Efficient Automation Systems Engineering Process Support
Based on Semantic Integration of Engineering Knowledge

Thomas Moser, Richard Mordinyi, Dietmar Winkler, Martin Melik-Merkumians, Stefan Biffl
Christian Doppler Laboratory ”Software Engineering Integration for Flexible Automation Systems”
Vienna University of Technology, Favoritenstrasse 9-11/188, 1040 Vienna, Austria
firstname.lastname@tuwien.ac.at

Abstract

Modern industrial automation systems engineering (ASE) environments have to accommodate for heterogeneity coming from the engineering disciplines involved, the software tools and their data models, and run-time data collection. In many ASE environments domain experts have to invest considerable effort to bridge the semantic gaps between common project-level engineering concepts and the diverse local data representations.

In this paper we discuss the needs for semantic integration and applications of machine-understandable knowledge engineering in three real-world ASE use cases from our industry partners. We provide an evaluation concept with empirical studies to measure the benefits and limitations of the proposed approach compared to the traditional expert-intensive approach. Major result of the initial evaluation is that semantic integration has good potential to make engineering processes more efficient and robust if supported well with user interfaces that end users find usable and useful.

1 Introduction

Modern industrial automation systems engineering (ASE) projects are large, complex and iterative, and often involve several heterogeneous engineering disciplines [13]; therefore the configuration of the engineering process must be tool-supported to ensure efficiency and correctness. ASE environments have to accommodate for heterogeneity coming from the engineering disciplines involved, the software tools and their data models [15], and run-time data collection [14]. In many ASE environments domain experts have to invest considerable effort to bridge the semantic gaps between common project-level engineering concepts and the diverse local data representations [10]. Semantic integration, i.e., the explicit and machine-understandable description of engineering knowledge, allows for data integration across tool and domain boundaries, as well as for automated Quality Assurance (QA) techniques [7].

In this paper we discuss the needs for semantic integration and applications of machine-understandable knowledge engineering based on three real-world ASE use cases. In the pre-engineering phase, engineering knowledge is needed to integrate heterogeneous engineering tools and to configure such integrated engineering environments, e.g., multi-disciplinary engineering projects used for Signal Change Management in power plant engineering. In the engineering phase, engineers and management personnel want to query the integrated engineering knowledge across engineering tools and their data models, e.g., to be able to perform project wide monitoring independent of individual engineering tools. In the run-time phase, engineering knowledge is required to configure integrated access to engineering knowledge and run-time data, e.g., to perform condition monitoring, support autonomous and flexible system reactions and to better detect and localize defects. We introduce a framework for semantic integration in the ASE life cycle and provide an evaluation concept with empirical studies to measure the benefits and limitations of the proposed approach compared to the traditional expert-intensive approach. In addition, for each of phases of Semantic Integration, we derive initial research issues to be addressed in further research and therefore try to position the research area of Semantic Integration for ASE projects in the scientific community. Major result of the initial evaluation is that semantic integration has good potential to make engineering processes more efficient and robust if supported well with user interfaces that end users find usable and useful.

The remainder of this paper is structured as follows: Section 2 summarizes related work on Automation Systems Engineering and on data integration. Section 3 introduces the use cases, while Section 4 describes the three phases of Semantic Integration in ASE, discusses the findings and presents initially derived research issues of the Semantic Integration research area. Finally, Section 5 concludes the paper and identifies further work.

2 Related Work

This section summarizes related work on Automation Systems Engineering (ASE)and on data integration.
2.1 Automation Systems Engineering

Automation systems engineering (ASE) projects typically involve heterogeneous environments, where engineers from different disciplines, e.g., mechanical, electrical and software engineering disciplines, collaborate and interact with each other [3]. Major challenges arise from this need for collaboration across disciplines as individual engineers apply domain specific tools based on domain-specific data models within domain specific and isolated environments, e.g., for constructing the mechanical layout (mechanical engineer), electrical circuit plans (electrical engineers), software models based on UML, and function plans for implementing functional control software (software engineers).

