Chapter 8

Improving Quality Assurance in Multidisciplinary Engineering Environments with Semantic Technologies

Dietmar Winkler, Marta Sabou and Stefan Biffl

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/66222

Abstract

In multidisciplinary engineering (MDE) projects, for example, automation systems or manufacturing systems, stakeholders from various disciplines, for example, electrics, mechanics and software, have to collaborate. In industry practice, engineers apply individual and highly specialized tools with strong limitation regarding defect detection in early engineering phases. Experts typically execute reviews with limited tool support which make engineering projects defective and risky. Semantic Web Technologies (SWTs) can help to bridge the gap between heterogeneous sources as foundation for efficient and effective defect detection. Main questions focus on (a) how to bridge gaps between loosely coupled tools and incompatible data models and (b) how SWTs can help to support efficient and effective defect detection in context of engineering process improvement. This chapter describes success-critical requirements for defect detection in MDE and shows how SWTs can provide the foundation for early and efficient defect detection with an adapted review approach. The proposed defect detection framework (DDF) suggests different levels of SWT contributions as a roadmap for engineering process improvement. Two selected industry-related real-life cases show different levels of SWT involvement. Although SWTs have been successfully applied in real-life use cases, SWT applications can be risky if applied without good understanding of success factors and limitations.

Keywords: Semantic Web Technologies, automation systems engineering, multidisciplinary engineering, quality assurance, defect detection, engineering process improvement

1. Introduction

In multidisciplinary engineering (MDE) environments, engineers coming from different disciplines have to collaborate and exchange data. For instance, in automation systems engineering (ASE) projects, mechanical, electrical, and software engineering disciplines work together to build manufacturing plants, power plants, or steel mills [1]. However, engineers typically use isolated tools with data models that provide only limited capabilities regarding data exchange and synchronization of engineering plans across disciplines and organizations [2]. Such loosely coupled tools and incompatible data models hinder efficient engineering processes, project management, quality assurance, and engineering process improvement [3]. Due to limited collaboration capabilities, engineering projects become risky and error-prone.

However, the collaboration of different engineering disciplines requires efficient mechanisms for quality assurance and defect detection [4]. In the context of this paper, defects represent errors in engineering plans and design documents, deviations and inconsistencies between engineering plans. Established tool suites, that is, all-in-one solutions, such as EPlan Engineering Configuration (EEC)[1] or COMOS[2], with integrated data models typically include basic quality assurance mechanisms, such as syntax checks or basic consistency checks to identify intra-disciplinary defects (i.e., defects within one engineering artifact or between artifacts within one discipline). It is worth noting that there is no strong support for identifying interdisciplinary defects, that is, defects that affect two or more disciplines. Thus, in tool networks, where no integrated data models are available, further research is required to enable effective and efficient defect detection. Performed research revealed that there is very limited support for defect detection in heterogeneous environments. The latter include but are not limited to identifying inconsistencies between variables of mechanical and software (control) engineering artifacts because of the technical heterogeneity of tools and the semantic heterogeneity of data models [5].

The application of reviews [6] (or software inspection) in software engineering supports early defect detection [6], typically executed by human experts in a paper-based way, that is, without or with limited tool support. Thus, defect detection requires a high effort and includes risks to oversee important defects, even in homogenous engineering artifacts such as design documents or software code. In distributed and heterogeneous engineering environments, reviews require additional knowledge of human experts, who are sufficiently familiar with at least two related disciplines and require high cognitive skills. Again, review tasks are often executed manually by these experts and require as such considerable effort. It is worth mentioning that preliminary research indicates that they often miss important defects [7], especially if more than one engineering discipline is involved. Thus, an important issue is the heterogeneity of data models, embodied within applied tools.

Semantic Web Technologies (SWTs) can provide concepts for closing the gap between data models for data management, that is, the identification of data entities and knowledge, data

[1] EPlan: www.eplan.de/.

[2] COMOS: w3.siemens.com/mcms/plant-engineering-software/de/Seiten/Default.aspx.

http://dx.doi.org/10.5772/66222

organization, storage, and querying for project analysis purposes [8]. Thus, the application of SWTs can provide the foundation for effective and efficient defect detection mechanisms. However, even without using SWTs, systematic defect detection approaches, such as (software) reviews or inspection [9], can support defect detection processes, by applying guidelines and reading techniques for defect detection support. Thus, the main research challenges focus on (a) providing expert support for identifying defects in an interdisciplinary context more effectively and efficiently and (b) improving defect detection in organizations with or without SWTs. A defect detection framework (DDF) aims at providing an approach to provide the degree of SWT contributions from solution approaches without SWT (lowest level) to a fully supported SWT solution (highest level). For every level, this chapter describes a prototype solution of a real-life use case, developed together with industry and research experts for evaluation purposes.

Thus, the main goals of this chapter focus on (a) providing a defect detection framework (DDF) to support the introduction of SWTs in an organization on various levels of granularity and (b) to provide lessons learned from industry prototypes in context of SWT applications for defect detection in industrial and research real-life use cases. Further, the chapter demonstrates different levels of SWT capabilities in industry use cases and report on lessons learned on the strengths and limitations of these approaches.

The remainder of this paper is structured as follows: Section 2 presents related work on multidisciplinary engineering (MDE), defect detection, and Semantic Web Technologies. Section 3 motivates research issues, and Section 4 presents the solution approach, that is, requirements and capabilities for defect detection in context of MDE and SWT and the defect detection framework with and without SWTs. Section 5 discusses industry (real-life) use cases and prototype implementations regarding defect detection capabilities, and Section 6 discusses qualitative assessment results and lessons learned on the limitations of individual levels of semantic integration.

2. Related work

This section summarizes related work on multidisciplinary engineering (MDE) environments, defect detection, and Semantic Web Technologies as foundation for defect detection tool prototypes in context of the defect detection framework (DDF) with and without SWTs.

2.1. Multidisciplinary engineering environments and automation systems engineering

In *multidisciplinary engineering (MDE)* environments for *automation systems engineering (ASE)*, various stakeholders coming from different disciplines have to collaborate along the automation systems life cycle [1, 10]. Typical examples of such MDE environments for ASE include hydro power plant engineering, steel mill engineering, or industrial production systems. Common to all application areas is the contribution of heterogeneous engineering disciplines, that is, mechanical, electrical, and software engineering, with specialized engineering tools and related specific and heterogeneous data models. However, these expert

tools typically have limitations regarding seamless data exchange and collaboration support of engineers [1, 2, 11], a challenge for effective and efficient defect detection. Defects are typically considered as deviations between engineering plans of heterogeneous disciplines, for example, a mismatch of variable definitions and their usage, missing components, or wrong components. Yet, these deviations could be real defects or required and intended changes, requested by an engineer of one specific discipline. For example, the exchange of a hardware sensor from analogue to a digital (required by the electrical engineer) might result in different numbers of pins that need to be addressed by the software engineer in the control software. This change needs to be identified and analyzed early in the engineering process. Changes that arise late in the engineering project, during the commissioning phase, typically require significantly higher efforts for rework [12], during the commissioning phase.

