Automating Cross-Disciplinary Defect Detection in Multi-disciplinary Engineering Environments

Olga Kovalenko[1], Estefanía Serral[2], Marta Sabou[1], Fajar J. Ekaputra[1], Dietmar Winkler[1], and Stefan Biffl[1]

[1] Christian Doppler Laboratory for Software Engineering Integration for Flexible Automation Systems, Vienna University of Technology, Favoritenstrasse 9-11/E188, A-1040 Vienna
{firstname.lastname}@tuwien.ac.at
[2] Department of Decision Sciences and Information Management, KU Leuven Naamsestraat 69, B-3000 Leuven, Belgium
estefania.serralasensio@kuleuven.be

Abstract. Multi-disciplinary engineering (ME) projects are conducted in complex heterogeneous environments, where participants, originating from different disciplines, e.g., mechanical, electrical, and software engineering, collaborate to satisfy project and product quality as well as time constraints. Detecting defects across discipline boundaries early and efficiently in the engineering process is a challenging task due to heterogeneous data sources. In this paper we explore how Semantic Web technologies can address this challenge and present the Ontology-based Cross-Disciplinary Defect Detection (OCDD) approach that supports automated cross-disciplinary defect detection in ME environments, while allowing engineers to keep their well-known tools, data models, and their customary engineering workflows. We evaluate the approach in a case study at an industry partner, a large-scale industrial automation software provider, and report on our experiences and lessons learned. Major result was that the OCDD approach was found useful in the evaluation context and more efficient than manual defect detection, if cross-disciplinary defects had to be handled.

1 Introduction

Multi-disciplinary engineering (ME) projects represent complex environments, where different disciplines (e.g., mechanical, electrical and software engineering), have to collaborate efficiently in order to deliver high-quality end products and to satisfy tight timeframes [12]. An example ME project is designing a power plant and corresponding control system. In ME projects, engineering data is spread over numerous heterogeneous data sources derived from various discipline-specific tools and corresponding data models and data formats. Typically the tools are loosely coupled with limited capabilities for cross-disciplinary data exchange or/and data analysis. As a consequence, risks of deviations and defects in project data increase.

Errors made during the early development stages are especially critical, as they can potentially affect the artifacts of all latter phases. Staying undetected, such defects can lead to costly corrections during the commission phase or even to failures during the operation phase. Hence, ensuring consistency of project data and detecting emerging deviations and defects early are crucial requirements [6]. To this end, handling cross-disciplinary defects raises additional challenges. A cross-disciplinary defect is a defect that can only be identified by analyzing the data from several disciplines. For instance, the sensor type specified in the physical topology (mechanical engineering) of the automation system has to match the information in the corresponding electrical plan (electrical engineering) and the value range for control variables (software engineering) to describe a correct system. It is important to mention that interrelations between data of different disciplines are often not explicitly documented, but instead are only implicitly available as internal knowledge of the engineers. As the cross-disciplinary relations are not represented in a machine-understandable way, they cannot be automatically checked. Therefore, cross-disciplinary defect detection typically requires manual checking by the engineers, which is time consuming and error-prone. Thus, there is a need for effective and efficient defect detection methods, including cross-disciplinary defect detection, in ME projects.

In this paper we present an ontology-based approach for automated defect detection across discipline boundaries. Ontologies allow representing the data models of different engineering disciplines and/or tools and cross-disciplinary interrelations between the heterogeneous project data in a machine-understandable form, thus, enabling automated processing and analysis across disciplines. Comprehensive queries can be defined and executed in order to (a) check whether or not project data satisfies specified cross-disciplinary dependencies; and (b) to detect deviations and candidate defects in the ME project data. As an advantage the Ontology-based Cross-Disciplinary Defect Detection (OCDD) approach does not require the project participants to change their well-known tools and workflows.

The remainder of the paper is structure as follows. Section 2 presents related work; Section 3 describes the engineering practice in ME projects; Section 4 introduces the OCDD approach in the context of an industrial case study; Section 5 reports on preliminary evaluation within the case study at the industry partner; Section 6 discusses the implementation strategy of the proposed approach in ME settings; and Section 7 concludes the paper and depicts future work.