Integration of engineering systems is a challenge as (particularly in the automation industry) typically a broad range of engineering tools from different vendors are used to solve specific problems [12]. Tools within one vendor are sometimes integrated to exchange data, but hardly between vendors. APIs and exchange formats often do not follow established (open) standards. Therefore the AutomationML [15] project provides a standardized XML data exchange basis for data integration between multi-vendor automation systems engineering tools as foundation systematic information exchange between engineering models.

The Automation Service Bus (ASB) provides a middleware platform [2][3] that enables defect detection in overlapping areas of data models, i.e., common attributes of individual disciplines, based on common concepts in a virtual common data model [7]. Semantically integrated data models can help detecting defects (a) in pre-engineering phases (i.e., project configuration and setup), (b) during the engineering phase (i.e., automation systems development in distributed heterogeneous environments), and (c) during run-time (i.e., linking run-time data to engineering objects for maintenance and condition monitoring purposes.

2.2 Data Integration

The fundamental reason that makes semantic heterogeneities between data schemas so hard to address, is the independent origin of data sets using varying structures to represent the same (or overlapping) concepts [1]. From a practical perspective, one of the reasons why schema heterogeneity is difficult and time consuming is that the solution requires both domain and technical expertise: a domain expert who understands the domain meaning of all schemas being reconciled and technical experts for writing transformations [11].

Global-as-View [17] is an information integration concept, which specifies relations of a global data schema as views on related local schemata. One specific implementation of the Global-as-View concept is the Engineering Database (EDB). The EDB [10] consists of (a) the data integration approach to align semantically heterogeneous tool-specific data models to a common domain model, (b) the version management approach based on data integration, and (c) the architecture and prototypical realization of tool support for data integration and version management. The EDB uses a so-called Virtual Common Data Model (VCDM) for modeling the exchanged information. The Engineering Knowledge Base (EKB) [7] is a layered semantic model using ontologies, which builds on the Global-as-View concept [17][10] and explicitly models the engineering concepts and mappings using machine-understandable syntax.

Currently the dominating communication standard for plant data acquisition system is OPC Classic [14]. The disadvantage of the OPC Classic Interface is that it consists of several different interfaces, using different address spaces and mechanisms for the data acquisition points. The most important interfaces are Data Access (DA) for actual process data, Historical Data Access (HDA) for historical process values, and Alarms & Events (A&E) used for alarming mechanisms. OPC Unified Architecture (UA), the evolution of OPC Classic, unifies all these different interfaces into one single interface, with one address space. OPC UA follows a service- and object-oriented approach. Each element in the OPC UA address space, called node, contains its value and meta-data (e.g., measurement unit, measurement range, location, etc.). Additionally each node contains links to related nodes of the plant. Therefore instead of the simple tree-hierarchy structure of OPC Classic, the nodes can be arranged in any possible structure up to a fully meshed network [18]. OPC UA node can also contain methods, which can be evoked by other OPC UA nodes.

3 Use Cases

This section describes the three use cases used for the presentation of the three different phases of Semantic Integration in ASE projects. As shown in Figure 1, the Automation Service Bus (ASB) consists of two sub-busses: on the left hand side of the figure, the Engineering Service Bus (EngSB) is used to connect multi-engineering tools...
such as EPlan, OPM or Mantis, and additionally provides internal components such as the Engineering Knowledge Base (EKB) or the Engineering Data Base (EDB), as well as interactive components such as the Engineering Cockpit. On the right hand side, the Control Service Bus (CSB) is used to connect real-time field devices such as PLCs or HMIs, and additionally provides interfaces to real-time data sources such as OPC UA. In order to bridge the two sub-busses, and therefore to form the overall Automation Service Bus (ASB) framework, there may on the one hand exist multi-engineering tools such as SCADA that are capable of interfacing with both the EngSB and the CSB at the same time, and on the other hand side there are specific internal ASB components that provide similar functionality.

