

Engineering Environment Integration Across Disciplines with the Engineering Service Bus

Stefan Biffli

Dietmar Winkler

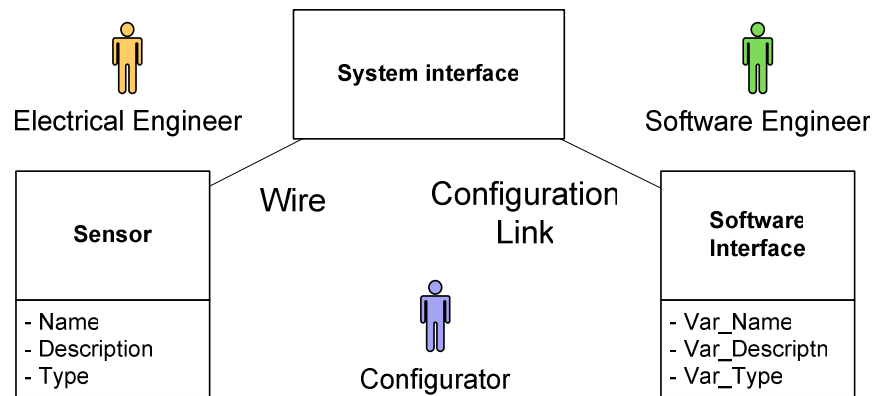
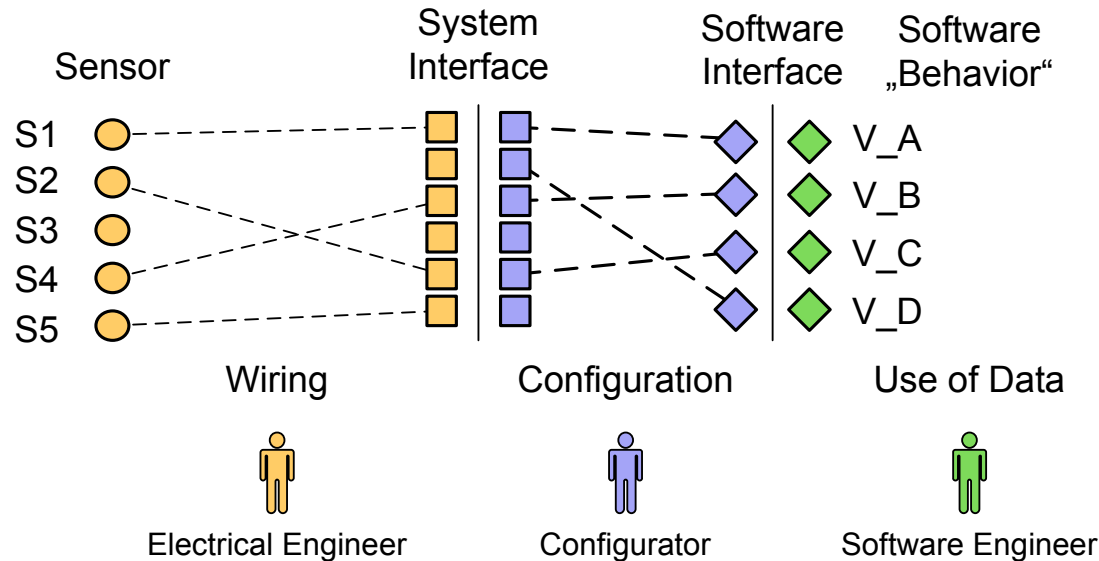
Institute of Software Technology and Interactive Systems (ISIS)
Vienna University of Technology



