Semantic Integration of Data Models Across Engineering Disciplines

Stefan Biffl   Thomas Moser
Christian Doppler Laboratory SE-Flex-AS
Institute of Software Technology and Interactive Systems (ISIS)
Vienna University of Technology
http://cdl.ifis.tuwien.ac.at
Agenda

- Scope
- State-of-the-Art
- Research Methods
- Semantic Integration Challenges
- Semantic Integration Solution Approach
- Use Case: End-to-End Analysis
- Planned Research
Scope

- **Focus**: Semantic Integration of heterogeneous Software Engineering Models across Tools and Domain Boundaries

- **“Engineering Knowledge Base” (EKB)**
  - make existing knowledge *explicit* and *machine-understandable* to *automate* processes on project level that build on this knowledge
Semantic Integration of Engineering Knowledge

Identification of common concepts across engineering disciplines

Tool A Data Model
Tool B Data Model
Tool C Data Model

Domain/project data model

Requirements
Location IDs
Components
Interfaces

Transmission lines
Terminal points

Mechanical equipment properties

Machine vendor catalogue

Data Types
Logical Behavior

Signals (I/O)

Electrical Engineer

Software Engineer

Process Engineer

Software
Engineer

Mechanical equipment properties

Analog

0 to 10 V

X.22.2.1

Tool A – Domain

Tool A – Tool B

Model Mapping

Derived Mapping

Common concept

Information
Analog
0 to 10 V
X.22.2.1

„Pump flow“
Real (l/min)
0 to 1,200

%I20.5.3
Use Case: Signal Engineering

Use of common concepts in models across engineering disciplines

Sensor

System Interface

Software Interface

Software “Behavior”

S1

S2

S3

S4

S5

V_A

V_B

V_C

V_D

Wiring

Configuration

Use of Data

Electrical Engineer

Configurator

Software Engineer

End-to-End Analysis

- List of sensor name/description/type with Variable name/description/type
- Warnings for incomplete chains between variables and sensors
State-of-the-Art

- **Semantic Heterogeneities**
  - Reconciling of data schema heterogeneities of data residing in multiple sources requires **both domain and technical expertise**
  - **usage of standard schemas** only has success in domains where the incentives to agree on standards are very strong

- **Semantic integration** of engineering knowledge across systems
  - **Unified modeling** approaches for software and systems (e.g., SysML)
  - Standards for **data and knowledge description** (e.g., XML, RDF)
  - **Engineering knowledge** representation, e.g., manufacturing ontology

- **Ontology Alignment (OA)**
  - In a safety critical domain, OA can not be done fully automated as a close to zero error rate is necessary
  - However, OA approaches can be used to **support manual alignment** and review of mappings
Research Methods

- **Systematic Literature Research**
  - Semantic Integration
  - Ontology Alignment

- **Explorative prototypes** to capture industry Use Cases

- **Concept Development**

- **Research Prototype Development**

- **Empirical Evaluation Criteria**
  - **Effectiveness**: the feasibility, validity and correctness of the EKB
  - **Efficiency**: the effort needed for setting up the EKB
  - **Performance**: the run-time performance of the EKB
  - **Robustness**: the identification and handling of defects by the EKB
  - **Scalability**: the extendibility of the EKB architecture
  - **Usability**: the usability of the EKB for typical non-IT personnel
Semantic Integration Challenges

Virtual common data model

1. Data integration and transformation?
2. Data model QA and use?

Tool A Data Model
- Electrical Plan
  - Tool Data
  - Cust_Signal
    - Address
    - Description
    - Value Range
    - Voltage
    - Digital/Analog

Tool B Data Model
- Function Plan
  - Tool Data
  - FB_Signal
    - Location
    - FB_Info
    - Value Defs
    - Input
    - Datatype

3. Tracing/Checking
   - Tool Data
   - Quality Engineer

Electrical Engineer
Machine Vendor
Customer
Process Engineer
Team Leader
Software Engineer
Semantic Integration Solution Concept

Virtual common data model

Tool A Data Model  Tool A Data Extract  Tool B Data Extract  Tool B Data Model

Engineering Data Base

2

Tool A Ontology

Cust_Signal
+ Address
+ Description
+ Value Range
+ Voltage
+ Digi/Analog

Tool B Ontology

FB_Signal
+ Location
+ FB_Info
+ Value Dets
+ Input
+ Datatype

Engineering Knowledge Base

1

Domain/Project Ontology

Common_Signal
+ Address
+ Description
+ Value Range
+ Voltage
+ ...

Support Point
+ location
+ Id
+ ...