3.1 Signal Change Management

Signals represent one of the base artifacts in the course of developing power plants and reflect different hardware components. Depending on the size of the commissioned power planed there are about 40 to 80 thousand signals to be managed, while around 25% of all signals change during the development. Moreover, late changes across disciplines and tool borders make projects more risky and error prone if not considered by related disciplines [19]. In such multi-disciplinary engineering projects system integrators want to be able to conduct automated change management across all tools that contribute project-level data elements regardless of the origin of the tool and data model. For instance, there is the need of cooperation between tools like EPlan1 representing electrical plans, OPM expressing control programs, CAEx protocol based applications for editing function block diagrams, a ticketing system like Mantis2 for tracking work progress, or source code management environments like github (refer also to tag number 1 in Figure 1). Apparently, today’s integrated tool suites often consist of a pre-defined set of tools and a homogeneous common data model, which work well in their narrow scope but do not easily extend to other tools in the project outside the tool’s scope.

3.2 Engineering Cockpit

One of the main challenges in multi-engineering projects is project monitoring and management. They are accompanying key activities of project managers along the project course to (a) keep track of the project progress and (b) to take countermeasures in case of deviations with respect to time, budget, and quality [4]. Thus, measuring and analyzing project data is the foundation for process observation and decision making processes.

The so called the Engineering Cockpit [9] is a social-network-style collaboration platform for automation system engineering project managers and engineers and has been introduced to provide a role-specific single entry point for project monitoring, project collaboration, and project management. As described before, information provided by engineering tools is scattered over heterogeneous data sources. As current tools suites are not well integrated, they do not support fully automated reporting. Therefore, costly and error-prone manual collection and aggregation of reporting data is required. Consequently, project progress reports are made seldom or on-request only. Project managers would welcome monitoring tools allowing them to overview project progress between milestones, to overview whether planned activities are worked on, or to identify hotspots of inefficiency where activity level differs considerably from expected level.

3.3 Run-time Defect Detection

The detection of run-time failures is an important topic in industrial automation. Undetected failures can lead to waste of production material or at worst case to the drop-out of whole production batches. Depending on the type and number of sensors failures can be easy or hard to detect. For example valves are usually not directly monitored, so the break-down of a valve, rendering it unable to open or close, will not issue an event or alarm. The consequences the valve failure will only be seen on the following equipment, as the behavior of the rest of the plant will be independent of the desired valve state. The semantic integration of engineering knowledge gives us a powerful new technique for efficient design of run-time defect detection systems.

A major obstacle is the extraction and combination of the required runtime data to perform such analysis. OPC UA, a basic component of the CSB, allows a model-based access to runtime data. Not only the interconnections between the different runtime data can be modeled by OPC UA, also the meaning of the connection can be saved in the OPC UA configuration, leading to a sufficient complete runtime plant meta-model. Based on this plant meta-model the runtime fault detection rules can be generated and evaluated. The generation of the OPC UA configuration is done at engineering time through combining the different engineering facts distributed over the different tools.

4 Semantic Integration Phases in ASE

This section describes the three phases of Semantic Integration in ASE based on three real-world ASE use cases.

4.1 Pre-Engineering Phase

In the pre-engineering application phase the EKB facilitates the usage of modeled engineering knowledge to derive configurations (e.g., transformation instructions) for integrated engineering environments. However, the configuration of heterogeneous tools with respect to the number of involved engineering disciplines is complex, needs engineering expertise for every (re-)configuration, and is therefore inefficient and error prone. Our proposed approach for derivation of configurations for engineering

---

1http://www.eplanusa.com/
2http://www.mantisbt.org/
projects requires the elicitation of expert knowledge on heterogeneous tools and their data models into a machine-understandable engineering knowledge base, e.g., an ontology facilitating an efficient and semi-automated approach.