Figure 1 presents a typical sequential engineering process approach in the ASE domain with parallel engineering activities along the project course. Furthermore, selected and important engineering artifacts of leading engineering disciplines and isolated and distributed quality assurance activities for individual process steps are included in this sample process. In common industry projects, engineers often follow such a simplified sequential engineering process [1, 4], that is, system design, implementation, test and commissioning, and operation (**Figure 1**). Changes and defects can have a critical impact on products, engineering projects, and processes, as they are likely to propagate from early to later stages. Therefore, a need arises for early support of defect detection mechanisms in MDE for ASE projects.

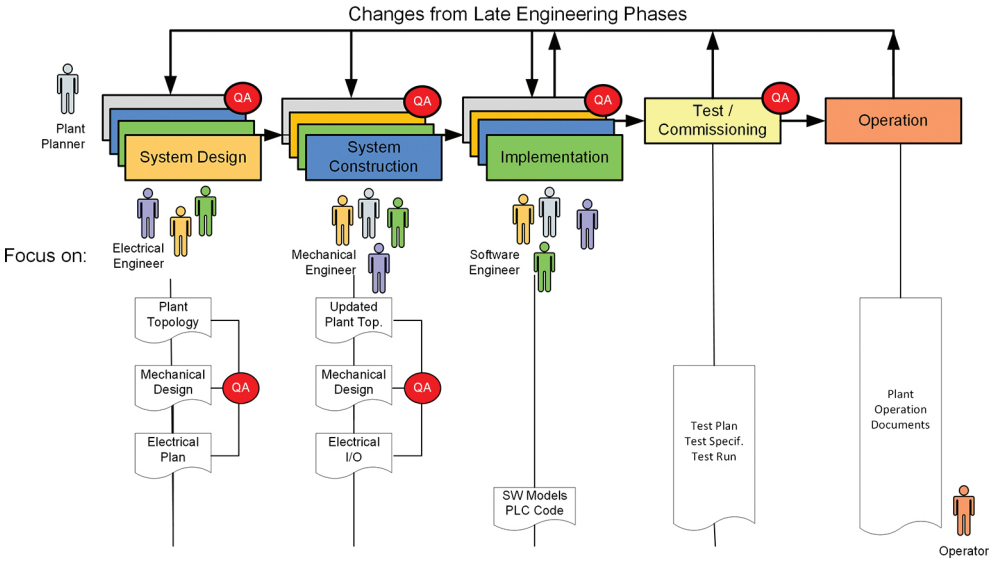

Figure 1. Typical sequential engineering process approach with parallel engineering activities [13].

http://dx.doi.org/10.5772/66222

Synchronization and data exchange in an engineering tool chain are typically executed manually or semi-automatically by applying local supporting tools (e.g., based on local database solutions or spread sheet solutions) [13]. However, these approaches incur high maintenance effort and assume the availability of experts, who are responsible for operating and maintaining these tools (often as add-on activities to their primary work tasks). Thus, there is a need for mechanisms that provide capabilities for mapping heterogeneous data models and for supporting defect detection processes across tools, domains, and data models. To overcome semantically heterogeneous data models, a common data exchange format could bridge the semantic gaps between engineering disciplines. The data format needs to be defined by consideration of all related tools; data to be exchanged need to be mapped to pass information and changes from one discipline (and data model) to the other. To overcome individual effort for data format definition and mapping, a standardized data exchange format is reasonable. *AutomationML* [14–16] is such an emerging data exchange standard that supports efficient synchronization of data produced by different engineering disciplines. However, *AutomationML* represents a data exchange standard and needs additional components that support data exchange, synchronization, and defect detection. The *AML.hub*[3] [17] is a platform for providing efficient data exchange/synchronization based on *AutomationML*. Although the *AML.hub* provides mechanisms for data exchange (e.g., mapping and merging), there is currently no support for quality assurance and defect detection.

2.2. Defect detection in software and systems engineering

Early *defect detection* is a key capability in engineering projects. In software engineering, software reviews are well established to identify defects in software artifacts early and efficiently [9, 18]. These reviews and inspections follow a defect detection process and enable the systematic identification of defects by single reviewers or a review team. Reading techniques are established approaches to support defect detection processes by providing guidelines on how to traverse and read an artifact under review [19]. For example, *scenario-based reading* supports the review of artifacts based on success-critical scenarios or business cases. The main advantage is the focus on real-life use cases that can be prioritized according to engineering risks or business value. *Perspective-based reading* takes the perspectives of different engineering roles, for example, systems architecture, test, or user perspectives to identify defects from various viewpoints. The primary advantage of this approach is the ability of the approach to identify defects from different perspectives based on individual experiences of the experts. In MDE environments, where engineers come from different disciplines, these disciplines may offer useful perspectives for reviewing engineering artifacts to find a variety of different defects.

However, in MDE environments review approaches are often conducted manually by experts. Thus, traditional reviews are effort consuming, expensive, and error-prone because reviewers can easily oversee related defects in engineering plans [20]. Mechanisms with tool-supported defect detection [21] aim to guide reviewers through the review process by focusing on most

[3] AML.hub video: https://www.youtube.com/watch?v=nPg66g46-eM&feature=youtu.be, access: 2016-09.

relevant aspects or critical system characteristics of the engineering artifacts [22]. In software engineering, *Gerrit Code Review*[4] is a widely used review approach for code reviews. Its main features include the comparison of two code variants (e.g., newly available/changed code components and previously developed code component versions, typically stored in a central code repository, GIT[5]), commenting of code fragments, and decision support (e.g., accepting or rejecting modifications). However, *Gerrit Code Review* does not provide any specific guidelines or process support for defect detection in code documents and is focused on software code or applicable on (structured) text elements. In context of defect detection in MDE environments, this approach is comparable to the *Focused Inspection approach* [23] that takes as input deviations or change analysis results between engineering plans to identify changes/defects from the perspective of mechanical, electrical, or software engineers. In order to reduce the manual effort in reviewing activities, SWT-based mechanisms can provide tool support for defect detection. Other tool solutions, such as *DefectRadar*[6], support defect detection by providing annotations and extensive commenting. *DefectRadar* has been applied in building automation without connection to software and/or systems engineering. As such, within the context of review support, the concept of *DefectRadar* can help to better identify entities, relationships, and attributes in different types of documents (e.g., engineering plans or models) to enhance understanding of the artifacts and to support defect detection. To the best of the authors' knowledge, there exists no tool that strongly supports defect detection in MDE environments in a comprehensive way. It is worth noting that Semantic Web Technologies (SWT) represent a promising approach to address these gaps and to support defect detection in MDE environments.

2.3. Semantic Web Technologies for defect detection

Semantic Web Technologies (SWTs) were used in several works to support consistency management across engineering models, including defect detection. Feldmann *et al*. [5, 25] focus on identifying inconsistencies that may arise among diverse engineering plans and engineering models created during the ASE process. Such inconsistency detection contributes to the increased productivity of the engineering process as it supports the detection of potentially severe defects early in the engineering process. The *resource description framework* (RDF) is used to uniformly represent engineering models, which are then queried with SPARQL,[7] a query language for ontologies similar to SQL for databases, to detect defects. Kovalenko *et al*. [26] present an ontology-based approach to automatically detect inconsistencies across heterogeneous engineering data sets. Ontologies are used to explicitly represent the discipline-/tool-specific knowledge and data in a machine-understandable form. Mappings are then defined between the ontologies to model cross-disciplinary (or cross-tool) relations between the data models and data sets explicit for knowledge integration. SPARQL queries are executed over the discipline/tool ontologies regarding the defined mappings in order to perform inconsis-

[4] Google Gerrit: https://www.Gerritcodereview.com, access 2016-09.