End-to-End Test Across Engineering Models



Use of common concepts in models across engineering disciplines



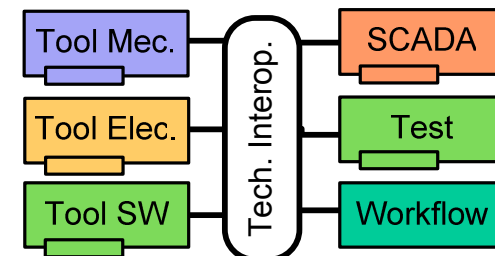
End-to-End Analysis

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

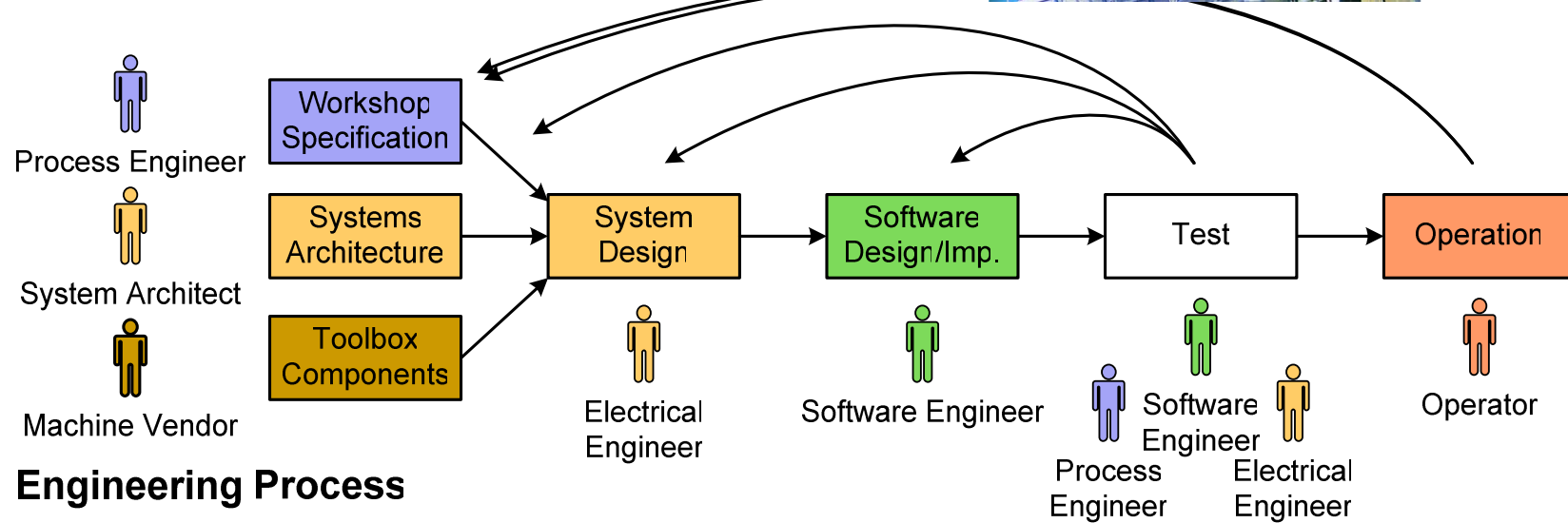
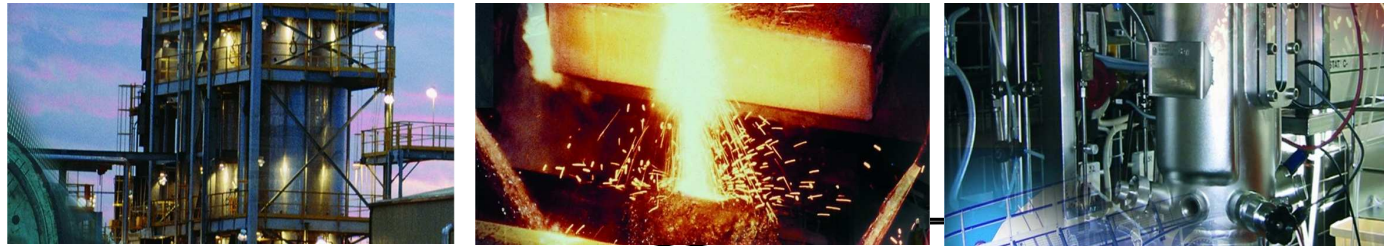
Motivation and Overview



