Chapter 7
Knowledge Change Management and Analysis in Engineering

Fajar Juang Ekaputra

Abstract Knowledge is changing rapidly within the engineering process of Cyber-Physical Production Systems (CPPS) characterized by the collaborative work of engineers from diverse engineering disciplines. Such rapid changes lead to the need for management and analysis of knowledge changes in order to preserve knowledge consistency. Knowledge change management and analysis (KCMA) in Multidisciplinary Engineering (MDEng) environments is a challenging task since it involves heterogeneous, versioned, and linked data in a mission-critical fashion, where failure to provide correct data could be costly. Although, there are several available solutions for addressing general issues of KCMA, from fields as diverse as Model-Based Engineering (model co-evolution), Databases (database schema evolution), and Semantic Web Technology (ontology versioning), solving KCMA in engineering remains a challenging task. In this chapter, we investigate issues related to KCMA in MDEng environments. We provide a definition of this task and some of its challenges and we overview technologies that can be potentially used for solving KCMA tasks from the three research fields mentioned above. We then define a technology agnostic solution approach inspired by the Ontology-Based Information Integration approach from Semantic Web research as a first step toward a complete KCMA solution and provide an indication of how this solution concept could be implemented using state of the art Semantic Web technologies.

Keywords Knowledge change management and analysis • Ontology evolution • Ontology versioning • Ontology-based information integration • Change detection • Change validation • Change propagation • Multidisciplinary engineering

F.J. Ekaputra (✉)
Institute of Software Technology and Interactive Systems, CDL-Flex,
Vienna University of Technology, Vienna, Austria
e-mail: fajar.ekaputra@tuwien.ac.at

S. Biffl and M. Sabou (eds.), *Semantic Web Technologies for Intelligent Engineering Applications*, DOI 10.1007/978-3-319-41490-4_7

7.1 Introduction

The process of designing a Cyber-Physical Production System (e.g., modern power plants or steel mills) often requires teams of engineers from diverse engineering domains (e.g., mechanical, electrical and software engineering) to work together. As a result, this design process typically takes place in a multidisciplinary engineering (MDEng) environment, in which experts from various engineering domains and organizations work together toward creating complex engineering artifacts (Serral et al. 2013). Figure 7.1 depicts such a typical MDEng setting. Domain specific engineers (shown on the right hand side) use their own tools to create models that represent parts of the final system. Therefore, the MDEng environment is highly heterogeneous, as it involves a wide range of data models, processes, and tools that were originally not designed to cooperate seamlessly. Despite this situation, as shown on the left-hand side of Fig. 7.1, other engineers and project managers need to perform tasks that require access to project-level data as opposed to domain specific data alone. For these actors, there is a need for accessing integrated data at project level. In response to this need, knowledge engineers aim to integrate models and data from different engineering domains based on the requirements and feedback of engineers and project managers.

In addition to the characteristics described above, the process of designing complex mechatronic objects, such as CPPS, requires iterations and redesign phases, which lead to continuous changes of the data and knowledge within the MDEng environment. To deal with these changes, industrial partners need to keep data versions, move backwards to previous versions, and query different versions of large data (schema and instances) from heterogeneous local data sources. Furthermore, the effective and considerate propagation of changes is essential to ensure

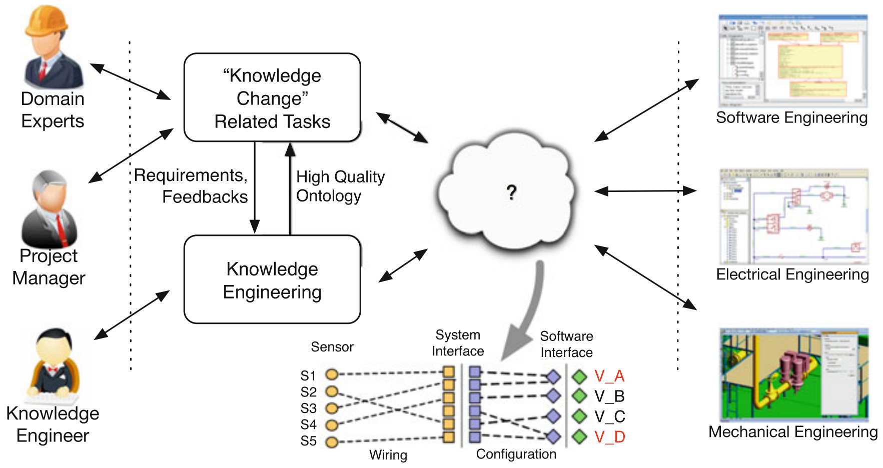

Fig. 7.1 Problem setting for KCMA in MDEng environments

a consistent view of the project, to minimize defects and risks, as well as facilitate acceptance of new solutions by domain experts. To achieve this, changes originating from one engineering discipline need to be communicated and coordinated with participants of other disciplines, where those changes are relevant. Ideally, this communication should focus on high-level changes (e.g., defined in terms of domain concepts such as “Motor X updated to new version”) as opposed to low-level individual changes (i.e., change operations on versioned files) to ease the data analysis process. To cater for all these needs, a process and tool support should be available for knowledge change management and analysis (KCMA) within MDEng environments.

Technology solutions for data integration and knowledge change management in general, and to some extent for multidisciplinary engineering in particular, have been proposed by research fields as diverse as Database Systems, Model-Based Engineering (MBE), and the Semantic Web. Although the general strengths and weaknesses of these solution approaches are somewhat known, a precise comparison of how well they could support KCMA in the MDEng environments is hampered by two major factors. First, there is a lack of understanding of requirements for KCMA in MDEng, i.e., a characterization of the problem that needs to be solved. Second, a baseline setting that would allow an objective comparison of these technologies is also missing.

