Software Engineering Integration
For Flexible Automation Systems

Stefan Biffl
Christian Doppler Laboratory SE-Flex-AS
Institute of Software Technology and Interactive Systems (ISIS)
Vienna University of Technology
http://cdl.ifc.tuwien.ac.at
(Software+) Engineering for Automation Systems

- Increasing flexibility and complexity of software-intensive systems
- (Software+) engineering: the models of non-software engineers contain requirements and design constraints for software engineers.
- Surprisingly little work on the flexible, efficient, and robust integration of engineering tools across engineering disciplines.

Example: End-to-End Test Across Engineering Models

![Diagram showing integration of engineering tools](image)
Engineering Research Challenges

Basic research challenges
- Early defect detection across engineering discipline and tool boundaries.
- Engineering process analysis using design- and run-time data sources.

Research applications in the industry partners’ domains
- Platform to build integrated tools for automation systems development & QA.
- SCADA systems with data analysis for monitoring automation systems.
Integration Challenges and Requirements

Challenges from weak integration of software tools for engineering
1. **Engineering process** on event level is hard to track and analyze.
2. **Integration of software tools** is often vendor-specific and/or fragile.
3. **Sharing of data models across software tools** is inefficient and risky.
4. **Run-time defect detection** cannot easily access design knowledge.
5. **Integration of run-time environments** is hard to observe for analysis.

Challenges from weak integration of software tools for engineering
1. **Engineering process** on event level is hard to track and analyze.
2. **Integration of software tools** is often vendor-specific and/or fragile.
3. **Sharing of data models across software tools** is inefficient and risky.
4. **Run-time defect detection** cannot easily access design knowledge.
5. **Integration of run-time environments** is hard to observe for analysis.
Research Approach – Automation Service Bus

Goal: Approaches for the integration of software tools in automation engineering.

- Semantic Integration: Engineering Knowledge Base (3).
- Flexible integration of SCADA (4) with data analysis/simulation (5).
- Defect detection approaches for design time (6) and run time (7).
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

Signals (I/O)

Machine vendor catalogue

Tool A – Domain

Model Mapping

Tool A – Tool B

Derived Mapping

Common concept

Pump flow

Real (l/min)

0 to 1,200

%I20.5.3

Information
Analog
0 to 10 V
X.22.2.1
Semantic Integration of Engineering Tool Data

Current Issues on data integration from heterogeneous tools

Virtual common data model

Data integration and transformation?

Data model QA and use?

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

2. Tool A Data Extract

3. Tool B Data Extract
   - Tool B Data Model
     - Function Plan
       - Tool Data
     - FB_Signal
       + Location
       + FB_Info
       + ValueDefs
       + Input
       + Datatype

4. Machine Vendor
5. Customer
6. Process Engineer
7. Team Leader
8. Quality Engineer
9. Software Engineer

10. Electrical Engineer
Defect Detection Across Engineering Models

Use of common concepts in models across engineering disciplines

Defect type examples

- Missing, wrong, inconsistent model elements or relationships
- **Conflicts from changes** of overlapping model elements
- Run-time violation of model constraints

Defect detection approaches

- Review of overlapping model parts
- Automated check of model assertions (syntactic and semantic)
- **Change conflict detection** and resolution
- Derivation of run-time assertions
Practical Example: End-to-End Analysis

Example Query Result
- (S1, “pressure”, “mbar”, C1, V_A, “pressure”, “mbar”)
- (S2, “level”, “cm”, C5, V_C, “level”, “m”)


WHERE {
  el:E_short ekb:mapsTo ?Electric_ID.
  ?dom:Config dom:Config_ID ?Config_ID.
  cfg:C_short ekb:mapsTO ?Config_ID.
  ?dom:Configuration dom:Config_ID ?Config_ID.
  ec:C_key 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.
}
Combining Engineering Models and Run-Time Data

Challenges of working with run-time data
- **No engineering models** at run time → data hard to correlate and analyze
- **No security concepts** for run-time data (e.g., recipes, diagnosis information)

Semantic integration as solution
- Add knowledge on engineering data model to run-time data collection
- Define access rights to run-time data

Practical benefits
- Locate defects by correlating sensor data
- Prioritize local alarms in the plant context
- Secure access to sensitive data
  - Process engineer → recipes
  - Maintenance engineer → diagnosis
  - Plant manufacturer → remote maintenance
Summary – Christian Doppler Laboratory
Software Engineering Integration for Flexible Automation Systems

- **Close fundamental gaps** in scientific knowledge on
  - Defect detection across engineering disciplines
  - Defect detection using design- and run-time data
  - Engineering process analysis: find sources of defects.

- **Empirical studies** with industry partners on
  - Technical and semantic integration of software tools and data models.
  - Defect detection, location, and recovery approaches.

- **Applications**
  - Better integrated development and QA tools for industrial automation systems engineering with the “Automation Service Bus”.
  - SCADA systems with data analysis for monitoring automation systems to detect defects before they occur.

- Visit us online at http://cdl.ifc.tuwien.ac.at