- Observed **trends in software-intensive systems** development
 - **Increasing flexibility and complexity** of software-intensive systems
 - The models of non-software engineers contain requirements and design constraints for software engineers.
 - Many tools focus on supporting **one specific engineering role**.
- Surprisingly little work on the flexible, efficient, and robust **integration of engineering tools** across engineering disciplines.
 - Human experts bridge technical and semantic gaps between models and tools of different engineering disciplines.
- Goal: concept & tools for agile (software+) engineering environments.
- In this work we introduce the “**Engineering Service Bus**” (EngSB)
 - Based on **service-oriented integration concepts** successfully used for business software engineering systems.
- **Concept evaluation** based on real-world use cases and prototypes
 - Technical Integration of Tools
 - Semantic Integration of Data
 - Quality Assurance Across Engineering Disciplines
- **Christian Doppler Laboratory** starting January 2010



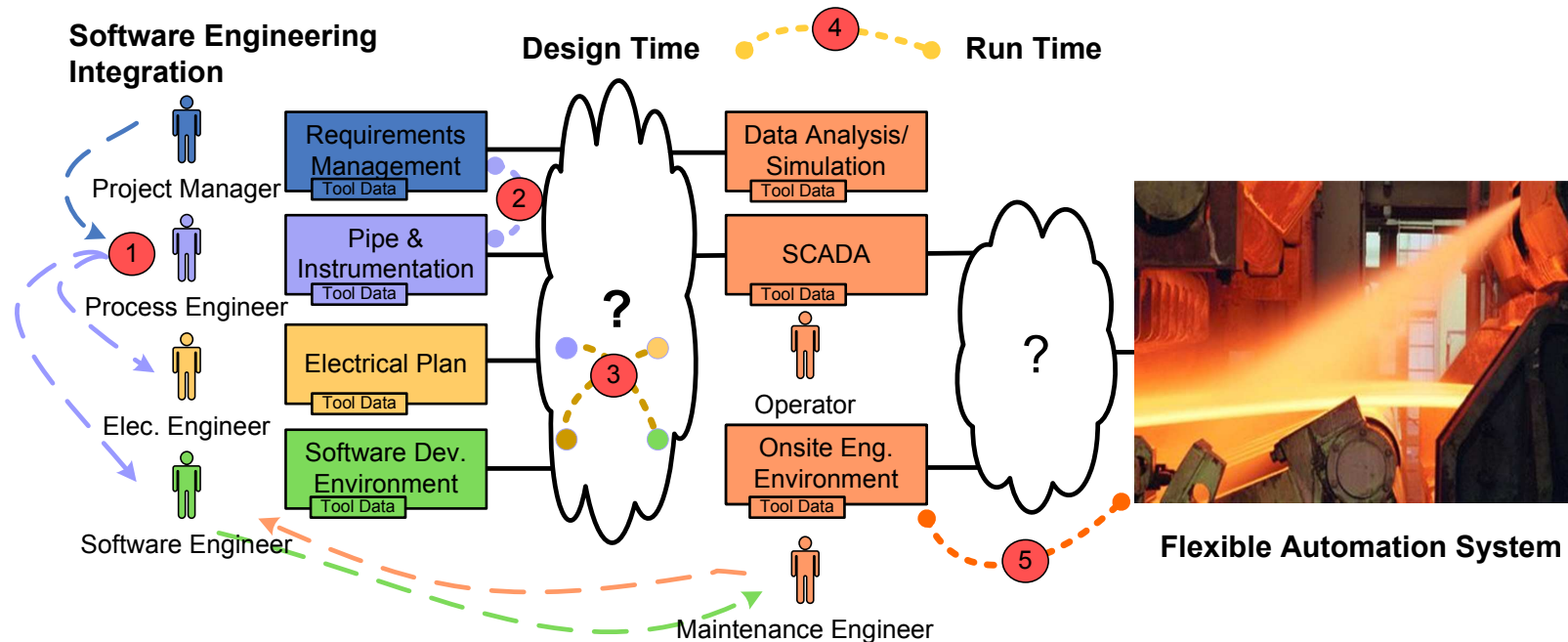
Engineering Research Challenges



Research challenges

- **Engineering model version** and change management.
- **Early defect detection** across engineering discipline and tool boundaries.
- **Engineering process analysis** to identify sources of defects.

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.