To overcome these shortcomings, this chapter makes two important contributions. First, it provides a characterization of KCMA by means of key requirements that should be fulfilled (Sect. 7.2). These requirements were derived from concrete industry specific projects where the author investigated the need for knowledge change management. Second, this chapter provides a brief introduction in typical solutions to knowledge change management from the areas of Database Systems, Model-Based Engineering (MBE), and the Semantic Web (Sects. 7.3 and 7.4). Based on this analysis spanning the three major research fields of interests, the chapter provides a technology agnostic reference process for solving KCMA (Sect. 7.5). This reference process is suitable to play the role of a baseline for comparing the strengths and weaknesses of implementations relying on either of the three major families of technologies described in this chapter. Additionally, a proposal of implementation based on Semantic Web technologies is presented in Sect. 7.6. We sum up the chapter and discuss future work in Sect. 7.7.

7.2 KCMA in Engineering

In this section, we take a detailed look to the needs for KCMA in engineering settings. To that end, we provide an illustrative real-world example (Sect. 7.2.1) and then discuss a set of requirements for KCMA in Sect. 7.2.2.

7.2.1 KCMA Example

An illustrative MDEng setting is the engineering of a modern power plant. Similar to any other large-scale project, the development of a power plant requires coordinated work of engineers from multiple disciplines that needs to converge into a high-quality product. Such a heterogeneous team of experts should be coordinated in a way that fulfills important technical and management project-level constraints (e.g., the mass and dimension constraints of the base plate are not exceeded by individual equipment). Such coordination requires aggregating relevant data across teams from various disciplines, but it is hampered by the semantic heterogeneity of the data, with different disciplines possibly using diverse terms to refer to the same entities.

For illustration purposes, we provide an excerpt of the data (usually called a "signal list") exchanged between the engineers participating the engineering process of a hydropower plant (Table 7.1). A signal list is typically serialized and used by engineers as spreadsheet files. The header of Table 7.1 represents the data schema used within the signal list, while its body represents data instances. The combination of the first four columns (ANR, L1, L2, and SIG) identifies the engineering signals/objects, while the later three (DIFF MAX, GRADIENT MAX, and RESISTANCE VALUE) represent signal/object properties.

To add more complexity, the MDEng project models and the project data change over time due to:

1. **Changes in the represented domains**, such as the introduction/removal of domain concepts (e.g., the removal of RESISTANCE VALUE column from Table 7.1 since it is not relevant anymore in the domain) or granularity changes (e.g., adding more detailed information than at signal level);
2. **Changes in the underlying data sources**, such as when new data elements become available and old data elements become obsolete (e.g., a new spreadsheet file is produced to replace the old file without changing the data schema); or

Table 7.1 Excerpt of the engineering data of a hydropower plant engineering process

ANR	L1	L2	SIG	S3 DIFF MAX	S3 GRADIENT MAX	RESISTANCE VALUE
0	BAA30	GS100	XB01	0.025	0.8	500
0	BAA30	GS191	YB01	0.025	0.8	500
0	BAA30	GS191	YB02	0.025	0.8	500
0	BFB10	GS100	XB01	0.025	0.8	500
0	BFB10	GS100	XM01	0.025	0.8	500
0	BFB10	GS100	YB01	0.025	0.8	500

3. **Changes in the intended use of the models and data**, such as by changing requirements of the currently supported tools or the design of new tools (e.g., a new data schema is introduced and it has to be mapped into the old schema in Table 7.1).

To address such changes, an integrated versioning of the MDEng data needs to be prepared for facilitating this evolution and the consequent data transformations and propagation, according to the evolved model.

7.2.2 Requirements for KCMA in Engineering

Dealing with the types of possible changes described above requires both activities for managing and analyzing changes. In terms of change management, it is important to record data versions and to be able to move backwards to previous versions, as well as to query different versions of integrated data (schema and instances) originating from heterogeneous local data sources.

Furthermore, on a more analytics related level, it is essential to enable the effective and considerate propagation of changes across data from different disciplines. This will ensure a consistent view of the project and it will minimize defects and risks. To achieve this, changes originating from one discipline need to be communicated and coordinated with participants of other disciplines, where those changes are relevant, especially in cases when data from different engineering disciplines is closely interlinked. The KCMA approach should also provide high-level change definitions instead of low-level ones to ease the analysis process of data.

Based on our involvement and experiences in several industrial engineering settings where KCMA was required, we have identified a set of requirements and characteristics of KCMA in engineering (specifically in MDEng environments) as follows:

1. **Closely interlinked knowledge**. In the engineering process of a CPPS, engineering models and data created by different engineering disciplines reflect diverse views on the same system and are therefore naturally interlinked (e.g., an electrical signal activates a mechanical component as a result of executing a certain software routine). Therefore, knowledge changes within one discipline may require changes in other disciplines too due to this relation between data in different disciplines. For example, a signal change in electrical engineering area will require the adaptation of the corresponding machinery part (mechanical engineering) or reprogramming of the relevant software components (software engineering).
2. **Large amounts of data**. Engineering projects typically deal with large amounts of data required to describe any complex system. For example, an average size power plant design data contains data about hundreds of thousands to tens of

millions of signals. This already large data size is further multiplied due to many iteration processes during the design time of the system, which should all be versioned and stored (Mordinyi et al. 2014).

3. **Changes in schema and instances**. In MDEng environments, both data models (i.e., schema) and actual data (instances) is likely to change. Indeed, the heterogeneity of data sources within MDEng environments and the environment's dynamism means that additional tools could be added anytime, which may imply changes in the data models of all engineering disciplines involved. At the same time, data instances (e.g., signals with changed characteristics, added, or deleted signals) within MDEng environment will change even more frequently due to revisions and engineering process iterations.
4. **Change validation support**. Given the mission-critical nature of projects in MDEng environments, domain experts and engineers do not want to fully rely on automatic change validation mechanisms (e.g., to decide whether changes initiated by one discipline will break the overall consistency of the project wide data). Therefore, instead of fully automated change validation, the involvement of domain experts in the validation workflow is important for making critical decisions about changes.
5. **High-level change definition and detection**. Typical tools currently used in individual engineering disciplines are able to produce report data that consists of signal lists that represent those parts of a CPPS, which these specific tools handle (Vogel-Heuser et al. 2014). The differences between two versions of signal lists represent changes between them. However, it is challenging for a project manager to grasp the meaning of such low-level changes in data, i.e., signal changes. Instead, they would highly benefit from changes to data being presented in a more meaningful manner as high-level changes. Such presentation could be achieved in terms of domain level common concepts, e.g., relocation of specific engine to different machine rack.
6. **Support for data evolution and versioning**. A KCMA approach should be able to address both data evolution and data versioning. Data evolution focuses on how to modify the data in response to the changes in the surrounding. Data versioning covers the management of different versions of data schema caused by the data evolution (Noy and Klein 2004; Roddick 1995).