[5] GIT: https://git-scm.com/, access 2016-09.

[6] DefectRadar: https://www.defectradar.com/de, accessed 2016-09.

[7] SPARQL: www.w3.org/TR/rdf-sparql-query/, access 2016-09.

http://dx.doi.org/10.5772/66222

tency detection across discipline/tool boundaries. An approach for the automated validation of plant models is presented in [27], where CAEX models are transformed to ontology-based representations and verified through SPARQL queries. In the area of requirements and test case management, Feldmann et al. [28] present a modeling approach that enables the early integration of requirements and test cases. Furthermore, the authors present a case study based on Semantic Web Technologies (SWTs) to ensure the consistency of requirements as well as of requirements and test cases. A conceptual model that describes the main elements for this use case is developed and then formalized as an ontology. Reasoning mechanisms are applied to support various consistency checks-related cases and requirements. Basically, these works show how SWTs can be useful for defect detection in the automation systems domain.

3. Research issues and approach

In the context of *multidisciplinary engineering (MDE)* for *automation systems engineering* (ASE), strong limitations regarding early and efficient *defect detection* for distributed tools and heterogeneous data models were observed, where SWTs can help to semantically integrate these heterogeneous data models. An important question is, therefore, how well SWT mechanisms can help improve defect detection. From this fundamental question, the following research issues are derived:

RI.1: What are success-critical requirements in MDE to enable early, effective, and efficient defect detection to reduce project risks? Based on observations and literature, this chapter will identify a set of success-critical requirements to support managers and engineers in identifying defects early, effective, and efficient.

RI.2: How can SWT contributions be organized in a defect detection framework (DDF) to support defect detection on different levels? Different levels of SWT contributions include defect detection without SWT, defect detection with common concepts (basic approach), and defect detection based on semantically integrated data (advanced approach). However, different levels of SWT contribution can require process changes, additional effort for implementation and application, knowledge experts for SWT implementation, and may need new/adapted methodological approaches and tooling support.

The main question is how this framework can be used to introduce SWT-driven defect detection in organizations and how to establish an appropriate improvement strategy.

RI.3: What are the benefits and limitations of selected industrial cases and prototype implementations with respect to SWT contributions and defect detection in MDE environments? In the context of the CDL-Flex[8] research laboratory, researchers and industry collaborators developed application scenarios and real-life cases that focus on engineering process improvement and defect detection in industry contexts. Selected industry prototypes are presented for every level of the DDF including discussions on the benefits and limitations in context of requirements and

[8] CDL-Flex: Christian Doppler Laboratory for Software Engineering Integration for Flexible Automation Systems, http://cdl.ifs.tuwien.ac.at.

expected capabilities. Thus, RI.3 discusses results of prototype solutions and the strengths and limitations of SWT mechanisms in the context of an MDE development processes.

4. Defect detection framework with/without SWTs

This section summarizes basic requirements, needed defect detection capabilities, and introduces the defect detection framework (DDF) for assessing the level of SWT contributions for defect detection in MDE environments.

4.1. Requirements for defect detection in MDE environments

Requirements and key capabilities have been derived by industry and research experts as a foundation for supporting (a) the development of the defect detection framework and (b) for providing key requirements for offering tool support for defect detection. Basically, requirements include four different core topics: process capabilities (ability to support systematic, traceable, and repeatable processes from quality assurance perspective), organizational requirements (for organizing activities and engineering knowledge), defect detection capabilities (i.e., the core component for defect detection), and tool capabilities that focus on needs for defect detection tool support.

Process capabilities include process support for defect detection, embedded within engineering processes:

- *Systematic defect detection processes* to enable repeatability and traceability of defects and defect detection processes.
- *Traceability* of quality assurance activities focuses on traceability of quality assurance processes, for example, review processes, and results, that is, defects, in various engineering models across disciplines.

Organizational requirements focus on company and organization issues as prerequisites for implementing (and applying) defect detection approaches.

- *Defined roles and responsibilities* for organizing quality assurance activities.
- *Effort for method implementation* is an important factor in context of method implementation because of trade-off considerations of benefits and costs.
- *Knowledge and skills* needed. Within the context of SWT, knowledge experts can be required to support method implementation and maintenance. These expert efforts typically result in additional costs and need to be considered accordingly.

Defect detection performance represents the core capability for defect detection with focus on defect detection performance (i.e., efficiency and effectiveness) and the effort for applying the method/tool.

http://dx.doi.org/10.5772/66222

- *Defect detection effectiveness* refers to the capability of identifying most defects in engineering artifacts. Tool support should also increase the coverage by encompassing all relevant parts of the engineering object.
- *Method application effort* and *defect detection efficiency* refer to the effort for defect detection related to the identified defects (e.g., defects found per time interval). Tool support might also increase defect detection efficiency.

Tool capabilities include basic requirements for a tool solution (on different levels) to support domain experts in the context of defect detection.

- *Tools support* can help to increase defect detection performance, that is, effectiveness and efficiency. However, annotation support can provide additional information, comments on candidate detects for better assessing the correct interpretation of candidate defects.
- *Browsing capabilities* can help to better cross-check information across disciplines and engineering domain borders. Thus, this capability is seen as most valuable to enable identifying relationships within the engineering artifacts.
- *Automation supported difference checks*. Highlighting differences between model versions, that is, related to changes such as added, modified, or removed parts of engineering artifacts, promises to focus on critical changes in and of engineering artifacts.
- *Reporting capabilities* have to provide capabilities to generate reports out of the defect detection process to provide an overview on identified changes/deviations or defects for (quality) management purposes.

The implementation of SWT mechanisms can make data exchange and synchronization of heterogeneous data models more effective and efficient and can enable effective and efficient defect detection.

4.2. Defect detection framework concept

The application of SWT mechanisms requires knowledge engineering capabilities for data management, mapping, and querying. However, the scarce availability of knowledge engineers and the general scarcity of engineering resources [29] in the automation systems engineering (ASE) domain requires also "light-weight" defect detection approaches with and without SWT capabilities. *Light-weight* refers to the application of basic SWT concepts in engineering projects including support for defect detection and without high additional effort for implementing a comprehensive SWT solution for defect detection. **Figure 2** illustrates the *defect detection framework* with different levels of semantic integration approaches.

Defect detection level 0: isolated engineering plans and heterogeneous data sources. Current and traditional MDE approaches take as input heterogeneous artifact sources coming from various disciplines. This defect detection level does not include SWT contributions. Thus, defect detection processes are purely human based and require considerable manual effort for reviewing. Experts have to bridge the semantic gaps between engineering artifacts manually or by applying isolated support solutions that require high effort for application and mainte-

nance [30]. This applies specifically in the case of data model changes. Beyond the effort-consuming manual activities, high cognitive effort is required by experts [31]. The complexity of engineering plans often hides defects very well in various parts of and views on engineering plans. Thus, in MDE, defect detection on this level is often ineffective and always inefficient. While defect detection can be improved by applying reading techniques, such as scenario-based reading or perspective-based reading [32], the high cognitive effort and the manual bridging of semantic gaps still limit defect detection effectiveness and efficiency.