Explicit Semantic Description of Common Engineering Concepts

Mappings between Common and Local Engineering Concepts

Advanced Applications (e.g., Checking Across Engineering Models)
Practical Example: End-to-End Analysis

Basic Analysis
List all elements used end-to-end
✦ supports the domain expert in plausibility checking
✦ can be supported with similarity analysis

WHERE {
el:E_short ekb:mapsTo ?Electric_ID.
?dom:Configuration dom:Config_ID ?Config_ID.
cfg:C_short ekb:mapsTO ?Config_ID.
?dom:Software dom:SW_ID ?SW_ID.
sw:S_short ekb:mapsTo ?SW_ID.
?dom:Software dom:SW_ID ?SW_ID.
}

Example Query Result
✦ (S1, “pressure”, “mbar”, C1, V_A, “pressure”, “mbar”)
✦ (S2, “level”, “cm”, C5, V_C, “level”, “m”)
**Advanced End-to-End Analysis**

**Assertion checks:** e.g., count of connections to a variable
- Variable: Exactly 1 connection
- Safety-critical variable: Exactly 2 connections

**Plausibility analysis**
- Basis for validation: similar concepts and compatible technical units
- Initial check of descriptions using *Natural Language Processing* (pressure sensor – pressure variable)

**Example Query Result**
- \((S_1, \text{“pressure”}, \text{“mbar”}, C_1, V_A, \text{“pressure”}, \text{“mbar”})\)
- \((S_4, \text{“pressure”}, \text{“mbar”}, C_3, V_B, \text{“temperature”}, \text{“degrees kelvin”})\)
- \((S_2, \text{“level”}, \text{“cm”}, C_5, V_C, \text{“level”}, \text{“m”})\)
Planned Research

Next Steps
- Explicit **engineering knowledge** model integration with three or more software tools, e.g., P&ID, EPlan, and logi.cad
- **Demonstrator**: Integration of a set of selected established software tools from industrial automation to demonstrate basic process automation and monitoring capabilities
- **Evaluation reports** on feasibility and effectiveness of prototypes with industry partners

Advanced Application of Semantic Web Technology
- Ontology-Supported Generation of Testcases
- Semantic Integration for Software Project Monitoring
- Ontology Alignment in a Safety-Critical Domain
Summary

- Semantic gaps between data models hinder process automation.
- **Semantic Integration** makes common engineering knowledge available explicitly and understandable to machines.

- **Common engineering knowledge** can be efficiently queried on project level
  - No need to learn tool-specific notations/syntax
- Exemplary application scenarios
  - Version management of engineering models across tools
  - End-to-End Analyses
  - Defect Detection Across Engineering Models
Backup Slides
Own Research Experience

- Dr. Thomas Moser
- Post-Doc Researcher CDL SE-Flex-AS
- http://www.ifs.tuwien.ac.at/moser
- Research Areas
  - Semantic Web / Ontologies in Software Engineering
  - Software Engineering for Automation Engineering
  - Simulation of Production Automation Systems
- Cooperations
  - logi.cals & Certicon (Christian Doppler Laboratory)
  - Siemens PSE (Software Process Improvement)
  - Frequentis (Semantic Dataintegration & Optimization)
  - JKU Linz; Systems Engineering & Automation; Prof. Grünbacher
  - TU Prag; Department of Cybernetics; Prof. Marik
  - Universidad Politécnica Valencia; PROS; Prof. Pastor
Comparison to Related Approaches

- **Usage of standards** in development processes
  - usage of standards (RUP, SysML) works well, if defined in an *early phase* of the project and if all *project partners adhere* to standard
  - hard to define and maintain standards for cross-domain engineering

- **Usage of common project repositories**
  - typical solution for modern data-driven tool integration which well solves the challenges of *persistency and versioning* of data
  - need to conform to a *common data schema*
  - many *existing tools already have a repository*

- **Complete Transformation** between project data models
  - seamless cooperation between project partners using well-known and established tools and notations
  - feasibility of this approach is *hard to verify* and *effort* required for establishing the needed transformations is *considerable*
Strengths and Weaknesses

- Perceived strengths of EKB-based approach
  - captures system configuration description in an **integrated model**
  - integrated model allows **tool support for QA checks** (reasoning to conduct consistency and plausibility checks among the model aspects)
  - provides a model that is directly usable both at **design and run time**

- Perceived weaknesses of EKB-based approach
  - Hard to provide **human readable visualization** of relationships among entities and a general overview on a domain
  - compared to traditional SE qualifications like UML, ontologies are a fairly new topic which **needs additional training or qualification** of the involved software engineers
Engineering Knowledge Base Architecture

- **EKB ontology**
  - Abstract description of components

- **Domain ontology**
  - General engineering concepts
  - Relations of the concepts

- **Tool ontologies**
  - Tool-specific concepts
  - Mappings between the tool ontologies and the generic concepts modeled in the domain ontology

- **Functions for transformation**
  - $T_1(\text{Electric.Config\_ID}) = \text{Configuration.Config\_ID}$
  - $T_2(\text{Configuration.SW\_ID}) = \text{Software.SW\_ID}$
Semantic Transformation Types

Supported Basic Transformation Types

- **Extraction of (parts of) data fields**
  - Only specific fields ("address") from a tool's signal data are needed

- **Aggregation ("Merge") of data fields**
  - Re-importing a signal's address (or receiving this address from another tools) requires merging several fields into one

- **Separation ("Split") of data fields**
  - A signal's hardware address consists of several parts contained in one string, i.e. "005.20.00.6.00" which are separate fields in other data models