2 Related Work

This section summarizes related work on defect detection in multi-disciplinary engineering (ME) projects and the use of semantic web approaches in industrial contexts.

Defect Detection in ME Environments. Engineers of individual disciplines typically apply isolated and discipline-specific tools and data models while designing their artifacts. Within each discipline, well-defined approaches are used for quality assurance (QA) and engineering data analysis according to domain specific industry standards and best practices, e.g., simulation of electrical circuits and systems [16]; or static analysis methods (e.g., inspections or code analysis techniques [8]) and testing of software components [13]. However, in ME environments these isolated disciplines have to collaborate and exchange data – a major challenge for defect detection and QA. Defects and inconsistencies have to be detected as early as possible to minimize risks, solve conflicts, and improve product quality.

In [2] an openness metric for a systematic assessment of interoperability capabilities in ME projects has been applied to a set of selected tools. The main outcome was that isolated tools can support data exchange standards, e.g., XML or AutomationML[1] to enable collaboration between disciplines. However, loosely coupled tools do not support cross-disciplinary QA. Tool suites, e.g., Comos[2], typically cover a range of disciplines and provide functionality also for cross-disciplinary collaboration. These tools suites include a common data model that could enable basic QA tasks, e.g., defect detection. As for the drawbacks there may be some reduction of feature sets compared to isolated and highly specific solutions. In addition, project participants have to switch from their well-known tools to the all-in-one solution which might represent a barrier regarding acceptance, time and cost. Therefore, there is a need for approaches and tool-support for the engineers in ME projects to: (a) explicitly specify the cross-disciplinary interrelations between the heterogeneous engineering artifacts; (b) perform data analysis and detect defects across disciplines: and c) keep their customary engineering workflows and tools.

Semantic Web-Based Approaches. Semantic based solutions have been developed for various industries such as the aeronautical industry [1], the automotive industry [5], chemical engineering [11], heavy industries [15], factory automation [9] or, more broadly, manufacturing in general [10,14]. As opposed to these efforts, which typically focus on a single domain (with the notable exception of [4, 5]), we aim to support mechatronic systems, which, by definition combine the work of multiple engineering disciplines such as mechanical, electrical and software engineering. This combination of disciplines is a key differentiating feature of our efforts with respect to the predominantly mono-domain approaches reported so far in the literature.

The major goal of the various ontology based solutions reported so far is the integration and consolidation of engineering data [15, 19]. Such integration is a prerequisite to enable a range of intelligent applications such as decision support [1], automatic cost estimation [10], semantic-aware multiagent systems for manufacturing [11] or enterprise specific tasks such as bid analysis, market analysis and continuous process improvement by analyzing and integrating employee feedback [18]. The identification of errors, faults and defects across engineering models from different disciplines is an important but less frequent application and has only been reported by [4]. A mono-disciplinary defect detection system for identifying faults of electronic control units in cars has been achieved using ontologies and rules at AUDI[3].

[1] AutomatonML: `https://www.automationml.org`
[2] COMOS: `http://www.automation.siemens.com/mcms/plant-engineering-software`
[3] `http://www.w3.org/2001/sw/sweo/public/UseCases/Audi/`

3 The Engineering Process in Multi-Disciplinary Environments

Multi-disciplinary engineering (ME) projects, e.g., in the automation systems engineering (ASE) domain, typically require concurrent engineering [7]. Due to tight project delivery times and project constrains engineering teams work in parallel rather than sequentially, often in a geographically distributed setting. This requires adjusting the engineering process and including a set of synchronizations at each project stage to (a) propagate changes across disciplines and (b) identify early defects and inconsistencies across disciplines from concurrent changes in heterogeneous project data. Figure 1 shows the data synchronization process with defect detections mechanisms.