Defect detection level 1: common concepts bridge heterogeneous data models based on commonly used data aspects. Wache *et al*. [33] distinguish between three ontology-based approaches:, *single-ontology*, *multiple-ontology*, and *hybrid approaches*, to achieve semantic integration, as follows:

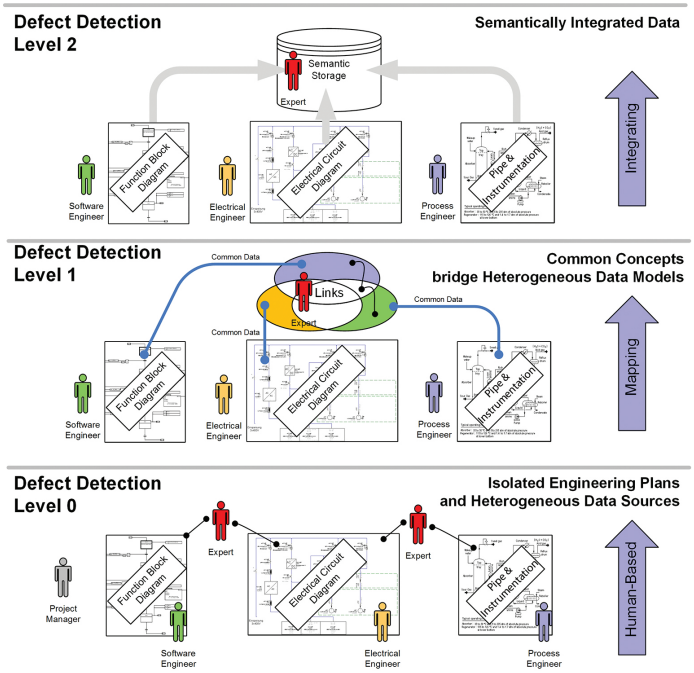

Figure 2. Conceptual approach for the defect detection framework (DDF) in the context of SWT contributions on three levels.

Single-ontology approaches use one ontology as a semantic bridge to integrate several data sources. The challenge is to create an ontology that is sufficiently broad in its coverage to describe all data sources that are integrated. Additionally, an introduction of a new data source requires changes to the ontology.

Multiple-ontology approaches create an ontology to semantically describe each data source to be integrated (i.e., local ontologies) and then create pair-wise mappings between these ontologies.

http://dx.doi.org/10.5772/66222

- *Hybrid approaches* on the other hand combine the previous strategies as follows: Local ontologies are created to describe each data source and are then mapped to a global ontology that contains concepts common to all integrated data sources (i.e., common concepts). The hybrid approach enables engineers to continue working in their well-known environments and enable data synchronization via these common concepts. For instance, in hydro power plant engineering, signals can be used as a common concept. A signal corresponds to electrical wiring or power levels of connectors (electrical discipline), physical layout and position of wires (mechanical discipline), and software variables for control software (software discipline). In the context of defect detection, common concepts enable reviewing engineering aspects by the review team, linked by common concepts. A major advantage of this approach is the lower cognitive burden of reviewers by providing clear links between engineering plans. Tools can be used to automate search tasks during the review by using querying mechanisms. However, this approach requires the upfront investment of defining and maintaining the common concepts data structures.

Defect detection level 2: semantically integrated data. All-in-one solutions (tool suites) typically consist of a common database including plans coming from all engineering disciplines [2] in a homogenous engineering environment. Yet, limitations of that approach include vendor lock-in, limited data exchange capabilities with tools that are not part of the tool suite (common in project consortia [34]), or low flexibility regarding the extensibility of the common data model.

In context of MDE for ASE environments, heterogeneous data models come from different sources and, as such, a solution that assumes a homogeneous data mode is not feasible. In such settings, SWTs can help to provide an integrated view on engineering data from various perspectives. SWT-based engineering model integration aims to bridge semantic gaps in engineering environments between project participants and their tools, who use different local terminologies [35]. Integrated data can then support the analysis, automation, and improvement in MDE processes. Semantic model integration is defined as solving problems originating from the intent to share information across disparate and semantically heterogeneous data [36]. These problems include the matching of data schemes, the detection of duplicate entries, the reconciliation of inconsistencies, and the modeling of complex relations in different data sources [37]. Integrated data and views on engineering data can support defect detection by (a) preparing review packages, that is, scoping specific parts of the overall system for reviewing purposes, and (b) enabling the detection of cross-disciplinary defects, thanks to the explicitly represented links between different disciplines established during the integration process.

While common concepts on level 1 represent manually stated links between data originating from different disciplines, links on level 2 are established semi-automatically by using explicit engineering knowledge. While the semantic integration with SWTs already supports defect detection activities, SWTs can be employed to create defect detection mechanisms, in particular, through semantic querying with SPARQL. Benefits of these approaches are: (a) defect detection tasks are explicitly represented in SPARQL queries, thus allowing periodic query repetition; (b) since queries typically address the ontology concepts, the checks remain valid even if the underlying data model, for example, of signal concepts, changes; and (c) semantic

query languages rely on reasoning mechanisms and are, therefore, well fitted [38] to check model consistency and coherency.

5. Industry use cases and lessons learned

This section presents two key *Use Cases* (UCs) developed in cooperation with industry partners and prototype solutions in context of the defect detection framework (DDF) without/with Semantic Web Technologies.

5.1. Defect detection without Semantic Web Technologies (level 0)

On Level 0, where a set of isolated engineering plans based on heterogeneous data sources is available, no SWTs are used. Engineering knowledge is implicitly embedded within domain experts [24], who are familiar with at least two engineering disciplines, for example, electrical and software or electrical and process discipline, as depicted in **Figure 2**. These experts bridge the semantic gap between aforementioned engineering disciplines manually. Thus, defect detection can become time-consuming and risky because experts can oversee defects easily within a huge amount of given data. For instance, in the hydro power plant domain, 40.000 data points (i.e., signals) are available and need to be reviewed/inspected by experts. Tool solutions like *Gerrit* can be used to compare different engineering model/plan versions to highlight changes to better drive the review process. However, the availability of a textual representation is required for enabling *Gerrit* applications. If no textual documents are available but engineering plans (such as diagrams) are present, *DefectRadar* can be used for annotation and review purposes [39].

Independent of applied tools, guidelines from Review/Inspection, such as perspective-based reading, scenario-based reading, or checklist-based reading technique approaches, can be used to focus on perspectives (defect detection from experts coming from specific disciplines) or scenarios (focus on use cases and application scenarios). Checklists based on the ISO 25010 standard [40] can represent a complementary approach to systematically check important (predefined and application-specific) items to support defect detection. Although such guidelines support review processes, the review itself is still performed manually and often paper based [41].