Industry partner requirements for integration solution

- Vendor- and platform-neutral; tailorable; incremental introduction process.

Engineering Environment Integration



Technical systems integration

- Mechanisms to exchange messages between systems that rely on different platforms, communication protocols and data formats

Technical systems **integration issues**

- Coupling of tools and data
- Coupling of business logic and communication logic
- Vendor-specific integration

Conversation styles (decision who actually gets called)

- Service-oriented: request/reply pattern (synchronous)
- Event-driven: complex event processing (high flexibility and decoupling)
- Process-driven: workflow (long-running development processes)

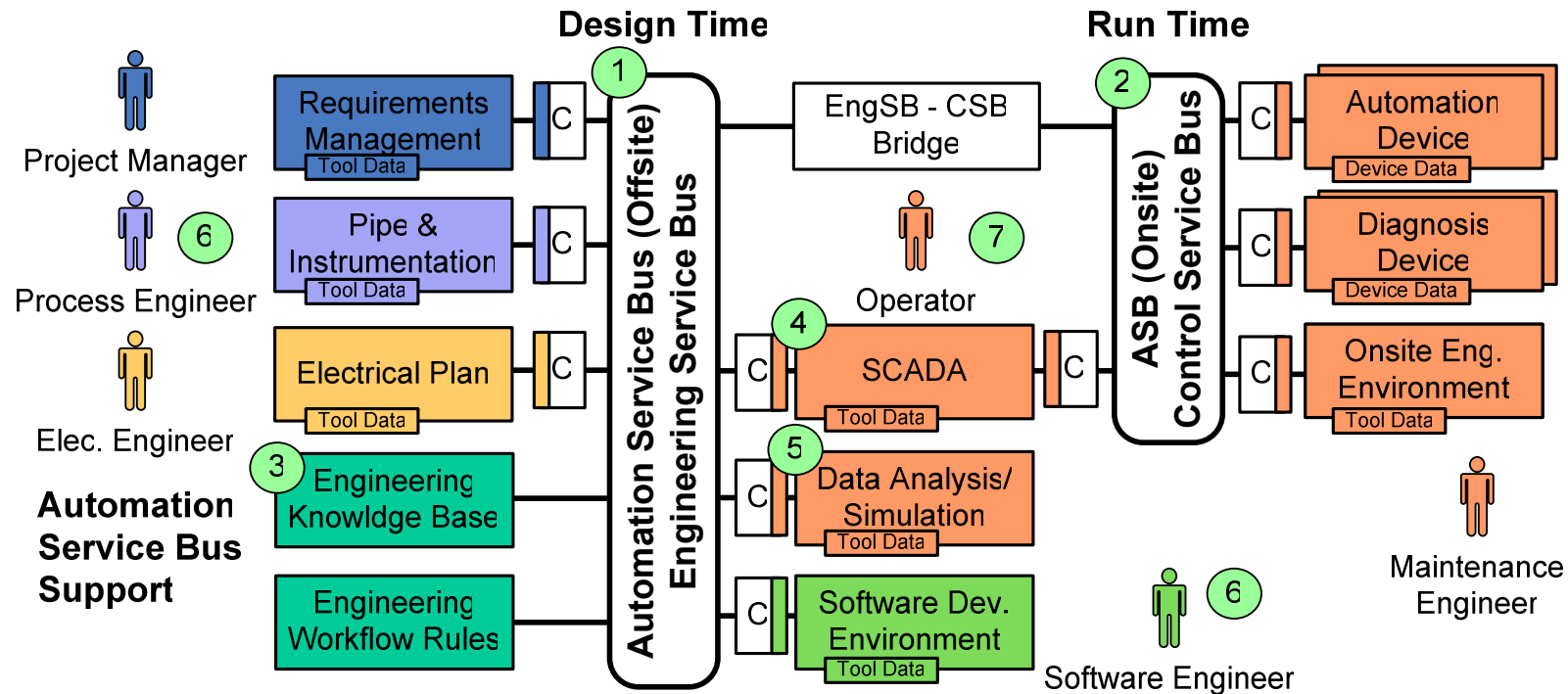
Neutral **Enterprise Service Bus** (ESB) concept in computing centers.