7.3 Solutions for KCMA in the Engineering Domain

This section introduces brief summaries of approaches related to the KCMA in the engineering domain from the Database systems (Sect. 7.3.1) and Model-Based Engineering research communities (Sect. 7.3.2).

7.3.1 Database Schema Evolution and Versioning

One of the earliest works concerning knowledge change management is reported in the field of database systems. Roddick summarized the issues of schema evolution and versioning in database systems (Roddick 1995). He explains that change management is closely related to data integration, claiming that both areas are the flavors of a more generic problem: using multiple heterogeneous schemata for various tasks. To solve the issues suggested by Roddick, there were several proposed conceptual approaches. One of them is the Hyper-graph Data Model (HDM), which targets schema evolution for heterogeneous database architectures (McBrien and Poulovassilis 2002). The HDM schema consists of Nodes, Edges, and Constraints. Nodes and Edges in the schema define a labeled (Nodes require unique names), directed (the Edges may link sequences of Nodes or Edges), and nested (Edges could link an unlimited number of other Nodes and Edges) hypergraph, while Constraints define a set of Boolean valued queries over the HDM schema.

In the field of MDEng, there are limited concrete solutions that are utilizing relational databases as the basis for managing and analyzing changes in engineering data from multiple engineering datasets. One exception is the Engineering Database (EDB), a solution based on relational databases that was introduced as an attempt to provide versioning management of engineering data using database technology (Waltersdorfer et al. 2010). The Engineering Database is a concrete implementation of Engineering Knowledge Base (EKB) that is explained in Chap. 4. The EDB stores engineering data as a flat database table, consisting of objects, properties, values, and important metadata information such as change commit information and provenance from the original data sources. The approach is capable of handling closely linked knowledge from different engineering disciplines.

To conclude, the maturity of database approaches in general provides a solid basis for handling change management of a large number of data. Additional KCMA-related solutions from database systems, not covered in this chapter, are also worth further investigations.

7.3.2 Model-Based Engineering (MBE) Co-Evolution

A lot of attention has been given to the comparison and versioning of software models in the Model-Based Engineering (MBE) research community, with more than 450 papers written in this area[1], covering various topics such as change modeling and definition of evolution frameworks. Similar to other research areas, the domain of Engineering is not the main application area of these works. An exception is the work of Göring and Fay, which proposes a metamodel language for modeling temporal changes of automation systems together with their physical and

[1]http://pi.informatik.uni-siegen.de/CVSM.

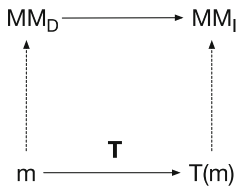

Fig. 7.2 Basic co-evolution schema, adapted from (Meyers and Vangheluwe 2011)

functional structures (Goring and Fay 2012). Their work extends the IEC 81346 standard, which already includes product, function, and location aspects. In the approach, however, they do not explain how to map their metamodel to other metamodels that are potentially used in the same system and therefore it is not clear how data integration is achieved.

Another line of MBE work focuses on change propagations of model variants of a single metamodel, to ensure the consistency of changes as well as the adoption of relevant changes in different model variants (Kehrer et al. 2014). Recently, the authors of (Berardinelli et al. 2015) provide means of a prototype-based model co-evolution, showing the capability of providing various levels of validation configuration to be applied in a top-down co-evolution approach.

Meyers and Vangheluwe propose one of the most recent frameworks for evolution of model and modeling languages, which claim that any possible model co-evolution could be derived as a composite of the basic co-evolution schema shown in Fig. 7.2 (Meyers and Vangheluwe 2011). It consists of a model *m* that conforms to metamodel domain (MM_D). Model *m* needs to be transformed into T (*m*) via transformation T, which again will conform to image metamodel (MM*i*). This co-evolution framework could theoretically address the requirements of knowledge change management within closely linked discipline data.

In conclusion, these works from the MBE research community partially address the requirements of KCMA in MDEng environment as mentioned in Sect. 7.2.2. Benefits of using MBE techniques for knowledge change management in engineering projects include the availability of a good tool support and solid theoretical frameworks (Meyers and Vangheluwe 2011; Taentzer et al. 2014).

7.4 Semantic Web for KCMA in Engineering

Advances in Semantic Web research could also provide partial solutions to the challenges of KCMA in MDEng discussed in Sect. 7.2. For example, the Provenance Ontology (PROV-O[2]), a recent W3C recommendation, is highly suitable to represent and interchange provenance information and could be used to record which changes were performed by which stakeholders.

[2]https://www.w3.org/TR/prov-o/.

An important observation is that, in MDEng environments, the *integration* of heterogeneous data sources is a prerequisite to addressing KCMA challenges and that knowledge change management must be applicable over the totality of integrated data sources that represent the data of an engineering project.

In the Semantic Web area, such integration settings can be addressed by relying on the Ontology-Based Information Integration (OBII) approach (e.g., in Calvanese et al. 2001; Wache et al. 2001), which provides a solution approach for integrating data from heterogeneous data sources using Semantic Web technologies. There are three different ways of applying the OBII approach as described by Wache et al. (2001).

