Christian Doppler Laboratory

Software Engineering Integration for Flexible Automation Systems

Modeling Cyber-Physical Production Systems

Manuel Wimmer
Institute of Software Technology and Interactive Systems
Vienna University of Technology
Engineering of Cyber-Physical Production Systems (CPPS)

- **Industrie 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
  - ...

---

![Diagram showing overall system design with mechanical, electrical, and software engineering domains, interconnected with tools and documents.](image)
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 a heterogeneous tool network by means of a **unified format** and **data model**

- AutomationML website: http://www.automationml.org
AutomationML = Automation (Markup | Modeling) Language?

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.

```xml
<InstanceHierarchy Name="Parent child relations example">
  <InternalElement Name="ObjectA" ID="GUID1">
    <InternalElement Name="ObjectA_1" ID="GUID2">
      <InternalElement Name="ObjectA_2" ID="GUID3">
        <InternalElement Name="ObjectA_2_1" ID="GUID4/>
      </InternalElement>
    </InternalElement>
  </InternalElement>
</InstanceHierarchy>
```
From Tree-based to Graph-based Representations
Language Engineering via Metamodelling

AutomationML by Example

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

System under Study

- PI-based controller
- Ethernet Cable (fh)
- Connector (sw)
- WAGO IO/A
- IO Register
- Digital I/O
- Modules
- Coupler
- Ethernet Cable_Motor
- IO Cable_Sensor
- Turntable
- Motor
- Sensor

System Unit Class Library (SUC, IE, ExtInt)

- Geck0ExampleSystemUnitClassLib_PlanComponents
- Motor (Class:)
  - MotorAn_PinA (Class: SignalInterface)
  - SupportedRoleClass: 27-02-25-03 DC-Motor (IEC)
  - SupportedRoleClass: Device
  - SupportedRoleClass: MechatronicAssembly

- InductiveSensor (Class:)
  - Turntable (Class:)
    - myMotor (Class: Motor Role:)
    - myInductiveSensor (Class: InductiveSensor Role:)
    - SupportedRoleClass: MechatronicAssembly

Role Class Library (RC, ExtInt)

- Resource (Class: AutomationMLRole)
- Product (Class: AutomationMLRole)
- Process (Class: AutomationMLRole)
- Structure (Class: AutomationMLRole)
- ProductStructure (Class: Structure)
- ProcessStructure (Class: Structure)
- ResourceStructure (Class: Structure)
  - Cell (Class: ResourceStructure)
  - MainGroup (Class: ResourceStructure)
  - SubFunctionGroup (Class: ResourceStructure)
  - MechatronicAssembly (Class: ResourceStructure)
  - MechanicalAssembly (Class: ResourceStructure)
  - MechanicalPart (Class: ResourceStructure)
  - Device (Class: ResourceStructure)

Interface Class Library (IC)

- AutomationMLInterfaceClassLib
  - Order (Class: AutomationMLInterface)
  - PortConnector (Class: AutomationMLInterface)
  - InterlockingConnector (Class: AutomationMLInterface)
  - PPRConnector (Class: AutomationMLInterface)
  - ExternalDataConnector (Class: AutomationMLInterface)
  - COLLADAInterface (Class: ExternalDataConnector)
  - PIoCopenXMLInterface (Class: ExternalDataConnector)
  - Communication (Class: AutomationMLInterface)
Metamodelling 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

Metamodel/Model Relationships

model:: IH
Further Benefits of Explicit Models

Model Transformation Pattern and Supporting Tools

Transformation Scenario Investigated
AML and SysML: Two Unrelated Modeling Standards

overall system design
mechanical engineering
software engineering

= domain
○ = tool
☐ = doc
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)
Flexible Manufacturing System in AML and SysML
Conclusion and Ongoing Work

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

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

- **Next steps**
  - **Mappings** between the behavioral modeling parts of AML PLCopen XML and SysML Activity Diagrams and State Machines
  - **Generative usage** of AML models by defining code generator chains
  - **Analytical usage** of AML models by transforming them to formal domains
  - **Catalogue** of CPPS related modeling languages
    - MPM4CPS EU Cost Action
    - http://www.cost.eu/COST_Actions/ict/Actions/IC1404