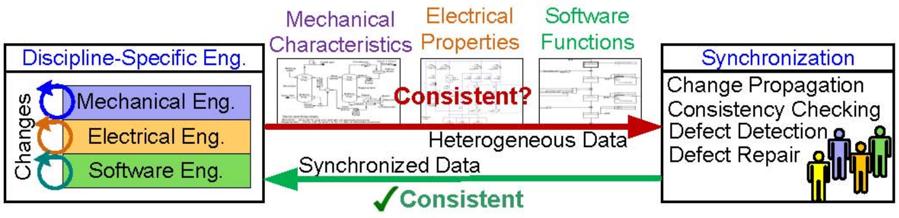

Fig. 1. Synchronization Process in Multi-Disciplinary Engineering Projects

The main aim of synchronization is addressing different types of heterogeneities that exist in ASE projects: technical heterogeneity (various engineering tools and technologies applied), semantic heterogeneity (dissimilar data models and formats), and process heterogeneity (tailored development processes). To manage these heterogeneities engineers from different disciplines work together during the synchronization on a) propagating changes that were made to data of a discipline since the last synchronization; and b) checking the consistency of engineering data across disciplines. If inconsistencies have been detected, a next step is to define actions and responsible roles for getting to a consistent state by defect repair or conflict resolution. Such checking is performed iteratively, until the data is proved to be consistent.

Two important types of defects affect engineering processes (a) *intra-disciplinary defects* (affect data within one discipline) and (b) *cross-disciplinary defects* (affect data in more than one discipline), e.g., changing a sensor (mechanical engineering) might have an effect on the related software component (software engineering) and might lead to defects if not addressed properly. While intra-disciplinary defects are usually discoverable by discipline-specific tools, cross-disciplinary defects are detected and fixed mainly during the synchronization phase. Because of the lack of tool support, cross-disciplinary data analysis is usually performed manually by project engineers, which is time consuming and error-prone.

Based on the described above, it is necessary in ME projects to provide efficient and effective mechanisms for defect detection. These mechanisms should be aware of the following requirements: (a) different disciplines involved; (b) large number of heterogeneous engineering artifacts; (c) cross-disciplinary dependencies in the project data; and d) concurrent engineering.

4 Automating Cross-Disciplinary Defect Detection: A Power Plant Automation System Case Study

This section presents the ontology-based cross-disciplinary defect detection (OCDD) approach for ME projects and introduces it within a case study at an industry partner. Figure 2 gives an overview on the approach and contrasts it with a traditional manual defect detection.

In traditional settings (lower part of Figure 2) the data and data models of the involved disciplines (e.g., mechanical, electrical and software engineering) are completely isolated and locked in discipline-specific tools using heterogeneous data models and data formats. Therefore all cross-disciplinary data exchange and analysis activities (including defect detection) rely on the knowledge of domain experts and require manual processes.

With the OCDD approach (upper part of Figure 2) an ontology layer is created for (1) capturing explicitly the discipline specific data models and knowledge in corresponding ontologies and (2) defining mappings between these ontologies corresponding to cross-disciplinary dependencies. This knowledge layer enables the automation of the defect detection process as follows. Firstly, domain experts formulate consistency checks focusing on the engineering objects that are important in more than one discipline. Secondly, a knowledge engineer translates the checks defined by the domain experts into corresponding SPARQL queries to be executed over the created ontological system, thus performing the cross-disciplinary data analysis and defect detection in an automated way.

