Software Engineering Integration For Flexible Automation Systems

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

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

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

- Identified needs
  a. A large number of heterogeneous models is involved in systems engineering
     → Need for process support
  
  b. A wide variety of models in systems engineering come from different disciplines
     → Need for management
     → Need for customized integration for concrete application scenarios
     → Vision: encompassing various viewpoints to get a better understanding

→ At TU Wien we have the expertise
  a. To **develop software tools the model driven way** (AutomationML Hub example)
  b. To **deal with a large number of heterogeneous** software and systems models
Model Exchange: AML as Common Format

- AutomationML (AML)

- Emerging **standard for tool data exchange**

- Foundation for harmonizing engineering data coming from a heterogeneous tool network by means of a **unified format and data model**
How to get from Data to Models?
Anatomy of modeling languages

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

- **Semantics**: Defines the meaning of the language concepts
  - How language concepts are interpreted

- **Abstract syntax**: Defines the language concepts and how these concepts can be combined (~ grammar)
  - However, it does not define the notation or meaning of the concepts

- **Concrete syntax**: Notation to illustrate the language concepts intuitively
  - 2 ways: textual or graphical (or mixture)

- **Serialization syntax**: For persistent storage and model exchange between tools
  - XML, proprietary format, …
Language Engineering via Metamodeling

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

---

Language Engineering via Metamodelling
First Interactive session

- **Reqirements**
  - AutomationML Metamodel ([https://github.com/amlModeling](https://github.com/amlModeling))
  - Xtext for textual concrete syntax ([https://eclipse.org/Xtext](https://eclipse.org/Xtext))
Interactive Session (1/2)
Model Transformations
Pattern

Model Transformations: AML and SysML
Two Unrelated Modeling Standards

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

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

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

**Model Transformations: AML and SysML**

**SysML in a Nutshell**

- **Additions to UML** for Requirements and Properties
  - Requirement: SysML provides modeling constructs to represent text-based requirements and relate them to other modeling elements.
  - Constraints and Parametric Diagram (constraint analysis)

- **Customization of UML** for structural modeling through Classes and Composite Structures
  - Block derives from CompositeStructures::Class
Model Exchange: AML as Common Format
Case Study: Six Axes Robot
Model Transformations: AML and SysML
From AutomationML to Enterprise Architect and Back again: Example

- **Commonalities and differences** between the structural modeling sublanguages of AML (CAEX) and SysML (Block Diagrams)
- **AML metamodel and profiles** for UML and SysML
- **Transformations** between AML and SysML (UML/SysML already available)
Model Transformations: AML and SysML
From AutomationML to Enterprise Architect and Back again: Example

AML Editor Tree

Export

Import
Model Transformations: AML and SysML

From AutomationML to Enterprise Architect and Back again: Example

AML Editor Tree
Model Transformations: AML, SysML and UML
Second Interactive session: From AutomationML to EA and Back again

- **Requirements**
  - Enterprise Architect (http://www.sparxsystems.de/uml/download-trial/)
  - AML Plugin for EA (Prototype: http://www.sysml4industry.org/?page_id=266)
Interactive Session (2/2)
Conclusions and Future Work

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

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

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