1. **Single Ontology approach**, which relies on a single ontology for integrating all data sources. The approach is typically hard to maintain since it is susceptible to changes in each information sources. In other words, the ontology must be updated anytime a change in a data source occurs and then its compatibility with the other data sources must be ensured.
2. **Multiple Ontology approach** is characterized by defining one ontology per data source and subsequently linking the ontologies with each other independently. The drawback of this approach is that the mappings between different ontologies are challenging to define and maintain, due to the different granularities and aggregation between ontologies.
3. **Hybrid approach**, which combines the benefits of both single and multiple ontology-based approaches thus eliminating most of their drawbacks. An example of a Hybrid-style OBII system used to integrate engineering data of a power plant is shown in Fig. 7.3 and consists of three components. First, local ontologies represent data specific to one engineering discipline, i.e., local data. Second, a common ontology represents the aggregation of relevant and related concepts at the organizational level, e.g., power plant. Third, mappings between local and common ontologies enable linking and integration between the heterogeneous data sources. Similar to (Calvanese et al. 2001), we will use the term OBII to refer to the Hybrid OBII approach, unless stated otherwise.

The OBII approach aims to solve data integration and, as such, by itself it does not provide support for KCMA, which is essential for MDEng environments. Ontology change management has been investigated as a generic problem, not in conjunction with data integration, to address the dynamics of ontology data and its derivative challenges (e.g., Klein 2004; Noy et al. 2006; Stojanovic 2004; Zablith 2011).

However, it is not clear how applicable the currently available generic ontology change management solutions are to improve the KCMA support of an OBII approach in the context of MDEng environment. Therefore, the availability of a generic reference process of KCMA in MDEng environment is crucial to assess whether these ontology change management approaches sufficiently address the

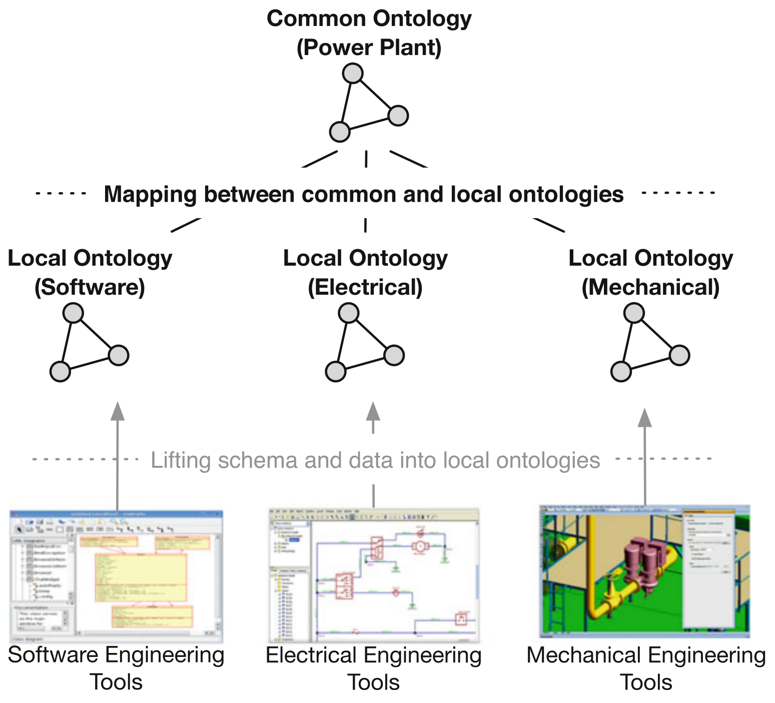

Fig. 7.3 Example of an OBII system for integrating engineering data from a power plant

KCMA requirements (e.g., change propagation to the overlapped data in other disciplines). In particular, an implementation of the KCMA process with ontology change management approaches would allow verifying their suitability to support KCMA in MDEng environments.

7.4.1 Ontology Change Management

The Semantic Web community has performed significant research in the area of ontology change management. In this section, we summarize works in this area based on our literature study (described in more detail in Ekaputra et al. 2015) and in terms of the requirements stated in Sect. 7.2. Table 7.2 provides an overview of the extent to which each requirement is addressed by each work we overview.

1. **Closely interlinked knowledge**. Closely interlinked knowledge is a condition where changes in one ontology within a system of interlinked ontologies may require propagation of the changes to the linked ontologies to maintain the global validity of the knowledge. This is not the typical setting for KCMA in Semantic Web community, which primarily focuses with open web data. This

Table 7.2 Semantic web generic solution alternatives for addressing knowledge change requirements within MDEng environment

	Type of interlinked knowledge			Amount of data	Changes in schema and instances		Change validation		Changes definition and detection		Support for data evolution and versioning	
	Single ontology	Loosely interlinked knowledge	Closely interlinked knowledge	Handle >1 M data points	Instance changes	Schema changes	Semi-automatic	Automatic	Low-level changes	High-level	Ontology evolution	Ontology versioning
MDEng KCMA			X	X	X	X	X		X	X	X	X
Klein (2004)		X				X					X	
Stojanovic (2004), Stojanovic et al. (2002)		X	(X)		X	X	X	X			X	X
Noy et al. (2006)	X					X			X		X	
Papavassiliou et al. (2009)	X			(X)	X	X			X	X	X	
Gröner et al. (2010)	X					X			X	X	X	
Zablith (2011)	X					X		X			X	
Vander Sande et al. (2013)		X			X				X			X
Horridge et al. (2013)		X		X	X	X	(X)		X		X	(X)
Graube et al. (2014)	X			(X)	X				X			X

[a]*X* fully supported; *(X)* partially supported

difference is reflected within most of traditional KCMA that focused on change management in a single ontology (Graube et al. 2014; Gröner et al. 2010; Noy et al. 2006; Papavassiliou et al. 2009; Zablith 2011) or multiple loosely interlinked ontologies (Horridge et al. 2013; Klein 2004; Redmond et al. 2008; Vander Sande et al. 2013), where changes in an ontology are independent and do not have to be propagated in order to maintain the validity of overall knowledge within a system. The work of Stojanovic is an exception to this trend, where she provided an attempt to propagate changes to relevant ontologies (Stojanovic 2004). However, her work is not further continued.