5.2. Common concepts with limited Semantic Web Technologies (level 1)

The first (basic) level of SWT contributions includes the application of common concepts as a hybrid ontology (see Section 4.2). The main idea of this approach is to elicit commonly used data from involved disciplines (even if they have different terminologies), transform them to a common concept, and map these common data between different disciplines. Thus, specifically used data are left within local tool solutions, and common concepts are centrally stored and are used for mapping data elements between related disciplines. Thus, changes in one discipline can be passed efficiently via common concepts to related disciplines by highlighting changes. Hence, the focus for review is based on deviations and changes (similar to

http://dx.doi.org/10.5772/66222

the *Gerrit* approach). **Figure 3** presents a simple workflow for difference/defect detection approach, and **Figure 4** shows a screenshot of the prototype implementation (i.e., highlighting deviations/changes of different engineering model/plan version as input for review).

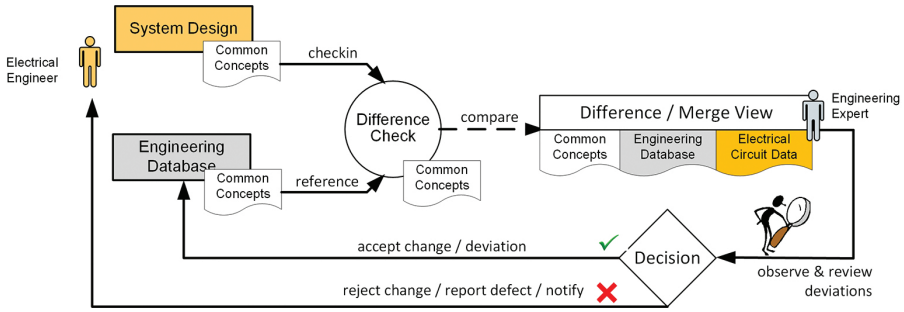

Figure 3. Basic workflow for difference/defect detection with focused reviews.

This concept has been implemented and evaluated as *"Engineering Object Change Management"* (UC 1) in the hydro power plant domain [42]. In the context of this use case *signals* can be seen as the foundation for common concepts that enable efficient data exchange between engineering disciplines and synchronization of heterogeneous engineering plans. A *Virtual Common Data Model* (VCDM) has been elicited with industry experts to link individual disciplines [43, 44]. Note that key parts of signals (coming from different disciplines) have to be identified, transformed to the VCDM, and mapped to enable a seamless data integration. A database holds the VCDM data and provides links to discipline-specific tools and data models. Based on this VCDM concept, a change management process supported by a web-based tool has been designed at the industry partner.

Figure 4. Prototype screenshot of difference checks of the prototype implementation.

The prototype tool is capable of handling changes, that is, new, modified, or removed signals across disciplines and domain borders. A *difference view*, that is, a comparison of already available signals and modified signals, enabled engineers to assess the signal data and to decide whether the deviation should be accepted as a change or be classified as a defect. The main results of the pilot study [4] were that the prototype solution enabled (a) more effective and

efficient data exchange in all phases of the engineering process and (b) observation capabilities of the change management process for project management and control.

In the context of defect detection, the explicit view on "deviation types" (i.e., changes or defects) provides the foundation for a more effective and efficient review process of signal data. An important benefit of this approach is the focus on the most critical deviations from a prior data version instead of reviewing the overall set of engineering data (signals) of the hydro power plant. Note that a typical hydro power plant includes a set of up to 40.000 signals and a small subset (2–5%) of deviations that allow to focus the review to 800 to 2.000 deviating signals. Although the prototype solution provides support for defect detection by presenting differences and deviations to the reviewer, the review process is executed by a human expert team without any specific guidance for defect detection. In this context, guidance refers to applying checklists, scenarios, or perspectives to identify defects more effective and efficient. Justification of defects typically relies on expert domain knowledge and engineering experience and is based on industry and organization best practices.

Although the automated identification of deviations and changes has been found useful by experts in context of the prototype implementation, additional guidelines, such as reading techniques, can help improve defect detection performance in terms of increased effectiveness (i.e., increased number of defects found) and efficiency (i.e., identifying more defects per time interval), and decreased false positives (i.e., wrongly reported defects).

Based on the provided focus on deviations and changes, the *"Focused Inspection"* [45] and "*Model Quality Assurance*" approaches [46] help to improve the unguided defect detection process by supporting (a) the review team selection and (b) their reading process. Individual engineers focus on defined aspects of the systems, for example, on mechanical, electrical, or software characteristics of the engineering plan, and their overlaps with partner disciplines. Thus, the reading technique can take into account individual disciplines, perspectives, and typical candidate defects that occur in discipline-specific engineering plans and across engineering plans. In this use case, *common concepts* (and the VCDM) are used as vehicle for data synchronization and for improving reviewing processes. The reviewing process is supported by a difference view (tool support) and reading technique (method support). In the pilot study [4], both the defect detection effectiveness and efficiency increased. However, an additional knowledge experts or key users that are familiar with knowledge engineering are required for supporting the common concept elicitation, transformation, and mapping process. In the prototype solution, a knowledge expert, that is, one of the authors, supports these process steps. However, additional tool support is planned as future work that guides domain experts and key users to conduct these knowledge management tasks without deep understanding of SWT mechanisms.

5.3. Integrated data with advanced Semantic Web Technologies (level 2)

The second (advanced) level of SWT contributions includes the establishment of integrated data based on a commonly used data exchange language (see Section 4.3). The *AutomationML*

http://dx.doi.org/10.5772/66222

Analyzer[9] (UC 2) uses SWTs to provide an interface for analyzing data from integrated *AutomationML* files.

AutomationML [14] is an emerging data format for exchanging engineering data [47]. While the concerted use of *AutomationML* in an engineering project makes the integration between data from different engineering disciplines easier, tools are still needed for navigating and analyzing integrated *AutomationML* data more easily. To that end, the *AutomationML Analyzer* [48] uses ontology-based technologies to integrate *AutomationML* data to provide easy navigation support within the *AutomationML* data as well as to detect project level inconsistencies and defects through SPARQL querying of the integrated data. The main SWT capabilities used are (a) semantic modeling of an *AutomationML* ontology, which enables semantically enriching the input data; (b) browsing and exploration of the semantic data through *Linked Data*-based mechanisms; and (c) the use of reasoning mechanisms as part of the SPARQL querying activities.

This prototype fits Level 2 of the framework because it uses SWTs both for data integration and for defect detection activities. **Figure 5** presents the concept for the *AutomationML Analyzer*, including review process support. Core components of the concept include the *AutomationML Analyzer* component, providing integrated data and the *review process support* component that enables required review functionality such as browsing through the plant topology, automated execution of defined queries to identify deviations and changes, and querying capabilities for reporting. As a prerequisite, engineering data are available in *AutomationML* data format. However, the concept is applicable to any structured data formats required within a project. In the context of this prototype, the *AML.hub* [17, 49] represents the platform for data synchronization (i.e., mapping and merging of data coming from heterogeneous sources).