As shown in Figure 2, the pre-engineering phase process consists of the following five process steps:

1. **Identification of VCDM concepts**: in the first step, engineering domain experts have to discuss about engineering concepts common in every domain. It requires them to sketch their knowledge and agree on a common engineering concept and their properties by indicating which data fields may be relevant for other engineering disciplines [3]. It is common in large engineering projects to use different expressions for the same concept. However, the process step only requires identification of common concepts, their structure and properties [10][7]. With respect to the given use cases domain experts need to agree on the definition of a "signal" and its elements (e.g., KKS as primary key, channel identifier, or the interface unit control board) and properties (e.g., units like watt, kilo- or mega-watt) common in every domain.

2. **Modeling of local tool concepts**: the second process step requires the modeling of tools (e.g., EPlan, OPM, Mantis, OPC UA) used in the engineering project without taking into account other tool concepts or the already defined common concepts in the VCDM [10][7]. Typically, this kind of information already exists in e.g., UML class models or EER diagrams which can be used to facilitate modeling.

3. **Mapping of local concepts to VCDM concepts**: the third process step deals with the specification of the way by which local concepts can be mapped onto common concepts defined in the VCDM [7][8]. This means that the mapping specifies which tool specific data elements refer to which common concept.

4. **Modeling of transformers**: the fourth process step identifies and models existing transformations used in already implemented transformation scripts or standards, and needed to execute the conversion between local concepts and VCDM concepts and vice-versa as specified in the mappings [8].

5. **Derivation of transformers**: once transformers have been modeled, the fifth process step derives specific executable transformation instructions according to the engineering projects’ requirements [7][8]. With respect to the described use cases, derived transformation instructions are written according to smooks framework3 specifications.

### 4.1.1 Research issues

With respect to the presented process steps (1-5), we initially derived a set of research issues (1.x-5.x) which will be briefly discussed in the following:

**RI-1.1**: In order to support domain experts with the identification of VCDM concepts, tool-support is required to automate time-consuming and error-prone manual process steps. Therefore, we will create domain-specific tool-support, which will be evaluated with domain experts from our industry partners regarding feasibility, effectiveness and usability. In addition, we will perform an empirical evaluation in order to measure the effort savings when using the proposed tool-support.

**RI-1.2**: The degree of acceptance mainly depends on the fact how already existing data models may be reused and integrated into the VCDM. In a scenario-based validation and verification step it has to be shown that the process support facilitates the effective and automated transformation of established standards and protocols (e.g., AutomationML) into the common data model.

**RI-1.3**: As standards are always used with respect to a specific domain scenario, it is therefore necessary to find a way of describing those scenarios and thus allowing automated validation and verification of the transformation with regard to completeness and consistency.

**RI-2.1**: Typically, local tool concepts are already described in some way by means of e.g., UML- or EER diagrams. The effective and automated transformation and integration of already captured domain-specific engineering knowledge into the common data model needs to be enforced by additional tool-support. As described in RI-1.3, scenario-based validation and verification steps will be used to ensure completeness and consistency between existing data models and the transformed ones.

**RI-3.1**: The larger an engineering project or the more engineering disciplines are involved in such projects, the complex is both the common data model and the mappings between local and common data models. In order to minimize the number of error sources and human effort ontology alignment methods have to be investigated which semi-automatically help identifying similarities between concepts and automatically create suitable mappings for transformations between those suggested candidates.

**RI-3.2**: Beside the VCDM, mappings between local data models and VCDM of high quality are the linchpin for efficient coordination of engineers from different disciplines of an engineering project. Therefore, aspects of quality assurance at communication and coordination level are necessary which check and verify according to

---

3http://www.smooks.org
defined project scenarios whether it is possible to transform correctly between two tools from different domains at concept level with the available set of mappings representing data transformations.

**RI-4.1:** Similar to engineering projects already having captured knowledge about tool concepts, there are also existing integration solutions supporting data exchange between tools from different disciplines. The challenge is the way of modeling already written transformation scripts so that they can be included into the process of transformation derivation.

**RI-4.2:** Nevertheless, it has to be investigated which are the most basic transformation types that the EKB should provide out of the box in order to support reuse of transformations between engineering projects. Furthermore, by using the described common and local data models automated test case generation supports testing of transformations between tools from different domains at instance level.