- **String Conversion**
  - For re-uploading the signal data to the tools newline characters have to be replaced with tabs

- **Data type Transformation**
  - $T(\text{int}) = \text{float}; T(\text{string}) = \text{boolean}$

- **External Service Calls** (e.g., database queries)
Semantic Transformations

- Extraction
  - Only specific fields ("address") from a tool's signal data are needed

```groovy
<g:groovy executeOnElement="address">
  <g:script>
    <!--
    String nodeText = element.getTextContent();
    element.setTextContent(nodeText.substring(1, nodeText.length() - 1));
    -->
  </g:script>
</g:groovy>
```
Semantic Transformations

- **Aggregation**
  - Re-importing a signal's address (or receiving this address from another tool) requires merging several fields into one
    *Line 3 shows field names, line 5 and 10 (remaining text cut off) show an aggregation of single values (preceded by $ and limited by { })

```xml
1   <ftl:freemarker applyOnElement="$document">
2   <ftl:template><! --"Adresse";
3   "Datentyp";"symbolische Adresse";"Funktionstext";"Signal Nummer"
4   <list contentEntityList.securedContentEntities as entity>
5   "$entity.properties["componentNumber"]?left_pad(3, "0")}.${entity.properties["cpuNumber"]?left_
6   ["inputOutputModule"]?left_pad(1, "0").${entity.properties["channelName"]?left_pad(2, "0")}};
7   ";
8   ";
9   "$entity.properties["functionTextOne"]!!"}"
10  "$entity.properties["kks0"]!!"}$entity.properties["kks1"]!!"}$entity.properties["kks2"]!!"}
11  </list>-->
12  </ftl:template>
13  </ftl:freemarker>
```
Semantic Transformations

- Separation ("Split")
  - A signal's hardware address consists of several parts contained in one string, i.e. "005.20.00.6.00" which are separate fields in other tool data models.

```groovy
<g:script>
<Document doc = element.getOwnerDocument();
Element componentNumber = doc.createElement("componentNumber");
Element cpuNumber = doc.createElement("cpuNumber");
Element peripheralBoardAddress = doc.createElement("peripheralBoardAddress");
Element inputOutputModule = doc.createElement("inputOutputModule");
Element channelName = doc.createElement("channelName");
String all = element.getTextContent();
String[] parts = all.split("\.");
for (int i = 0; i < parts.length; i++) {
    switch (i) {
        case 0:
            componentNumber.setTextContent(String.valueOf(Integer.parseInt(parts[i])));
            break;
        case 1:
            cpuNumber.setTextContent(String.valueOf(Integer.parseInt(parts[i])));
            break;
        case 2:
            peripheralBoardAddress.setTextContent(String.valueOf(Integer.parseInt(parts[i])));
            break;
        case 3:
            inputOutputModule.setTextContent(String.valueOf(Integer.parseInt(parts[i])));
            break;
    }
}
</g:script>
```
Semantic Transformations

- **String Conversions**
  - For re-uploading the signal data to the tools newlines have to be replaced with tabs

```groovy
<groovy xmlns="groovy" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <script>
    <![CDATA[
      if (element == null) return;
      NodeList owner = (NodeList) element.getParentNode().getChildNodes();
      Element out = element.getOwnerDocument().createElement("keyValue");
      int first = 0;
      String id = "";
      String name = "";
      Element firstElement;
      Element secondElement;
      for (int j = 0; j < owner.getLength(); j++) {
        if (owner.item(j) instanceof Element) {
          Element transformerAtt = (Element) owner.item(j);
          if (first == 0) {
            id = transformerAtt
              .getTextContent().replaceAll("\n", "").replaceAll("\t", "");
            firstElement = transformerAtt;
            first++;
          } else {
            name = transformerAtt
              .getTextContent().replaceAll("\n", "").replaceAll("\t", "");
            secondElement = transformerAtt;
          }
        }
      }
    ]]>
  </script>
</groovy>
```
Application Scenario:
Process Data History

Challenges
- Large amount of unrelated data → hard to identify, correlate, and analyze
- No segmentation of process-related data (e.g., recipes) or maintenance data (e.g., diagnosis information)

Solution Approach
- Enhance process data with semantics and access rights
- Semantics contain data model, relations, and their meaning

Advantages
- Process history maintains data structure
- Well-defined user access levels
  - Process engineer → recipes
  - Maintenance engineer → diagnosis
  - Plant manufacturer or machine builder → remote maintenance
Application Scenario: Alarm Management

Challenges
- Alarm showers without priorities
- False positives, dependent alarms
- Alarm maintenance along plant life cycle (e.g., alarms from non-existent devices)

Solution Approach
- Link alarm data with engineering data (e.g., location, cause & effect matrix)
- Reason on the plausibility of an alarm based on plant data, previous alarms, and engineering data (e.g., maintenance protocols)
- Group alarms according to their cause (e.g., power failure alarms of several devices because of a broken fuse)

Advantages
- Ability to filter and prioritize alarms
- Reduce number of alarms
- Identification of the root cause of alarms
- Guide on the plausibility of alarms