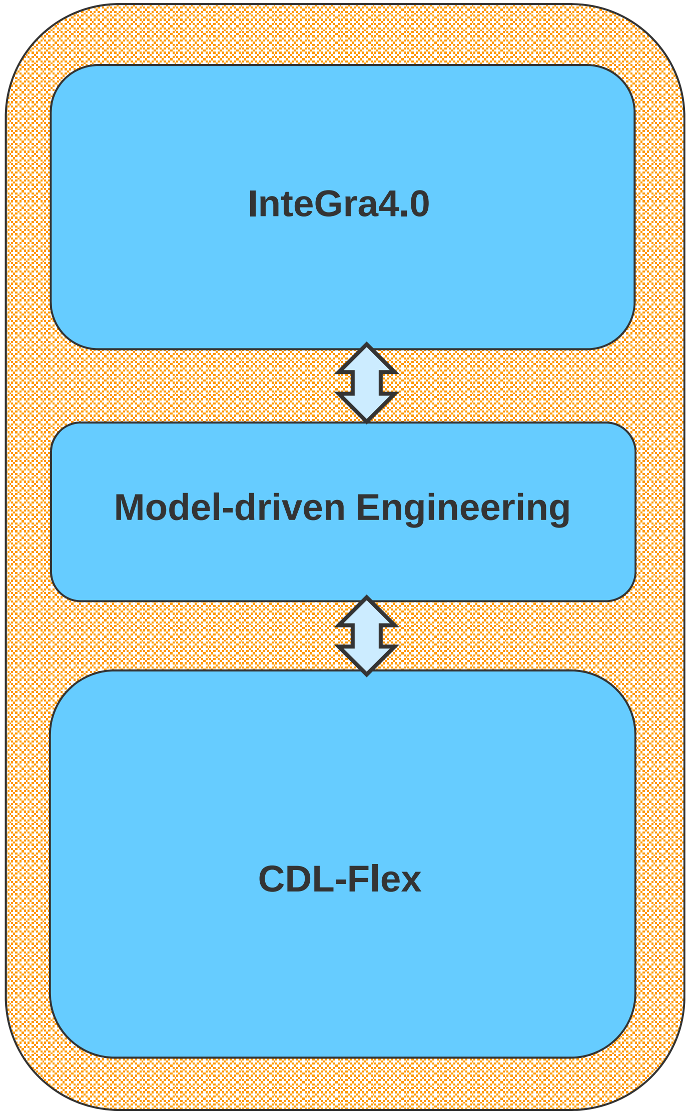

Christian Huemer


Tanja Mayerhofer


Manuel Wimmer


Luca Berardinelli


Emanuel Mätzler



LieberLieber software gmbh

CERTICON ADDED VALUE SOLUTIONS

logi.cals®

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG


Stefan Biffl


Arndt Lüder