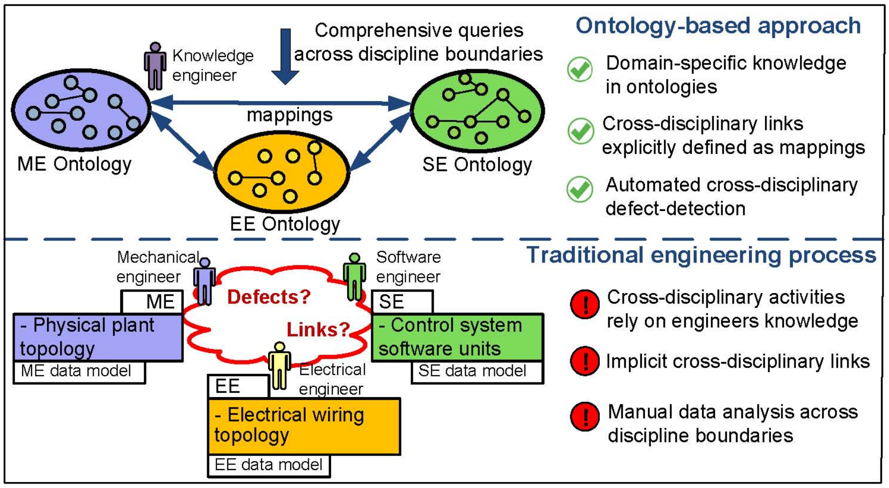
Knowledge engineer
Comprehensive queries across discipline boundaries
Ontology-based approach
Domain-specific knowledge in ontologies
mappings
Cross-disciplinary links explicitly defined as mappings
ME Ontology
SE Ontology
Automated cross-disciplinary defect-detection
EE Ontology
Mechanical engineer
Software engineer
Traditional engineering process
ME
Defects?
SE
Cross-disciplinary activities rely on engineers knowledge
- Physical plant topology
Links?
- Control system software units
ME data model
Implicit cross-disciplinary links
SE data model
Electrical engineer
EE
- Electrical wiring topology
Manual data analysis across discipline boundaries
EE data model

Fig. 2. Ontology-based Cross-Disciplinary Defect Detection (OCDD) vs. a traditional manual approach in ASE (ME – mechanical engineering; EE – electrical eng.; SE – software eng.).

We continue with an overview of a case study in a ME project at an industry partner, a large-scale industrial automation software provider (Sections 4.1- 4.3).

4.1 Case Study Overview

The case study was performed in the ME project concerning the development of a power plant and the corresponding control system. Engineering data from three domains were considered: (a) hardware configuration (HC) domain (sub-part of mechanical engineering), responsible for designing the physical plant topology; (b) control system (CS) domain (a sub-part of software engineering), responsible for developing PLC code for the corresponding control system; and (c) project configuration (PC) domain, responsible for managing the projects' data, people involved in the development and history of changes in the engineering data of these projects.

The following links, implicitly known by the project engineers, exist between these domains: (a) *variables* link CS and HC domains, i.e. for each device a set of global variables from the PLC code, are defined for its inputs and outputs; and (b) *artifacts* link the PC domain with HC and CS domains, i.e. an artifact can represent either a code unit or a software variable on a CS side; or a certain hardware device on a HC side.

4.2 Case Study: Knowledge Representation

We hereby describe the knowledge representation for the Hardware Configuration (HC), Control System (CS) and Project Configuration (PC) domains. All presented ontologies are OWL ontologies.

Hardware Configuration Ontology

The HC domain operates with data about devices, their inputs and outputs and variables that get and set values on certain inputs/outputs. Corresponding ontology was populated by transformation from the XML files obtained from the industry partner.

Figure 3 presents the part of the ontology relevant within the case study. Arrows denote the object properties. An important concept in the ontology is *Device*, which represents a specific device within a power plant (e.g., a temperature sensor). Specifying device information is stored via concept *DeviceDef*. Another important concept is *Variable*, which represents variables defined on devices' inputs and outputs. E.g., value measured by a specific temperature sensor will be set as a value of the corresponding variable. *VariableDef* comprises additional variable information, such as its size, type, and extra flags. Variables are grouped via *VarGroup*, which is assigned to a certain device. A specific device can have many variable groups assigned to it.

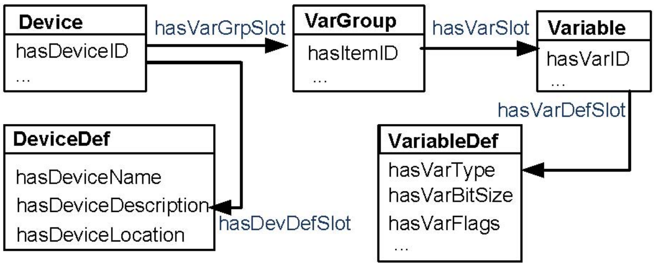

Fig. 3. Part of the Hardware Configuration Ontology relevant within the case study

Control System Ontology

The CS domain manages the control system software. Corresponding ontology (depicted in Figure 4) was obtained by transformation from the XML files. The data is compliant with the IEC61131-3 standard for representing programmable logic controllers.