2. **Large amounts of data**. Horridge et al. provide an answer to the large-scale challenge of the changed data by introducing binary formats for storing ontology data and differences between ontology versions (Horridge et al. 2013). Their approach is claimed to handle more than one million triples. A different approach is adopted by Graube et al., where named graphs are used to store changes and ontology versions (Graube et al. 2014). Their approach did not scale well for change data analysis, since the query performance on the change data dropped significantly after several thousands of triples. Papavassiliou et al., on the other hand, successfully experimented their approach on almost 200 k triples (Papavassiliou et al. 2009). While the current approaches seem promising, given the closely coupled nature of the engineering data, these approaches need to be reevaluated in order to asses their feasibility.
3. **Changes in schema and instances**. Instance changes are required for addressing changes in the underlying data sources, whereas schema changes are crucial for addressing changes in the represented domains and changes in the intended use of the models and data, as previously mentioned in Sect. 7.2. Several ontology change management approaches are already able to deal with both schema and instance level changes (Horridge et al. 2013; Papavassiliou et al. 2009; Stojanovic 2004), where other approaches either focus on schema (Gröner et al. 2010; Noy et al. 2006; Zablith 2011) or focus on instances (Graube et al. 2014; Vander Sande et al. 2013).
4. **Change validation support**. This aspect of validation provides a mechanism to ensure the validity of data changes according to a predefined set of validation rules (automatic validation) or in combination with domain experts' involvement according to certain workflows (semi-automatic validation). Several approaches already support automatic change validations (Horridge et al. 2013; Stojanovic 2004). Furthermore, there are approaches from general Semantic Web concerning data validation and linked data quality, e.g., RDFUnit (Kontokostas et al. 2014) and Shape Expression (Boneva et al. 2014) that can be adapted to support ontology change validation. In the direction of semi-automatic validation, Stojanovic et al. proposed a mechanism to involve domain experts to check the semantic validity of ontology changes over multiple ontologies (Stojanovic et al. 2002). This involvement of stakeholders is indeed important in the MDEng environment due to the mission-critical characteristic of the domain, as we previously mentioned in Sect. 7.2.1.

5. **High-level change definition and detection**. One of the goals of KCMA is to provide stakeholders with a better decision support system. The high-level change definition and detection process helps to achieve this goal by providing a mean to detect and encapsulate atomic changes into more meaningful and higher level changes in terms of domain concepts, which are easier to understand, especially to non domain experts. In this regards, Papavassiliou et al. have developed an algorithm to support the detection of high-level changes from low-level changes, simplifying the effort to analyze changes in large datasets and without compromising performance (Papavassiliou et al. 2009). Alternatively, Gröner et al. used a subset of OWL-DL reasoning to recognize high-level change patterns (Gröner et al. 2010). One of the prerequisites for high-level change definition and detection is the formalization of low-level changes. This formalization can be achieved using triple patterns (Papavassiliou et al. 2009; Gröner et al. 2010; Vander Sande et al. 2013; Horridge et al. 2013; Graube et al. 2014) or specialized ontologies (Papavassiliou et al. 2009; Palma et al. 2009).
6. **Support for data evolution and versioning**. Ontology evolution (i.e., how to modify the data according to relevant changes in the surrounding) and versioning (i.e., management of different versions of data schema and instances caused by ontology evolution) are both important to the KCMA process and should be available and easily accessible by relevant stakeholders. Most of the ontology change management approaches focus either on ontology evolution (Gröner et al. 2010; Klein 2004; Noy et al. 2006; Zablith 2011) or on ontology versioning (Graube et al. 2014; Vander Sande et al. 2013). The rest of the approaches we surveyed try to address both ontology evolution and versioning (Stojanovic 2004; Horridge et al. 2013).

To conclude, parts of KCMA requirements in MDEng environment are already well explored in Semantic Web research. Schema and instance changes, for example, are addressed already by most approaches. Likewise, approaches for change detection, ontology evolution and ontology versioning are well researched and reported, providing ample options to choose from. However, due to the open nature of web data, approaches for ontology changes in closely interlinked knowledge settings are rarely investigated. Similarly, approaches for handling changes of large amounts of data and validating changes are currently limited, probably since these aspects are not the focus in current ontology change management research. There are options to use general ontology validation approaches for ontology change validation, i.e., by adapting RDFUnit (Kontokostas et al. 2014) or Shape Expression (Boneva et al. 2014) approach, but these adaptations are not yet seen as integral part of general ontology change management approaches. We therefore see the need to advance and combine existing approaches such that all KCMA requirements are sufficiently addressed. The KCMA reference process presented next provides a conceptual framework to guide this process.

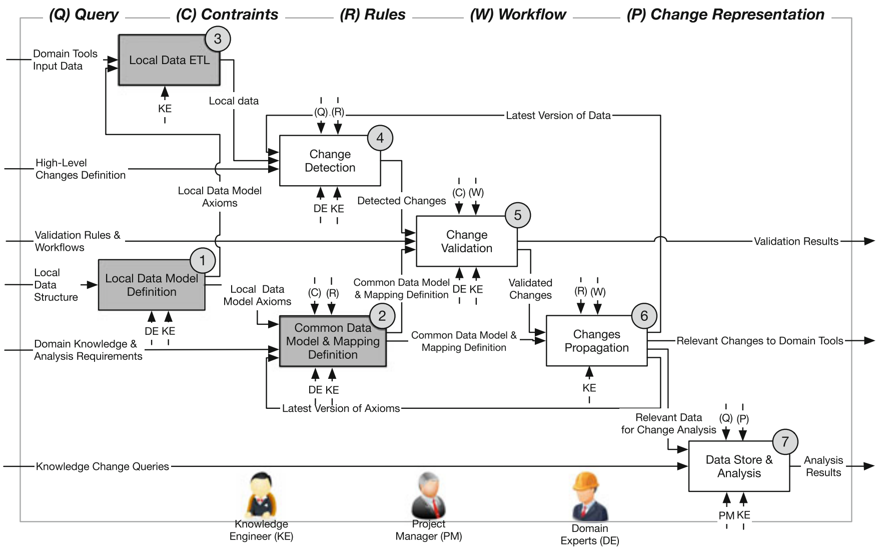

Fig. 7.4 Reference process model for KCMA in the MDEng environment. The *white boxes* extend a typical OBII process (*gray boxes*)

7.5 Reference Process for KCMA in MDEng Environment

In order to address the challenges of providing support for KCMA in MDEng environments, we propose a generic reference process shown in Fig. 7.4. This is a technology agnostic process that could be implemented with technologies drawn from any of the three areas we previously discussed. Implementations using different techniques but following this reference process will be easier to compare and will support a more objective comparison of the strengths and weaknesses of the available technologies. There are KCMA requirements for which we cannot cater at the process level but which should be considered during the implementation of the reference process (e.g., dealing with large amounts of data).