**RI-5.1:** The suggested approach introduces changes in engineering project management processes as it requires from domain experts the definition of a common concept across engineering disciplines. Based on empirical evaluations it has to be investigated how much effort has to be invested into the entire process resulting in an automated derivation of transformer configurations compared to manual transformer configuration. This includes the comparison of initial effort, effort in case of changes or adaptations (e.g., when introducing similar projects) with traditional approaches by taking into account the size and complexity of the engineering project.

**RI-5.2:** As changes are introduced into current engineering project management processes, the effectiveness of the proposed approach has to be investigated as well by identifying defect classes, potential sources of errors, and tool-support capabilities by comparing it with the traditional way of transformer derivation.

### 4.1.2 Discussion

In the pre-engineering phase of an automation systems engineering project, QA focuses on verifying and validating (a) project configurations, i.e., the technical setup of required tools and data models, and (b) engineering processes (sequence of engineering steps for collaboration), which could not be measured without an integrated engineering support approach [19]. A common question is whether an engineering process behaves like expected. Data capturing during configuration testing enables detection of product and configuration defects. The availability of machine understandable project and process data, provided by the ASB implementation represents the foundation for QA during pre-engineering and enables defect detection in the project configuration (e.g., due to automation supported reasoning) and verification of the required tool domains, tools, and data models in an automation-supported way (by integrated test scenarios).

![Figure 3. Process Steps of the Engineering Phase.](image)

### 4.2 Engineering Phase

In the engineering application phase, the EKB is used for queries to engineering knowledge across software tools and their data models. Today, QA (e.g., model checking) of data models across heterogeneous engineering tools takes considerable expert effort. Our proposed approach elicits the expert knowledge on heterogeneous tools and their data models for the support of queries on project-level, i.e., VCDM level, concepts that are mapped to local engineering tool concepts for the resolution of the queries used for QA.

As shown in Figure 3, the engineering phase process consists of the following four process steps:

1. **Identification of Project Metrics:** in the first step, the engineering project manager has to clarify metrics help him/her monitor the progress of the engineering project. Metrics with regard to a single engineering domain may be covered by already established tool environments of that domain as today’s integrated tool suites work well in their narrow scope. The challenge is to find project metrics taking into account the entire spectrum of engineering data provided by different engineering disciplines rather than each of them one by one. The power of the Engineering Knowledge Base (EKB) is that it supports queries over the combination of domain-specific queries by facilitating abstractions of heterogeneous concepts.

2. **Definition of Query to the VCDM:** once project relevant metrics for effective decision making have been identified, the second process steps requires the definition and assembly of the query at concept level. Typically, queries are defined in a ”report-driven” integration style, i.e. queries for metrics are created ad-hoc as multiple heterogeneous databases have to be queried and retrieved information aggregated manually. The limitation of such traditional approaches is that it takes a lot of skills to overview and human effort and time to manage the complexity of multiple logically interconnected databases.

3. **Execution of the Query:** in the third process step the defined query is executed in the Automation Service Bus environment. Queries may be executed if general or detailed information about project progress status is re-
quired, or if checks on data exchanged between the engineering disciplines have to be performed. The latter enables to run plausibility checks on data both to be processed and already available in the Engineering Database. This makes sure that only valid data entries are exchanged between engineering domains.

4. Presentation of the Results: the fourth process step is concerned with the proper way of presenting gathered information according to the defined metrics [9]. With regard to the third processing step, visualization is not only concerned with the proper presentation of project progress, but also requires means to properly inform the engineer in case plausibility checks have failed or conflicts have been detected.

4.2.1 Research issues

With respect to the presented process steps (1-4), we initially derived a set of research issues (1.x-4.x) which will be briefly discussed in the following:

RI-1.1: For project managers it is essential to be kept informed about project progress based on certain set of relevant information. The presented data supports decision making as it may point out limitations, hot-spots in the project. However, such information is based on historic data. The Engineering Knowledge Base uses ontologies to capture concepts which queries are defined on. It needs to be investigated how mechanisms built in ontology-based technologies support the definition of queries which based on historical data try to reason about future data (e.g., project progress).