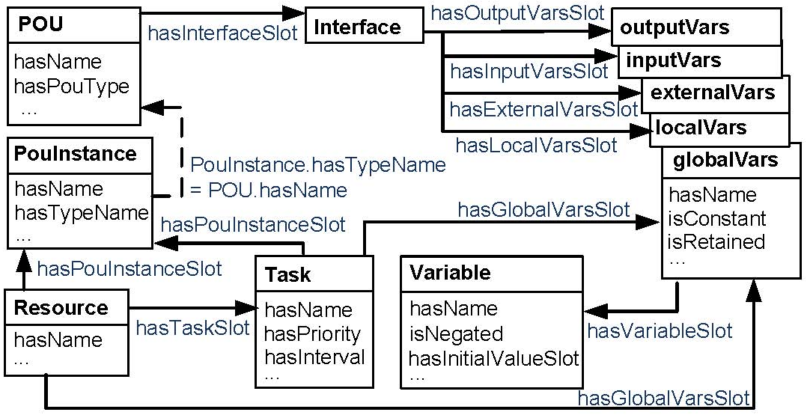

Fig. 4. Part of the Control System Ontology relevant within the case study

An important concept within this ontology is *POU* (program organization unit), which can represent a function, a function block (containing a set of functions) or a program (containing a set of functions and functions blocks). Every POU has an *Interface*, where the variables are declared. The *Variable* concept generalizes five different variable types: *global variable* (visible within all POUs), *local variable* (visible within one POU); *external variable* (referencing a global variable); *input variable* (transferred to a specific POU as an input parameter before execution); and *output variable* (takes a value after the POU execution). The concepts *globalVars*, *localVars*, *externalVars*, *inputVars* and *outputVars* comprise a set of variables of a corresponding type, which can be then assigned to a specific *Interface* (for local, external, input and output variables) or to a specific *Resource* or *Task* (for global variables).

Resource represents a device controller. For each resource a set of tasks are defined. *Task* combines a set of POUs with the execution configuration. To be executed in run-time *POU* must be assigned to a resource and/or task. This is done via concept *PouInstance* (similar to classes and their instances in Object Oriented Programming). POU instances exist only in run-time and refer to a name of a corresponding POU (a dashed arrow in Figure 4 depicts this connection that should have been represented via object property, but due to peculiarities of data representation in initial XML files, must be checked additionally, by comparing the string values of the properties *POU.hasName* and *PouInstance.hasTypeName*.

Project Configuration Ontology

The PC domain comprises more general project related information such as: engineering projects under development; project members and their responsibilities; and the history of changes in the engineering data of these projects. Figure 5 presents the ontology part relevant within the case study.

The *Project* concept represents an engineering project and comprises such details as its id, name, start and end dates and current stage. Engineers working in a project are represented via the concept *ProjectMember*. Every project member has a set of assigned responsibilities. *Responsibility* is a tuple relating a specific project with a specific project role. The following project roles are defined in the ontology: "RequirementsEngineer", "Modeler", "Developer", "Tester", "Validator" and "ProjectLeader". Thus, a project member can be a tester in one project and developer in another. *Artifact* represents various artifacts that are managed during the development (e.g. a specific variable or a piece of code). *Activity* represents a single change in the project engineering data and is used to store a history of changes. Following information is defined for each activity: who performed it? (refers to a *ProjectMember*); which artifact was influenced? (refers to an *Artifact*); what kind of activity it was? (refers to *ActivityType*, which can be "Create", "Delete", "Modify" or "Validate").