The reference process was derived by adapting and extending the Ontology-Based Information Integration approach (Wache et al. 2001; Calvanese et al. 2001) thus closely connecting process steps for data integration and change management. The reference process is technology agnostic: for this, we replaced all Semantic Web specific terms with general terms, e.g., "ontologies" with "data models." The extension of OBII consisted in adding four more phases. These phases were derived from relevant related work and requirements and are shown as white boxes in the Fig. 7.4 while the original phases are shown as gray boxes. We utilize an IDEF-0[3] style diagram to structure the proposed approach, in which

[3]http://www.idef.com/idefo-function_modeling_method/.

processes are shown as boxes and resources are shown as directed arrows. The diagram clearly defines input (incoming arrows from the left-hand side of the box), output (outgoing arrows to the right-hand side of the box), consumable resources and stakeholders (input arrows from the bottom of the box) and standards (incoming arrows from the top of the box) used in the reference process.

There are three domain expert roles involved in the framework: Knowledge Engineer (KE), Project Manager (PM), and Domain Expert (DE). Input and output of the system is shown in the left- and right-side of the diagram, respectively. In the following, we explain the seven main phases of the KCMA reference process:

1. **Local Data Model Definition**. This phase requires the Knowledge Engineer and Domain Experts to translate the local tools data structure (e.g., MCAD model for mechanical engineer) to the local data model instance definition.
2. **Common Data Model and Mapping Definition**. KE and DE will define the common data model and its mappings to the local data models. To support this goal, vocabularies and standards are required to formalize the data model and mapping.
3. **Local Data Model Extraction, Transformation, and Load (ETL)**. With regards to the heterogeneous domain tools and their data formats within the MDEng environment, we need to provide the suitable extract, transform, and load (ETL) functions phase to produce the data in the required data model formats.
4. **Change detection**. This phase focuses on the detection of low-level (i.e., triples) and high-level (e.g., semantic and domain-specific) changes between two versions of engineering data. An important point to consider within this phase is to balance the expressiveness of high-level changes defined as input and the computational complexity of the detection algorithm, as mentioned in (Papavassiliou et al. 2009).
5. **Change validation**. The phase of change validation requires the definition of constraints for preserving the validity of data in the local (e.g., mechanical engineering) and global data models (e.g., power plant). Workflow definition is another important element, in order to configure involvement of validation components (e.g., constraint validation engine and domain experts) in the validation process.
6. **Change propagation**. Changes in the MDEng environment need to be propagated to the relevant components (i.e., common data model and other relevant local data models). This phase requires the common data model and mapping definitions, as well as validated changes. The knowledge engineer will configure the propagation based on the mapping definitions to make sure that no corrupted or irrelevant data is included in the propagation process.
7. **Data Store and Analysis**. The goal of this phase is to enable relevant stakeholders (e.g., project manager) to access and analyze the data and its changes within the projects. The changed data will be stored within a designated data store. Examples of queries that will be relevant to this data are: (1) Provenance

information of the changes (e.g., committer, date, reasons of change), (2) Change overview on specific objects, and (3) Analysis of completeness and inconsistencies over changes.

7.6 A Potential Semantic Web-Based Implementation of the KCMA Reference Process

We hereby provide an explanation of how the seven phases of the proposed reference process approach could be implemented with tools and techniques from the Semantic Web community. The framework draws on several standards and technologies, (e.g., SPARQL[4] for querying, PROV-O to represent provenance information) which can be used for structuring and implementing the approach.

1. **Local Data Model Definition**. Based on our experience in the domain, the W3C standard of RDF(S)[5] and OWL[6] languages are sufficient to define local data models. For a more detailed explanation of data model definition and some examples of engineering ontologies, we refer readers to Chap. 5.
2. **Common Data Model and Mapping Definition**. Similar to the case of local data models, we assumed that the RDF(S) and OWL languages are sufficient for common data model definitions. The mapping between common and local data models, however, is more complicated due to the unavailability of W3C standards for this task. There are several proposed approaches, e.g., SPIN[7] and EDOAL[8], as well as the generic SPARQL construct that could be utilized to define mappings between ontologies. However, they have to be investigated further to determine their suitability in the domain. Chapter 6 provides a detailed investigation of technologies for establishing mappings between ontologies.
3. **Local Data Model Extraction, Transformation, and Load (ETL)**. To extract data from heterogeneous data formats, ontology programming frameworks, e.g., Apache Jena[9] or OpenRDF Sesame[10] can be used. Furthermore, there are a number of applications for extracting data from various specific data formats, e.g., RDF123 (Han et al. 2008) and for spreadsheets and XMLTab Protégé plugin[11] and for XML-based documents. Chapter 5 contains information about data extraction from legacy data sources in engineering.

[4]http://www.w3.org/TR/sparql11-overview/.

[5]https://www.w3.org/TR/2004/REC-rdf-primer-20040210/; https://www.w3.org/TR/rdf-schema/.

[6]https://www.w3.org/TR/owl-ref/.

[7]https://www.w3.org/Submission/spin-overview/.

[8]http://alignapi.gforge.inria.fr/edoal.html.

[9]http://jena.apache.org/.

[10]http://rdf4j.org/.

[11]http://protegewiki.stanford.edu/wiki/XML_Tab.