- Tailorable to project context & incremental introduction process
- Needs adaptation to (software+) engineering environments, e.g., better resource efficiency and offline capabilities.

Research Approach – Automation Service Bus



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

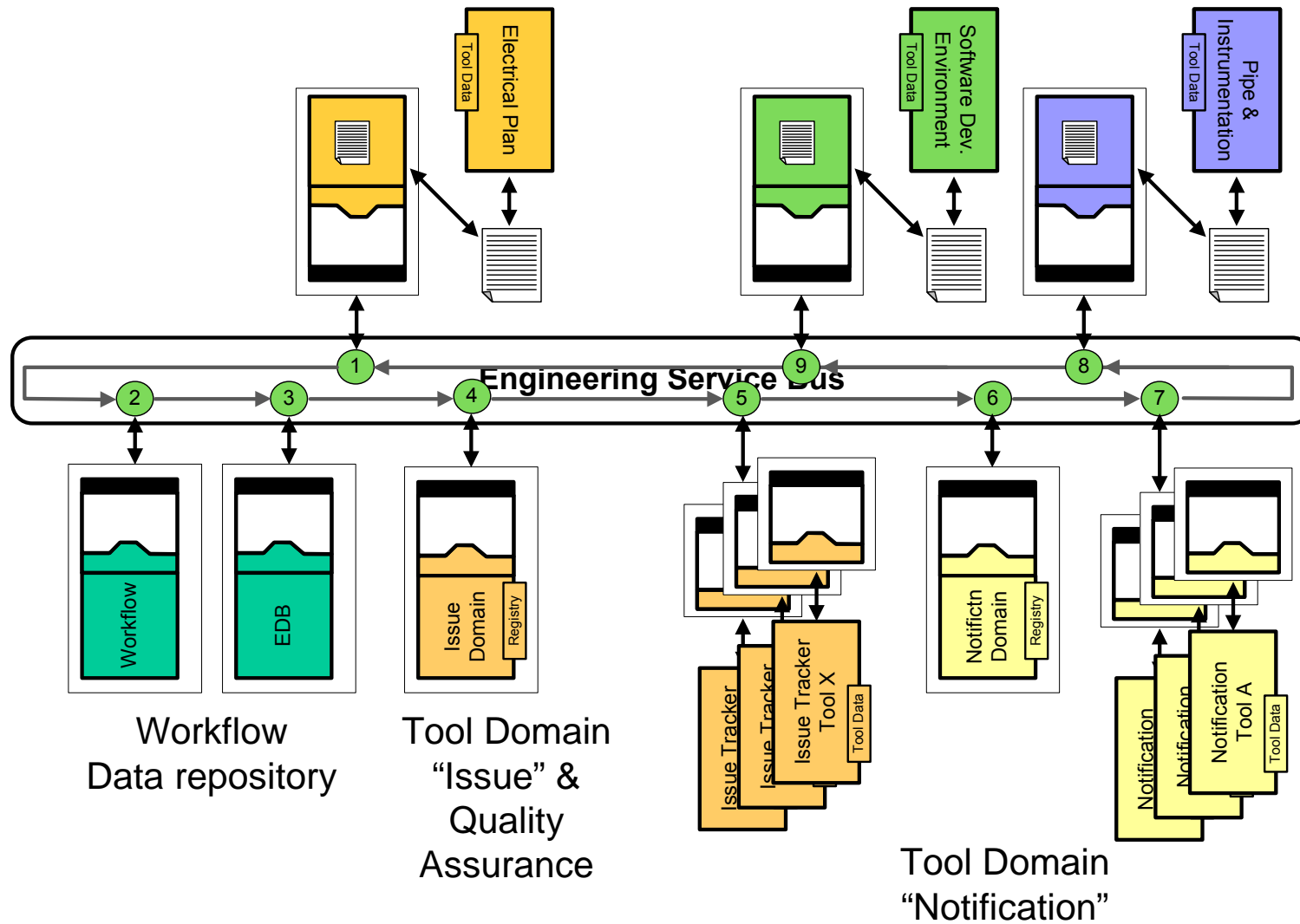


- Technical Integration: Engineering Service Bus (1), Control Service Bus (2).
- Semantic Integration: Engineering Knowledge Base (3).
- Flexible integration of SCADA (4) with data analysis/simulation (5).
- Defect detection approaches for design time (6) and run time (7).