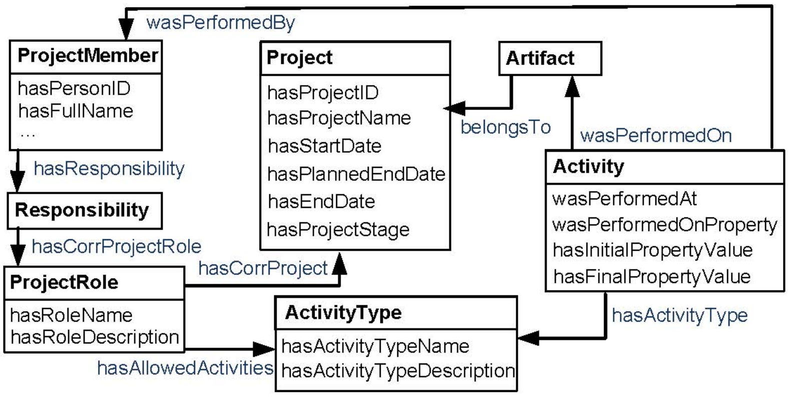

Fig. 5. Part of the Project Configuration Ontology relevant within the case study

4.3 Case Study: Cross-Disciplinary Defect Detection

In the case study we focus on checking the variable data across the HC, CS and PC domains. Variables are important engineering objects in ASE and often link several domains/disciplines. In particular, variables are defined on the devices inputs and outputs (HC domain); then the same variable data are included into the PLC code (CS domain); and finally there are changes made on variable data (PC domain). According to the industry partner up to 40,000 variables can be managed in a specific project [12], which is a hard task, if doing this with the traditional manual approach.

To make the relations between the variables in HC, CS and PC domains explicit, the following mappings have been defined: a) *CS:Variable* and *HC:Variable* are specified as subclasses of *PC:Artifact*; and b) *CS:Variable* is specified to be equivalent to *HC:Variable*. Although few and straightforward, these mappings already enable executing comprehensive checks on variable data across domain boundaries, which was not possible to perform automatically before. Below two sample cross-disciplinary checks are explained in details (SPARQL implementation can be found online[4]).

Q1: Which global variables on CS side are not declared on HC side? Each global variable declared in the control system software (CS ontology) should be declared on a specific device input or output in the hardware system topology (HC ontology). If a corresponding declaration is missing on the HC side, this might indicate two possible problems: a) either there is a redundant global variable (CS side); b) or a variable declaration is missing in the physical system topology (HC side).

Q2: Which changes on global variables declared at a certain device were not allowed by a project role of the project member that performed them? Each project member has a project role (or a set of them) that specifies which activities (s)he can perform in this project (e.g., a developer can create, modify and delete artifacts, while a validator can only validate). If there are doubts on the consistency of global variables in a project, one way to find the cause could be checking whether someone has performed an activity not allowed by his role. If such changes are found, corresponding global variables are the first candidates to be tested for consistency.

Of course, the range of possible checks over HC, SC and PC domains is not limited to variable data only. For instance, there is an implicit connection between the *CS:Resource* and *HC:Device* (both represent specific aspects of a hardware device) and at the same time they both can be seen as an *PC:Artifact.* After defining explicit mappings, it becomes possible to perform various checks concerning hardware devices across the domains (similar to those described above concerning variables).

5 Evaluation of the Case Study Results

In this section we discuss the results of prototypic implementation of OCDD approach obtained within the case study in the ME project at the industry partner.

We performed several interviews with the domain experts at the industry partner to ask for their feedback and opinion on the OCDD approach comparing with a traditional manual defect detection approach, with a particular focus on the foreseen benefits and limitations of applying the OCDD in practice. The interviews were semi-structured (acc. to [3]) and were performed in two stages: 1) requirements capture: the desired consistency checks for the case study were identified and formulated; and 2) approach validation: to validate that the captured requirements were efficiently addressed by the OCDD approach.

In the interviews, the industry experts were concerned about the additional modeling complexity introduced: it is necessary to specify an ontology for each discipline and the mappings between the ontologies. Since the domain experts at the industry partner do not possess such skills in their current setup, this also implies the need for a Knowledge Engineer to manage semantic technologies.

In spite of these concerns, the industry experts do believe that the approach provides valuable improvements: 1) the cross-disciplinary relations between the

[4] SPARQL implementation of the queries in this paper: 128.130.204.52/ekaw14-queries.html

engineering artifacts are explicitly specified and presented in machine-understandable form; 2) having the connections defined in a machine understandable format makes possible to identify the exact relevant data set (in the engineering data of other disciplines) that must be checked to solve a defect; 3) knowing the defect's origin, it is possible to identify how this defect was produced, and therefore, take measures to avoid it in the future; 4) defects across the disciplines can be detected automatically, leading to significant time savings and higher recall compared to the traditional manual defect detection approach. Experts at the industry partner also estimated the overall effectiveness of the OCDD approach as being higher due to the fact that the process relies on formally defined parameters (data models and relations between them) and not on subjective human-based estimation.