4. **Change detection**. Generic programming frameworks, e.g., Apache Jena or OpenRDF Sesame, are typically able to detect low-level changes between two ontologies at the level of triples. Specialized algorithms are however required to detect high-level changes, either based on heuristics (Noy and Musen 2002), structural differences (Papavassiliou et al. 2009; Redmond and Noy 2011), or OWL reasoning (Gröner et al. 2010). Changes are represented either as triples (e.g., DBPedia change representation from Stadler et al. 2010) or specialized ontologies (e.g., change representation for OWL2 (Palma et al. 2009) and CHAO (Noy et al. 2006)).
5. **Change validation**. To formulate the constraints in a Semantic Web based solution, one possible option is to utilize the upcoming Shapes Constraint Language (SHACL[12]) by SHACL W3C working group, which aims to provide the standard constraint vocabulary for RDF graph data. SHACL strives to aggregate previous efforts of ontology constraint language definition and ontology quality assessment, e.g., RDFUnit (Kontokostas et al. 2014) and Shape Expression (Boneva et al. 2014). Chapter 13 contains examples of using SHACL.
6. **Change propagation**. The change propagation phase is closely related to the ontology mappings defined in the second phase. An example of possible Semantic Web-based solutions in this phase is to develop a change propagation solution based on SPIN mapping or SPARQL constructs.
7. **Data Store and Analysis**. From the Semantic Web perspective, one possible data storage solution is to utilize an RDF triplestore (e.g., OpenRDF Sesame). The PROV-O W3C standard can be used for representing change provenance information. Another important aspect in this phase is to provide different types of stakeholders with comfortable access for analyzing the data (Ekaputra et al. 2013). An example of end-user interface for ontology data analysis in the MDEng domain is presented in (Sabou et al. 2016).

To conclude, the current set of Semantic Web Technology tools and methodologies provides a relatively mature foundation for the implementation of the KCMA process. Semantic modeling, for instance, is required for *local data model definition* and *common data model and mapping definition phases*. This process can be achieved by relying on widely accepted W3C standards such as RDF(S) and OWL. The current set of ontology programming frameworks (e.g., Apache Jena and OpenRDF Sesame; required for *ETL* and *change detection phase*) could also be considered stable and well-maintained, while storage options for RDF graph data shows encouraging signs with more than 15 tools (commercial and open source) available at the moment[13] (required for *data store and analysis phase*).

However, there are critical issues to address for Semantic Web-based implementations of MDEng KCMA, namely the standardization of ontology constraint and mapping languages. Ontology constraint languages are required for the *change*

[12]https://w3c.github.io/data-shapes/shacl/.

[13]http://db-engines.com/en/ranking/rdf+store. Accessed on 16.02.2016.

validation phase. The upcoming SHACL standard is an important development in this direction and will be important for the ontology constraint aspect of the KCMA. Ontology mapping languages are required for the *common data model and mapping definition* and for the *change propagation* phases. Their development, however, is still yet to converge to W3C standards. EDOAL, SPIN, and SPARQL construct are currently the best options available. To overcome the lack of standards for mapping definitions, the adaptation of mapping methodologies from other research communities, (e.g., Triple Graph Grammars (Schürr and Klar 2008) from Model Driven Engineering) could be considered as additional options.

7.7 Summary and Future Work

In this book chapter, we have defined the context and challenges of KCMA in MDEng environments. To address the challenges, we have identified key requirements of KCMA in MDEng and provided an overview of techniques from three relevant research areas, namely Database Systems, Model-Based Engineering, and Semantic Web Technologies.

Our contribution in this book chapter is to generalize and extend the OBII approach, previously proposed for the purposes of data integration, to become a generic reference process of KCMA in MDEng environment. This generic and technology agnostic reference process is meant to lay the foundation towards a solution for providing a fully functional OBII-based KCMA solution for MDEng.

As our future work, we plan to provide a first concrete implementation of the reference process utilizing a selection of Semantic Web Technologies as discussed in Sect. 7.6. Additional concrete implementations using techniques from other research area (i.e., database and MBE) are also planned to enable comparison of the strengths and limitations of these different technologies for the KCMA tasks. We also plan to generalize the approach to address similar problem settings in other application domains, such as scholarly data management in empirical software engineering (Biffl et al. 2014).

Acknowledgments 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 in Austria.

References

Berardinelli, L., Biffl, S., Mätzler, E., Mayerhofer, T., Wimmer, M.: Model-based co-evolution of production systems and their libraries with automationML. In: Proceedings of the 20th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2015), pp. 1–8 (2015)

Biffl, S., Kalinowski, M., Ekaputra, F.J., Serral, E., Winkler, D.: Building empirical software engineering bodies of knowledge with systematic knowledge engineering. In: Proceedings of the 26th International Conference on Software Engineering and Knowledge Engineering (SEKE 2014), pp. 552–559 (2014)

Boneva, I., Gayo, J.E.L., Hym, S., Prud'hommeau, E.G., Solbrig, H., Staworko, S.: Validating RDF with shape expressions. Technical Report (2014). arXiv:1404.1270

Calvanese, D., De Giacomo, G., Lenzerini, M.: Ontology of integration and integration of ontologies. Description Logics **49**, 10–19 (2001)

Ekaputra, F.J., Serral, E., Winkler, D., Biffl, S.: An analysis framework for ontology querying tools. In: Proceedings of the 9th International Conference on Semantic Systems (iSEMANTICS 2013), pp. 1–8. ACM (2013)

Ekaputra, F.J., Serral, E., Sabou, M., Biffl, S.: Knowledge change management and analysis for multi-disciplinary engineering environments. In: Proceedings of the Posters and Demos Track of 11th International Conference on Semantic Systems (SEMANTiCS 2015) (2015)

Goring, M., Fay, A.: Modeling change and structural dependencies of automation systems. In: Proceedings of the 17th Conference on Emerging Technologies and Factory Automation (ETFA 2012), pp. 1–8. IEEE (2012)

Graube, M., Hensel, S., Urbas, L.: R43ples: Revisions for triples. In: Proceedings of the 1st Workshop on Linked Data Quality co-located with 10th International Conference on Semantic Systems (SEMANTiCS 2014) (2014)

Gröner, G., Parreiras, F.S., Staab, S.: Semantic recognition of ontology refactoring. In: Proceedings of the 9th International Semantic Web Conference (ISWC 2010), pp. 273–288. Springer (2010)