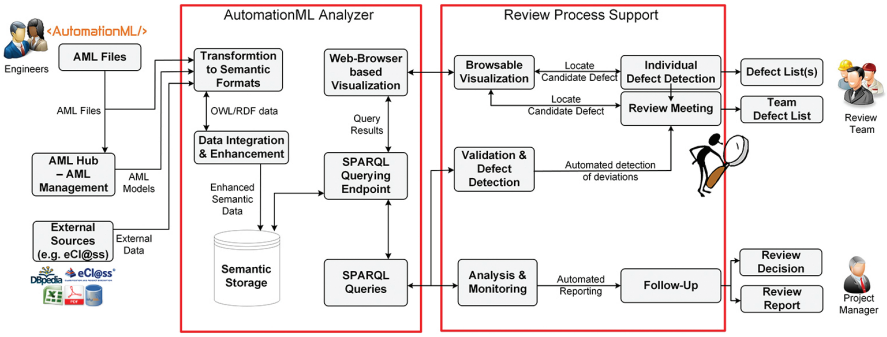

Figure 5. *AML Analyzer* concept for review support [48].

For establishing the *AutomationML Analyzer*(and the underlying mechanisms), a knowledge engineer is required to perform a multitude of tasks, such as transformations, data integration,

[9] AutomationML Analyzer: http://data.ifs.tuwien.ac.at/aml/analyzer, access 2016-09.

and querying capabilities. However, in common industry projects where similar projects are conducted, this effort represents initial activities and the results, for example, transformation, mapping strategies, and queries, can be reused for similar projects within the organization.

6. Discussion, limitations, and future work

The main goal of this paper focuses on building a platform for users and domain experts, that is, experts in MDE and ASE environments, for executing review activities more effectively and efficiently (process improvement) and to improve *defect detection* performance in terms of increased defect detection effectiveness and efficiency (method application improvement). Several strategies can help to improve defect detection processes and activities by using a certain degree of Semantic Web Technology involvement. The defect detection framework (DDF) provides a basic framework for approaches in context of defect detection with a different degree of SWT contributions.

Table 1 presents *identified requirements and expected key capabilities* (y-axis) for tool solutions (RI.1). These requirements and capabilities have been elicited in collaboration with industry and research experts (Section 4.1). RI.2 focuses on the development a *defect detection framework (DDF),* illustrated in **Figure 2** and represented by the x-axis in **Table 1**, which is capable of providing an overview of different levels of SWT contributions for defect detection. Within the framework of this chapter, three levels of SWT contributions have been found useful for assessing the level of SWT contribution and for driving improvement initiatives in context of quality assurance and defect detection. On different level of SWT contributions, selected industry real-life prototypes have been developed and conceptually evaluated (RI.3) to elicit benefits and limitations of individual SWT-related solutions. Domain experts bridge technical gaps between isolated tools and semantic gaps between heterogeneous data models manually with limited tool support. In addition, defect detection is executed manually with high cognitive effort. To overcome these limitations (i.e., high effort and error-proneness of manual defect detection), common concepts help to bridge the gap based on commonly available data. *Defect Detection Level 1* focuses on these common concepts, that is, commonly used data that support data exchange between disciplines. Transformations and mappings of local tool data to common concepts help to bridge the semantic gaps of individual data. Based on these mappings, changes can be propagated via common concepts from one discipline to the other. Furthermore, defect detection is supported by providing difference/deviation checks to focus on differences between artifact versions or artifacts derived from different disciplines (see **Figure 4** in Section 5.2). This approach has been successfully implemented in industry (see Section 5.2) which supports the engineering team *Defect Detection Level 2*, uses advanced approaches from SWTs, and builds on semantically integrated data that enable browsing, querying, and automation-supported defect detection and reporting—implemented in the *AutomationML Analyzer* (see Section 5.3). Note that this approach uses *AutomationML*, an emerging standard for data exchange in MDE environments for enabling data management, that is, data mapping and synchronization. Yet, the conceptual approach supports various types of data formats and has been implemented with *AutomationML* data (see Section 5.3).

http://dx.doi.org/10.5772/66222

	Level 0 (human based)	Level 1 (common concepts)	Level 2 (integrated data)
Process capability			
Systematic defect detection	Low	High	High
Traceability	Low	High	High
Organizational requirements			
Defined roles and responsibilities	Medium	Medium	Medium
Effort for method implementation	Low	Medium	High
Knowledge and skills needed	Medium	High	High
Defect detection performance			
Defect detection effectiveness	Low	High	High
Defect detection efficiency	Low	High	High
Method application effort	High	Medium	Low
Tool capabilities			
Tool support	Low	Medium	High
Browsing capabilities	Low	Low	High
Automated supported difference checks	Low*	High	High
Reporting capabilities	Low	Medium	High

*Depending on the involved alternative tools, for example, *Gerrit* or *DefectRadar*.

Table 1. Conceptual evaluation of the defect detection framework.

However, to enable the elicitation of common concepts, transformation and mapping (for Level 1) and the construction of integrated data (for Level 2) requires knowledge engineers who support users and domain experts, elicit, and implement SWT approaches.

Table 1 presents an assessment of SWT contribution levels (and prototype solutions) and identified requirements for process and tool support in context of defect detection. *Human-based defect detection* (Level 0) is based on expert activities and their expertise with limited process support and limited tool support. However, due to organizational activities, reviews and inspection can still be implemented in MDE environments on a basic level (e.g., by using organizational guidelines and supporting tools). *Common Concepts* (Level 1) support defect detection by enabling links and mechanisms for data synchronization and defect detection, for example, highlighting changes in artifacts derived from different disciplines or artifact versions, to focus on recent changes and deviations. Instead of reviewing thousands of engineering data, they can focus on deviations, i.e., changes, inconsistencies, or defects. In the evaluation context, the prototypes showed:

- improved process and organizational support,
- improved defect detection capabilities, and

- some basic tool support.

Integrated Data (Level 2) builds on a high degree of SWT and data integration and enables advanced defect detection by also supporting review processes, querying, navigation, and enhanced tool support.

With focus on the *defect detection framework* (RI.2), the three-level concept has considerable benefits for classifying SWT contributions as starting point for improving/introducing SWTs in an organization to improve collaboration and defect detection performance. On the other hand, defect detection performance can also be improved with low effort, by introducing systematic reading techniques or tool support to focus on most critical and important deviations. Nonetheless, to benefit from SWTs capabilities, for example, queries, reporting, and navigation, knowledge engineering capabilities are required to support engineers in data management and query definition.

6.1. Limitations and threats to validity

The *Requirements and Tool Capabilities* have been detailed and discussed with representative industry and research experts in the field of automation systems engineering. These industry and research experts have been recruited from the automation systems development domains, for example, the hydro power plant domain. Although identified requirements and capabilities match to the needs of these experts, additional requirements and needed tool capabilities might arise if different application domains are considered, for example, building automation, factory automation, or chemical production systems. For instance, common concepts need to be adapted according to available data in related engineering domains.

The *defect detection framework (DDF)* has been developed within the context of the current research work. It has been executed in the CDL-Flex, which is driven by industry partners and their respective individual needs. However, it is the view of the authors that the framework may assist industry and research entities so as to drive and foster improvement initiatives. Although this basic framework has been found useful in the evaluation context, additional levels of SWT contributions might be reasonable and applicable for different application domains. Further research work is needed to include additional applied SWT contributions applicable in different domains.