6 Discussion

This section discusses in details specific aspects of the implementation strategy of the proposed OCDD approach.

Domain Knowledge Modeling. Ontology enables gathering knowledge from the heterogeneous data sources and tools within the domain (e.g. CAE tools that correspond to the mechanical engineering) and making it explicit. However, this means extra time and complexity for modeling the domain knowledge in terms of classes, relations and axioms. This step requires close collaboration of the engineers of the ME project and knowledge engineer(s), as the domain experts typically have no expertise in the semantic web technologies. From the positive side, knowledge captured by the ontology is reusable, flexible and customizable and, therefore, can be easily used to implement the cross-disciplinary defect detection in next projects.

Data Import. Most of the engineering tools allow their data to be exported in one of the widely used formats, e.g. XML or spreadsheet data. In this case, there are tools already available that support automated data transformation from these formats into ontology (for an instance, XMLTab[5] and MappingMaster[6]). In the worst case, if the data can be only obtained in a proprietary tool format, one converter has to be implemented for each engineering tool. Although this requires extra effort from the project engineers, once the converters have been implemented they can be reused for the next projects, as the tool data model typically is stable and do not change with time.

Comprehensive Querying. Semantic technologies provide high expressiveness and enable (in contrary to UML diagrams and SQL queries) the possibility to create mappings between different models and to query the different models (corresponding to disciplines) at the same time. For the time being, cross-disciplinary data analysis and defect detection are mainly performed manually in industrial ME practice. For instance, several engineers from different disciplines (e.g. mechanical engineering and software engineering) typically spend several days to complete one or at most several specific consistency checks for their engineering artifacts. Having in mind that a number of checks are needed during the synchronization and at least several syn-

[5] `http://protegewiki.stanford.edu/wiki/XML_Tab`
[6] `http://protege.cim3.net/cgi-bin/wiki.pl?MappingMaster`

chronizations are needed during the ME project development, the ability to automatically execute checks (encoded in SPARQL queries) will significantly reduce the time and efforts to perform the cross-disciplinary data analysis in a ME project.

Limited Scalability of Ontologies. Semantic web technologies provide an expressive and explicit way to define domain knowledge, mappings, and queries. However, the performance of the data storage and the memory usage become challenging when managing larger datasets (e.g., millions of instances) [17]. Especially high computational load is to be expected when it comes to reasoning and querying for large instance data sets [19]. Therefore, the difficulty of using semantic technologies is the need for very powerful computer hardware to perform within reasonable time. In future work, we would like to analyze how a selection of specific mapping and querying techniques and/or technologies can help mitigating these problems.

Domain-Expert Support. In the current approach, a knowledge engineer is needed to work with semantic web technologies since domain experts do not typically have the appropriate skills. This could be mitigated by creating a GUI that hides the technological details and allows the domain experts performing the needed activities to manage the project data (e.g., data import, creation of mappings, and querying). This will also encourage the usage of the system from non-experts in semantic web.

7 Conclusion and Future Work

In the multi-disciplinary engineering projects (ME) participants from different engineering disciplines collaborate to deliver a high-quality end product. Typically, the disciplines are rather isolated and use heterogeneous data models to represent common concepts in the project team. Therefore, it is difficult to efficiently analyze data and perform defect detection activities across the disciplines. In this paper we introduced the ontology-based cross-disciplinary defect detection (OCDD) approach that supports automated cross-disciplinary defect detection. We presented the preliminary evaluation based on prototypical implementation of the OCDD approach in a case study ME project at an industry partner, an industrial automation software provider. Major result was that the OCDD approach was found useful in the evaluation context and more efficient than traditional manual defect detection if heterogeneous data originating from different engineering disciplines have to be handled.