Example: Technical Systems Integration



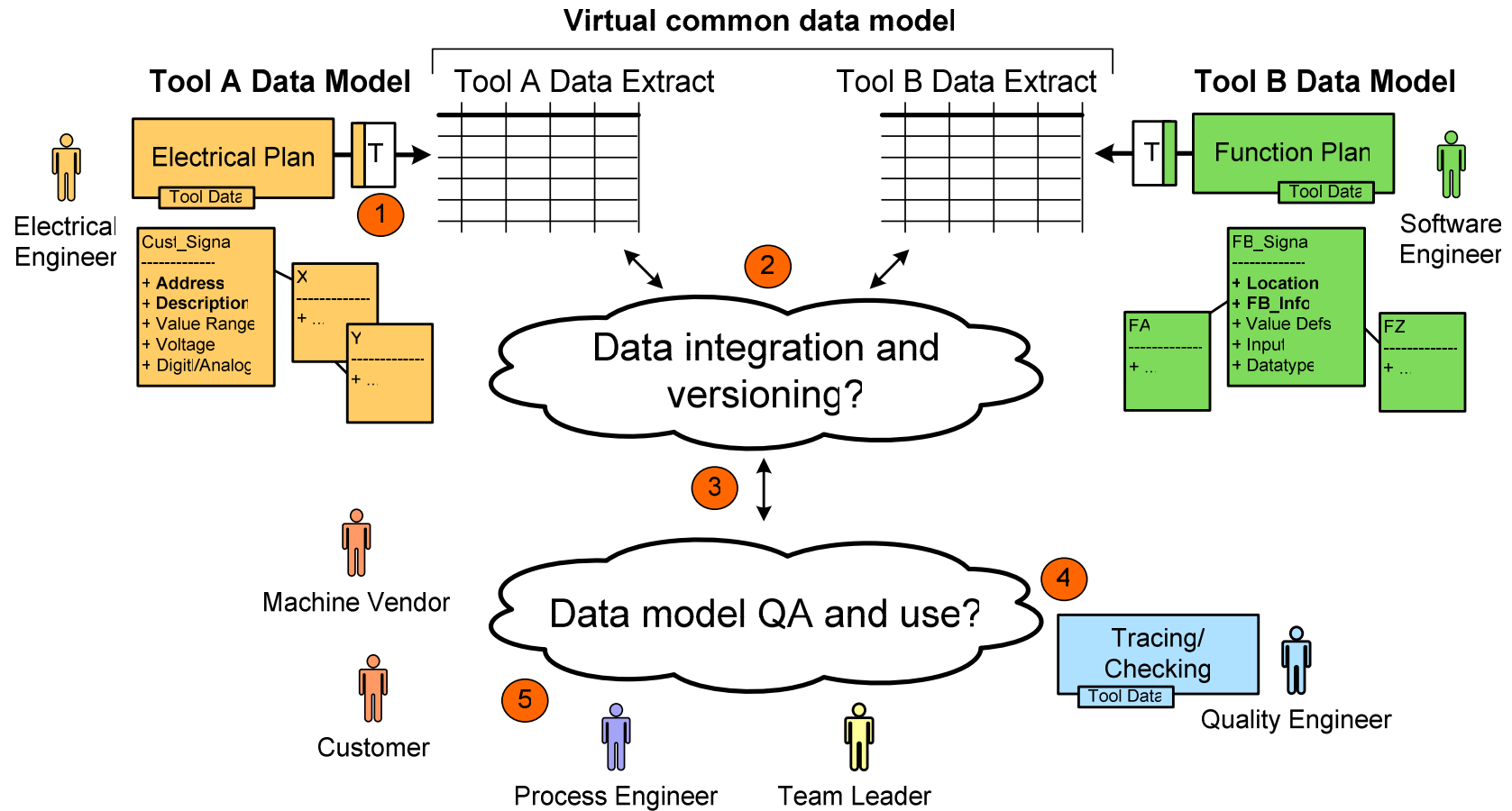
Itinerary-based work flow example of 3 heterogeneous engineering tools