Han, L., Finin, T., Parr, C., Sachs, J., Joshi, A.: RDF123: From Spreadsheets to RDF. In: Proceedings of the 7th International Conference on the Semantic Web (ISWC 2008), pp. 451–466. Springer (2008)

Horridge, M., Redmond, T., Tudorache, T., Musen, M.A.: Binary OWL. In: Proceedings of the 10th International Workshop on OWL: Experiences and Directions (OWLED 2013), co-located with 10th Extended Semantic Web Conference (ESWC 2013) (2013)

Kehrer, T., Kelter, U., Taentzer, G.: Propagation of software model changes in the context of industrial plant automation. Automatisierungstechnik **62**(11), 803–814 (2014)

Klein, M.: Change management for distributed ontologies. Ph.D. thesis, Vrije Universiteit Amsterdam (2004)

Kontokostas, D., Westphal, P., Auer, S., Hellmann, S., Lehmann, J., Cornelissen, R., Zaveri, A.: Test-driven evaluation of linked data quality. In: Proceedings of the 23rd International Conference on World Wide Web (2014), pp. 747–758 (2014)

McBrien, P., Poulovassilis, A.: Schema evolution in heterogeneous database architectures, a schema transformation approach. In: Proceedings of the 14th International Conference of Advanced Information Systems Engineering (CAiSE 2002), pp. 484–499. Springer (2002)

Meyers, B., Vangheluwe, H.: A framework for evolution of modelling languages. Sci. Comput. Program. **76**(12), 1223–1246 (2011)

Mordinyi, R., Serral, E., Winkler, D., Biffl, S.: Evaluating software architectures using ontologies for storing and versioning of engineering data in heterogeneous systems engineering environments. In: Proceedings of the 15th Conference on Emerging Technologies and Factory Automation (ETFA 2012), pp. 1–10. IEEE (2014)

Noy, N.F., Klein, M.: Ontology evolution: not the same as schema evolution. Knowl. Inf. Syst. **6** (4), 428–440 (2004)

Noy, N.F., Musen, M.A.: Promptdiff: a fixed-point algorithm for comparing ontology versions. In: Proceedings of the 18th National Conference on Artificial Intelligence (AAAI/IAAI 2002), pp. 744–750 (2002)

Noy, N.F., Chugh, A., Liu, W., Musen, M.A.: A framework for ontology evolution in collaborative environments. In: Proceedings of the 5th International Conference on the Semantic Web (ISWC 2006), pp. 544–558. Springer (2006)

Palma, R., Haase, P., Corcho, O., Gómez-Pérez, A.: Change representation for OWL 2 ontologies. In: Proceedings of the 6th International Workshop on OWL: Experiences and Directions (OWLED 2009) (2009)

Papavassiliou, V., Flouris, G., Fundulaki, I., Kotzinos, D., Christophides, V.: On detecting high-level changes in RDF/S KBs. In: Proceedings of the 8th International Conference on the Semantic Web (ISWC 2009), pp. 473–488 (2009)

Redmond, T., Noy, N.: Computing the changes between ontologies. In: Proceedings of the Joint Workshop on Knowledge Evolution and Ontology Dynamics (EVODYN 2011), pp. 1–14 (2011)

Redmond, T., Smith, M., Drummond, N., Tudorache, T.: Managing change: an ontology version control system. In: Proceedings of the 5th International Workshop on OWL: Experiences and Directions (OWLED 2008) (2008)

Roddick, J.F.: A survey of schema versioning issues for database systems. Inf. Softw. Technol. **37** (7), 383–393 (1995)

Sabou, M., Ekaputra, F.J., Kovalenko, O., Biffl, S.: Supporting the engineering of cyber-physical production systems with the AutomationML analyzer. In: Proceedings of the Cyber-Physical Production Systems Workshop (CPPS 2016) (2016)

Schürr, A., Klar, F.: 15 years of triple graph grammars. In: Proceedings of the 4th International Conference on Graph Transformations (ICGT 2008), pp. 411–425. Springer, Berlin (2008)

Serral, E., Mordinyi, R., Kovalenko, O., Winkler, D., Biffl, S.: Evaluation of semantic data storages for integrating heterogenous disciplines in automation systems engineering. In: Proceedings of the 39th Annual Conference of Industrial Electronics Society Conference (IECON 2013), pp. 6858–6865. IEEE (2013)

Stadler, C., Martin, M., Lehmann, J., Hellmann, S.: Update strategies for DBpedia live. In: Proceedings of the Sixth Workshop on Scripting and Development for the Semantic Web (SFSW 2010) (2010)

Stojanovic, L.: Methods and tools for ontology evolution. Ph.D. thesis, Karlsruhe Institute of Technology (2004)

Stojanovic, L., Maedche, A., Motik, B., Stojanovic, N.: User-driven ontology evolution management. In: Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2002), pp. 285–300. Springer (2002)

Taentzer, G., Ermel, C., Langer, P., Wimmer, M.: A fundamental approach to model versioning based on graph modifications: from theory to implementation. Softw. Syst. Model. **13**(1), 239–272 (2014)

Vander Sande, M., Colpaert, P., Verborgh, R., Coppens, S., Mannens, E., Van de Walle, R.: R&Wbase: git for triples. In: Proceedings of the Linked Data on the Web Workshop (LDOW 2013) (2013)

Vogel-Heuser, B., Legat, C., Folmer, J., Rösch, S.: Challenges of parallel evolution in production automation focusing on requirements specification and fault handling. Automatisierungstechnik **62**(11), 758–770 (2014)

Wache, H., Voegele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., Hübner, S.: Ontology-based integration of information: a survey of existing approaches. In: IJCAI-01 Workshop: Ontologies and Information Sharing, pp. 108–117 (2001)

Waltersdorfer, F., Moser, T., Zoitl, A., Biffl, S.: Version management and conflict detection across heterogeneous engineering data models. In: Proceedings of the 8th International Conference on Industrial Informatics (INDIN 2010), IEEE, pp. 928–935 (2010)

Zablith, F.: Harvesting online ontologies for ontology evolution. Ph.D. thesis, The Open University, UK (2011)