Selected *Use Cases and Prototype Solutions* have been developed based on industry-specific requirements that have been derived from industry partners in the automation systems domain, mainly hydro power plant engineering, steel mill engineering, or production automation engineering. Assuming comparable requirements in different domains, SWT contributions need to be evaluated in these additional application domains. Nonetheless, the conceptual real-life use cases and prototype implementations have been evaluated in various industry contexts successfully (see Section 4 for the solution concepts and Section 5 for individual industry real-life user cases). Based on described prototype solutions, additional application experiences in the defined industry context (i.e., business domain of industry partners) are needed. In addition, requirements from related business domains need to be explored and considered in the presented solution concepts. Therefore, the plans include more

http://dx.doi.org/10.5772/66222

detailed large-scale empirical studies (e.g., case studies) in real-world industry settings at existing industry partners and beyond. SWTs can make defect detection more effective and efficient. It is worth noting though that a higher level of SWT contribution requires considerable knowledge engineering capabilities to support data management and query definition. Although SWT concepts have been successfully applied in industry, their application can be risky without good understanding of their success factors and limitations.

6.2. Future work

Future work focuses on evaluating and improving the presented solution concepts and prototype solutions, extending and enhancing the prototype solution within the current application domains and addressing needs from additional application domains, and extending the platform toward a comprehensive process support for enabling and improving comprehensive reviewing with SWT and tool support. More specifically, future work includes:

- Improving solution concepts and prototype solutions based on experiences from real-world applications, for example, pilot application in real-life industry projects at target industry partners from the hydro power plant domain, steel mill engineering, and manufacturing system development. Future goals also focus on exploring requirements and needs in additional industry application domains for upcoming development steps.
- Prototype evaluation will be extended toward more detailed investigations in large-scale industry contexts with large-scale real-world data (beyond pilot applications and prototype applications in small contexts).
- Future plans also include strengthening the process support for defect detection by providing a more comprehensive view on quality assurance based on established review and inspection processes (see Section 2.2 for the traditional review process approach).
- Finally, the defect detection framework with SWT contributions needs to be revisited, improved, and extended to establish a framework to drive engineering process improvement with SWT mechanisms.

It is the authors' view that SWTs can make defect detection more effective and efficient. It is worth noting though that a higher level of SWT contribution requires considerable knowledge engineering capabilities to support data management and query definition. Although SWT concepts have been successfully applied in industry, their application can be risky without good understanding of their success factors and limitations [50].

Acknowledgements

Part of this work was supported by the Christian Doppler Forschungsgesellschaft, the Federal Ministry of Economy, Family and Youth and the National Foundation for Research, Technology and Development, Austria.

Author details

Dietmar Winkler[1,2*], Marta Sabou[2] and Stefan Biffl[2]

*Address all correspondence to: dwinkler@sba-research.org

1 SBA Research gGmbH & Institute of Software Technology and Interactive Systems, CDL-Flex, Vienna University of Technology, Vienna, Austria

2 Institute of Software Technology and Interactive Systems, CDL-Flex, Vienna University of Technology, Vienna, Austria

References

[1] Biffl S., Schatten A., Zoitl A. Integration of heterogeneous engineering environments for the automation systems lifecycle. In: Proceedings of the 7th International Conference on Industrial Informatics (INDIN); 23–26 June 2009; Cardiff, Wales. IEEE; 2009. p. 576–581. doi: 10.1109/INDIN.2009.5195867

[2] Fay A., Biffl S., Winkler D., Drath R., Barth M. A Method to evaluate the openness of automation tools for increased interoperability. In: Proceedings of the 39th Annual Conference of the IEEE Industrial Electronics Society (IECON); 10–13 November; IEEE; 2013. doi: 10.1109/IECON.2013.6700266

[3] Biffl S., Lüder A., Winkler D. Multi-disciplinary engineering for Industrie 4.0: semantic challenges, needs, and capabilities. In: Biffl S., Sabou M., editors. Semantic Web for Intelligent Engineering Applications. 1st ed. Springer, New York; 2016.

[4] Winkler D., Moser T., Mordinyi R., Sunindyo W.D., Biffl S. Engineering object change management process observation in distributed automation systems projects. In: Industrial Proceedings of the 18th EuroSPI Conference; Roskilde, Denmark; 2013.

[5] Feldmann S., Herzig S.J.I., Kernschmidt K., Wolfenstetter T., Kammerl D., Qamar A., Lindemann U., Krcmar H., Paredis C.J.J., Vogel-Heuser B. Towards effective management of inconsistencies in model-based engineering of automated production systems. In: Proceedings of the IFAC Symposium on Information Control in Manufacturing (INCOM); 2015.

[6] Rössler P. Schlich M., Kneuper R. Reviews in der System- und Softwareentwicklung: Grundlagen, Praxis, kontinuierliche Verbesserung. 1st ed. dpunkt.verlag; 213. 176 p.

[7] Kovalenko O., Serral E., Sabou M., Ekaputra F.J., Winkler D., Biffl S. Automating cross-disciplinary defect detection in multi-disciplinary engineering environments. In: Proceedings of 19th International Conference on Knowledge Engineering and Knowledge Management (EKAW); Linköping, Sweden. 2014.

[8] Biffl S., Sabou M., editors. Semantic Web for Intelligent Engineering Applications. 1st ed. Springer, New York; 2016.

[9] Aurum A., Petersson H., Wohlin C. State-of-the-art: software inspections after 25 years. Software Testing, Verification and Reliability. 2002;12(3):133–154.

[10] Biffl S., Moser T., Winkler D. Risk assessment in multi-disciplinary (Software+) engineering projects. International Journal of Software Engineering and Knowledge Engineering (IJSEKE), Special Session on Risk Assessment. 2011;21(2):211–236.

[11] Barth M., Drath R., Fay A., Zimmer F., Eckert K. Evaluation of the openness of automation tools for interoperability in engineering tool chains. In: Proceedings of the 17th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA; 17.-21.09.2012); Krakow, Poland. IEEE; 2012.

[12] Winkler D., Biffl S. Improving quality assurance in automation systems development projects. In: M. Savsar, editor. Quality Assurance and Management. Intech Publishing; 2012. p. 379–398. doi: 10.5772/33487

[13] Biffl S., Mordinyi R., Steininger H., Winkler D. Integrationsplattform für anlagenmodellorientiertes Engineering - Bedarfe und Lösungsansätze. In: Vogel-Heuser B., Bauernhansl T., tenHompel M., editors. HandbuchIndustrie 4.0. 2nd ed; 2016.

[14] IEC 62714-1: Engineering data exchange format for use in industrial automation systems engineering—Automation Markup Language—Part 1: Architecture and general requirement. International Standard; 2014

[15] IEC 62714-2: Engineering data exchange format for use in industrial automation systems engineering—Automation Markup Language—Part 2: Role class libraries. International Standard; 2015

[16] IEC 62714-3: Engineering data exchange format for use in industrial automation systems engineering—Automation Markup Language—Part 3: Geometry and Kinematics. International Standard; 2017 (forthcoming)

[17] Winkler D., Biffl S., Steininger H. Integration von heterogenen Engineering Daten mit AutomationML und dem AML.hub: Konsistente Daten über Fachbereichsgrenzen hinweg. develop3. 2015;3/15.

[18] Laitenberger O., DeBaud J-M. An encompassing life cycle centric survey of software inspection. Journal of Systems and Software. 2000;50(1):5–31.