Semantic Integration of Engineering Knowledge



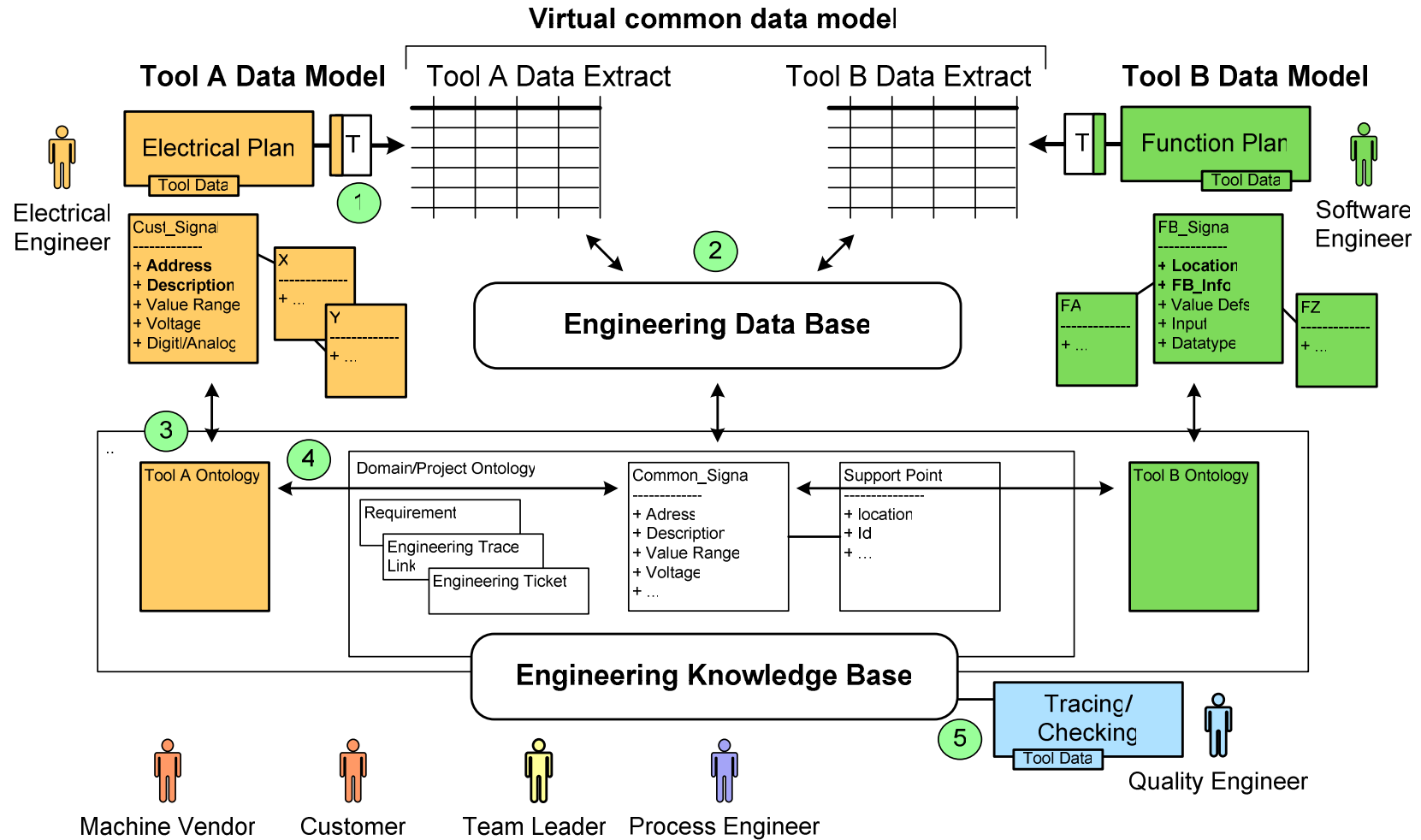
Current Issues on data integration from heterogeneous tools