RI-2.1: Typically, the definition of queries executed over multiple engineering domains requires from experts knowledge about every domain affected. The EKB abstracts domain-specific concepts by introducing mappings between them. Nevertheless, due to the high number of concepts and failure rate, an effective tool-supported way for query definition for domain experts is needed. This way the reuse of queries between projects may be increased and complexity compared with e.g., SQL minimized.

RI-3.1: The efficient execution of queries requires investigation regarding performance and scalability. It needs to be evaluated at which point the EKB and the EDB have to be distributed in order to still respond to high load quickly.

RI-4.1: With respect to defined metrics, it needs to be investigated how complex structures can be visualized best.

4.2.2 Discussion

During engineering project execution, a major goal of project managers focuses on project monitoring and control based on observed product and process metrics. Captured process events enable the observation of engineering processes in an ASE development environment. In addition appropriate metrics, e.g., number of changes per time interval and/or project phase enable reasoning on the project progress and the product quality. The Engineering Cockpit (EC) summarizes the results of process step observation, product progress, and quality as an important specific view from project management perspective. In addition various views on engineering data provide observed related data and drive engineering use case based on related perspectives, e.g., an electrical engineer might be interested the number of changes and open tickets assigned to him/her. Based on the individual role (and EC view) conclusions on the current project state can be drawn to identify defects in engineering artifacts more efficient and effective.

4.3 Runtime Phase

Following the engineering phase, the first task in the runtime application phase is to configure and initialize the data acquisition system. The runtime data model and therefore the configuration of the acquisition system (OPC Classic, OPC UA, etc.) can be derived from the engineering model, which is usually done manually. Automatic detection of non-directly observed plant equipment breakdown or the starting fault of a failure mode is currently not possible, as runtime data and engineering plans are not interconnected in the running system.

As shown in Figure 4, the runtime phase process consists of the following seven process steps:

1. Identification of Relevant Runtime Failure Types: in the first process step, domain experts have to discuss and identify the relevant runtime failure types that should be detectable using the proposed approach [6]. Based on their experience, the domain experts select the runtime failure types that by now are not easy to detect using traditional methods. A Failure Mode and Effect Anal-
lish (FMEA) is performed to identify the severity and the probability of the identified failure sources. Based on the FMEA the faults are selected, which will probably cause more financial losses if undetected, as the additional engineering costs for the detection system are.

2. Identification of Engineering Models Relevant for Failure Detection: engineers, who want to detect failures at runtime which can not be compassed by analyzing singular sensor values or failures at sensor-less components (e.g. broken actuators which are not monitored by a sensor), need information from software models that reflect dependencies between components at design and runtime, e.g., the workshop layout, recipes and production procedures. Therefore a Failure Tree Analysis (FTA) is performed on the selected runtime failures. By that the relevant engineering artifacts which participate in a given error are identified.

3. Integration of Engineering Models and Runtime Data Models: by now, domain and software experts are needed to integrate the fragmented views (e.g., propagating model changes into other models, cross-model consistency checks) from these models, which often is an expensive and error-prone task due to undetected model inconsistencies or lost experience from personnel turnover. We propose to use the EKB [7] to provide a better integrated view on relevant engineering knowledge in typical design-time and runtime models, which were originally not designed for machine-understandable integration [5].

4. Definition of Rules: based on the integrated knowledge in the EKB, rules for several failure classes are derived manually by domain experts. These rules are constructed as If-Then-Else clauses, as If-Then-Else rules are natural to human reasoning and understanding [16].

5. Runtime Data Collection: additionally the runtime data acquisition has to be configured. This is accomplished by configuring the OPC UA servers in the plant. These configurations are in fact runtime plant models, which are parameterized with the equipment measure point settings. Therefore the runtime data can be accessed in a model-based way, whereas this model corresponds to the engineering plant model [14].