[19] Travassos G., Shull F., Fredericks M., Basili V.R. Detecting defects in object-oriented designs: using reading techniques to increase software quality. ACM Sigplan. 1999;34(10):47–56.

[20] Biffl S., Halling M. Investigating the defect detection effectiveness and cost benefit of nominal inspection teams. IEEE Transactions on Software Engineering. 2000;29(5):385–397.

[21] Lanubile F., Mallardo T., Calefato F. Tool support for geographically dispersed inspection teams. Software Process Improvement and Practice. 2003;8(4):217–231.

[22] Anderson P., Reps T., Teitelbaum T., Zarins M. Tool support for fine-grained software inspection. IEEE Software. 2003;20(4):42–50.

[23] Winkler D., Biffl S. Focused inspections to support defect detection in multi-disciplinary engineering environments. In: Proceedings of the 16th International Conference on Product-Focused Software Process Improvement (PROFES); Bozen-Bolzano, Italy. Springer, New York; 2015.

[24] Winkler D., Musil J., Musil A., Biffl S. Collective intelligence-based quality assurance: combining inspection and risk assessment to support process improvement in multi-disciplinary engineering. In: Proceedings of the 23rd EuroSPI Conference; 14.-16.09.2016; Graz, Austria. Springer, New York; 2016. 163–175 p.

[25] Feldmann S., Herzig S.J., Kernschmidt K.,Wolfenstetter T., Kammerl D., Qamar A., Lindemann U., Krcmar H., Paredis C., Vogel-Heuser B. A comparison of inconsistency management approaches using a mechatronic manufacturing system design case study. In: Proceedings of the IEEE International Conference on Automation Science and Engineering (CASE). IEEE; 2015.

[26] Kovalenko O., Moser T. Using explicit and machine-understandable engineering knowledge for defect detection in automation systems engineering. In: Proceedings of International Doctoral Symposium on Software Engineering and Advanced Applications (IDoSEAA); Oulu, Finland; 2011.

[27] Abele L., Legat C., Grimm S., Müller A.W. Ontology-based validation of plant models. In: 11th IEEE International Conference on Industrial Informatics (INDIN); IEEE; 2013. 236–241 p.

[28] Feldmann S., Rösch S., Legat C., Vogel-Heuser B. keeping requirements and test cases consistent: towards an ontology-based approach. In: 12th IEEE International Conference on Industrial Informatics (INDIN); 2014. 726–732 p.

[29] Ekaputra F., Serral E., Winkler D., Biffl S. An analysis framework for ontology querying tools. In: Proceedings of the 9th International Conference on Semantic Systems (I-SEMANTICS) in conjunction with the 13th International Conference on Knowledge Management and Knowledge Technologies (I-KNOW). ACM; 2013.

[30] Biffl S., Mordinyi R., Moser T. Anforderungsanalyse für das integrierte Engineering – Mechanismen und Bedarfe aus der Praxis. ATP Edition; 2012;54(5):28–35.

[31] Blackwell A., Green T. Notational systems—the cognitive dimensions of notations framework. In: Carroll J., editor. HCI Models, Theories, and Frameworks: Toward a Multidisciplinary Science.Morgan Kaufmann, San Francisco; 2003. 103–134 p.

http://dx.doi.org/10.5772/66222

[32] Winkler D. Improvement of Defect Detection with Software Inspection Variants: A Large-Scale Empirical Study on Reading Techniques and Experience. VDM. Saarbrücken, Germany, 2008.

[33] Wache H., Vögele T., Visser U., Stuckenschmidt H., Schuster G., Neumann H., Hübner S. Ontology-based integration of information—a survey of existing approaches. In: Proceedings of IJCAI workshop: ontologies and information sharing; 2001. pp. 108–117.

[34] Ekaputra F., Serral E., Winkler D., Biffl S. Addressing data integration and communication issues in project consortia. In: Proceedings of the International Conference on Data and Software Engineering (ICoDSE), Bandung, Indonesia; 2014.

[35] Aldred L., van der Aalst W., Dumas M., Hofstede A. Understanding the challenges in getting together: the semantics of decoupling in middleware. BPM Center Report. 2006;BPM-06-19.

[36] Halevy A. Why your data won't mix. Queue. 2005;3(8):50–58.

[37] Noy N.F., Doan A.H., Halevy A.Y. Semantic integration. AI Magazine. 2005;26(1):7–9.

[38] Moser T., Mordinyi R., Mikula A., Biffl S. Efficient integration of complex information systems in the ATM domain with explicit expert knowledge models. Complex Intelligent Systems and their Applications. 2010;41:1–19.

[39] Winkler D., Biffl S. Collaborative Model Review Support for AutomationML. 4th AutomationML User Conference, Esslingen, Germany, 2016 (upcoming).

[40] ISO/IEC 25010:2011. Systems and Software Engineering—Systems and Software Quality Requirements and Evaluation (SQuaRE)—System and Software Quality Models. Standard, March 2011.

[41] Melo W., Shull F., Travassos G. Software Review Guidelines. Technical Report ES-556/01, COPPE/UFRJ, 2001.

[42] Winkler D., Biffl S. Focused inspection to support defect detection in automation systems engineering environments. Technical Report. TU Wien. IFS-CDL 15-02. Sep 2015. Online available: http://qse.ifs.tuwien.ac.at/publication/IFS-CDL-15-02.pdf.

[43] Sunindyo W., Moser T., Winkler D. Project progress and risk monitoring in automation systems engineering. In: Proceedings of the 5th Software Quality Days. Springer, New York; 2013.

[44] Sabou M., Kovalenko O., Novak P. Semantic modelling and acquisition of engineering knowledge. In: Biffl S., Sabou M., editors. Semantic Web for Intelligent Engineering Applications. 1st ed. Springer, New York; 2016.

[45] Winkler D., Ekaputra F.J., Biffl S. AutomationML review support in multi-disciplinary engineering environments. In: Proceedings of the 21st IEEE International Conference on Emerging Technologies and Factory Automation (ETFA); Berlin, Germany. IEEE; 2016.

[46] Winkler D., Wimmer M., Biffl S. Model quality assurance for multi-disciplinary engineering. In: Biffl S., Lüder A., Gerhard D. (eds): Multi-Disciplinary Engineering of Cyber-Physical Production Systems, Book Chapter, Chapter 17, 2017 (upcoming).

[47] Drath R., editor. Datenaustausch in der Anlagenplanung mit AutomationML: Integration von CAEX, PLCopen XML und COLLADA. 1st ed. Springer; 2010.

[48] Sabou M., Ekaputra F.J., Kovalenko O. Supporting the engineering of cyber-physical production systems with the AutomationML analyzer. In: Proceedings of the CPPS Workshop; Vienna, Austria. 2016.

[49] Mordinyi R., Wimmer M., Biffl S. Versioning in multi-disciplinary engineering with the AutomationML Hub. 4th AutomationML User Conference, Esslingen, Germany, 2016 (upcoming).

[50] Steyskal S., Wimmer M. Leveraging semantic web technologies for consistency management in multi-viewpoint systems engineering. In: Biffl S., Sabou M., editors. Semantic Web for Intelligent Engineering Applications. 1st ed. Springer, New York; 2016.