Semantic Integration of Engineering Knowledge



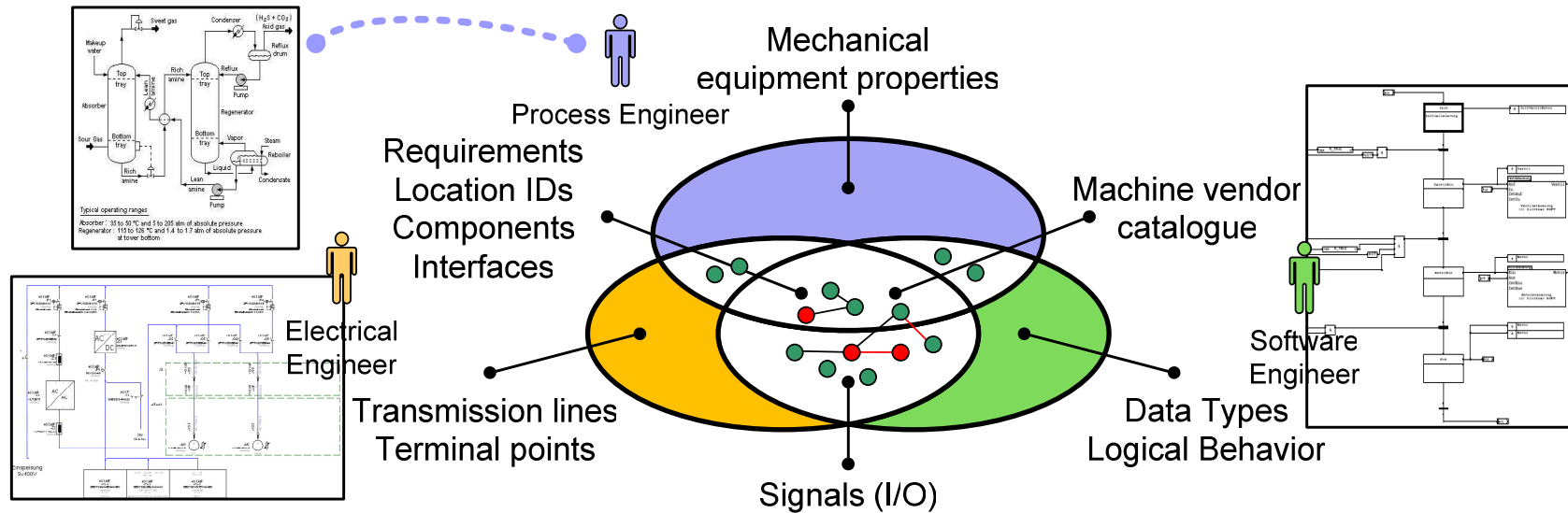
Planned Work and Research Goals



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

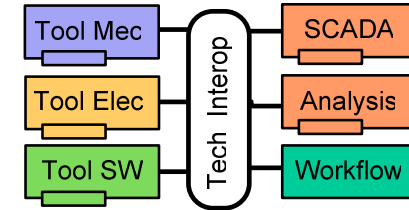
- Review of overlapping model parts
- Automated check of model assertions (syntactic and semantic)
- **Change conflict detection** and resolution
- Derivation of run-time assertions

Summary – Christian Doppler Laboratory

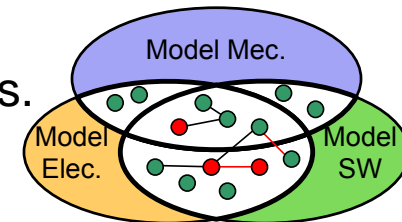


Software Engineering Integration for Flexible Automation Systems

- **Close fundamental gaps** in scientific knowledge on
 - Defect detection across engineering disciplines
 - Defect detection using design- and run-time data
 - Engineering process analysis: 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 quality assurance tools for industrial automation systems engineering.
 - -> Relieve human experts from tedious and error-prone administrative activities between tools in the engineering process.