6. Evaluation of Rules: in the running system the failure detection occurs based on the rules defined in step four. The rules are used as runtime assertions which are evaluated every time new runtime information is imported into the EKB. The results of this evaluation can either be specific and personalized notifications of operators (7a. Notification of Operators) or autonomous reactions to failures (7b. Autonomous Reaction to Failures).

4.3.1 Research issues

With respect to the presented process steps (1-7), we initially derived a set of research issues (1.x-7.x) which will be briefly discussed in the following:

RI-1.1: Often historical failure information is useful for identifying important types of runtime failures, therefore further investigation on the usage of historical failure information to support domain experts in the identification of relevant runtime failure types is required.

RI-2.1: Tool support for domain experts is required to support them in their task of identifying relevant engineering models used for the runtime failure detection.

RI-3.1: An empirical investigation of the effort needed to import the data from the relevant models into the EKB is required in order to compare the proposed approach with alternative or traditional approaches.

RI-4.1: Domain experts should be able to intuitively define rules using simple natural language-based methods and techniques. Therefore, tool support is required to support the domain experts in the definition of rules.

RI-5.1: Since runtime failure detection is an at least near-real time task, the performance of data access and reasoning at runtime needs to be measured and improved to allow the proposed approach to become accepted in the automation systems domain.

RI-6.1: Runtime assertions in the models of the EKB could possibly also be used for checking QoS parameters like system throughput. An extensive evaluation is required to show the applicability of the proposed approach for checking additional types of parameters.

RI-7.1: Further research is needed to integrate our ontology-based fault detection systems into automation devices to enable them to analyze their own operational mode.

RI-7.2: Automation devices may have the ability to predict and prevent their own and system wide failure modes on basis of the knowledge stored in the EKB. If an imminent failure mode is detected by the reasoner, appropriate counter measures could be initiated to prevent the failure mode to occur or to minimize the consequences of such failure modes. Therefore, concepts for such autonomous reactions need to be researched and evaluated.

RI-7.3: The proposed approach enables automation devices and the system as a whole to identify its error condition based on the basic error events and to prioritize the alarms according to their severity or relevance.

4.3.2 Discussion

Observations in a runtime environment are typically on-site and not linked to engineering environments, where automation systems focus on as-built documentations without any feedback to the engineering artifacts. Applying ASB concepts, data - captured in a runtime environment and semantically linked to engineering environments enable more efficient defect detection and location (e.g., hyperlinks to related components and signals). In addition, captured data supports defect prediction as a foundation for maintenance and condition monitoring (e.g., based on knowledge on the frequency and type of sensor defects). In automation systems, defects result in alarms and alarm chains. Engineering knowledge - linked to run-time data - enable classification and prioritization of alarms,
caused by defects and support maintenance engineers in efficiently and effective problem solving tasks. Thus, engineering knowledge drives defect detection and alarm handling in runtime environments.

5 Conclusion and Further Work

In modern industrial ASE projects improved tool support for engineering environment integration would be beneficial to minimize the effort of domain experts for bridging semantic gaps between the local data models of software tools used in the project. However, most engineering environments assume a homogeneous project-level data model, which would be easier to handle but seldom exists in real-world projects.

In this paper we discussed in three real-world ASE use cases the needs for semantic integration and applications of machine-understandable knowledge engineering: 1) the use of engineering knowledge to configure integrated engineering environments; 2) queries to engineering knowledge across software tools and their data models using project-level concepts; and 3) the use of engineering knowledge to configure integrated access to engineering knowledge and run-time data (e.g., for condition monitoring, flexible systems, defect detection and localization).

We introduced a framework for semantic integration in the ASE life cycle and provided an evaluation concept with empirical studies to measure the benefits and limitations of the proposed approach compared to the traditional expert-intensive approach. Major results of the initial evaluation were: (a) semantic integration has good potential to make engineering processes more efficient and robust; (b) these semantic integration approaches need to be supported well with user interfaces that end users find usable and useful.

Further work will be to improve the research prototypes based on the feedback of the empirical evaluation and to conduct empirical studies to strengthen the external validity of the initial experimental results.

References