As future work we plan a) to align our domain ontologies to existing domain-agnostic, core engineering ontologies that model concepts and patterns suitable for representing any type of engineering knowledge (e.g. MASON [10] and OntoCAPE [11]); and to existing ontologies partially covering one of the domains of interest (e.g., the W3C Organization ontology[7] partially covers Project Configuration domain); b) to investigate how the selection of a certain mapping and querying mechanism influences the efficiency of the approach; and c) to provide a user-friendly interface for data querying to hide the complexity of SW technologies from domain experts.

[7] `http://www.w3.org/TR/vocab-org/`

References

1. Adams, T., Dullea, J., Clark, P., Sripada, S., Barrett, T.: Semantic integration of heterogeneous information sources using a knowledge-based system. In: Proceedings of 5th International Conference on Computer Science and Informatics (CS&I 2000), Citeseer (2000)
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 39th Annual Conference of the IEEE Industrial Electronics Society, IECON 2013, pp. 6844–6849. IEEE (2013)
3. Gray, D.E.: Doing research in the real world. Sage (2009)
4. Hästbacka, D., Kuikka, S.: Semantics enhanced engineering and model reasoning for control application development. Multimedia Tools and Applications 65(1), 47–62 (2013)
5. Hefke, M., Szulman, P., Trifu, A.: An ontology-based reference model for semantic data integration in digital production engineering. In: Proceedings of the 15th eChallenges Conference. Citeseer (2005)
6. Kovalenko, O., Winkler, D., Kalinowski, M., Serral, E., Biffl, S.: Engineering process improvement in heterogeneous multi-disciplinary environments with the defect causal analysis. In: Proceedings of the 21st EuroSPI Conference (2014)
7. Kusiak, A.: Concurrent engineering: automation, tools, and techniques. John Wiley & Sons (1993)
8. Laitenberger, O., DeBaud, J.M.: An encompassing life cycle centric survey of software inspection. Journal of Systems and Software 50(1), 5–31 (2000)
9. Lastra, J.L.M., Delamer, I.M.: Ontologies for production automation. In: Advances in Web Semantics I, pp. 276–289. Springer (2009)
10. Lemaignan, S., Siadat, A., Dantan, J.Y., Semenenko, A.: MASON: A proposal for an ontology of manufacturing domain. In: IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications, DIS 2006, pp. 195–200. IEEE (2006)
11. Morbach, J., Wiesner, A., Marquardt, W.: OntoCAPE - a (re) usable ontology for computer-aided process engineering. Computers & Chemical Engineering 33(10), 1546–1556 (2009)
12. Mordinyi, R., Winkler, D., Moser, T., Biffl, S., Sunindyo, W.D.: Engineering object change management process observation in distributed automation systems projects. In: Proceedings of the 18th EuroSPI Conference, Roskilde, Denmark (2011)
13. Naik, S., Tripathy, P.: Software testing and quality assurance: theory and practice. John Wiley & Sons (2011)
14. Obitko, M., Marik, V.: Ontologies for multi-agent systems in manufacturing domain. In: Proceedings of the 13th International Workshop on Database and Expert Systems Applications, pp. 597–602. IEEE (2002)
15. Peltomaa, I., Helaakoski, H., Tuikkanen, J.: Semantic interoperability-information integration by using ontology mapping in industrial environment. In: Proceedings of the 10th International Conference on Enterprise Information Systems – ICEIS 2008, pp. 465–468 (2008)
16. Sage, A.P., Rouse, W.B.: Handbook of systems engineering and management. John Wiley & Sons (2011)
17. Serral, E., Mordinyi, R., Kovalenko, O., Winkler, D., Biffl, S.: Evaluation of semantic data storages for integrating heterogeneous disciplines in automation systems engineering. In: 39th Annual Conference of the IEEE Industrial Electronics Society, pp. 6858–6865 (2013)
18. Uschold, M., King, M., Moralee, S., Zorgios, Y.: The enterprise ontology. The Knowledge Engineering Review 13(01), 31–89 (1998)
19. Wiesner, A., Morbach, J., Marquardt, W.: Information integration in chemical process engineering based on semantic technologies. Comp. & Chem. Eng. 35(4), 692–708 (2011)