NordDesign 2018
August 14 – 17, 2018
Linköping, Sweden

Consistent digitalization of engineering design – an ontology-based approach

Patricia Kügler[1], Benjamin Schleich[2], Sandro Wartzack[3]

[1]Engineering Design, Friedrich-Alexander-Universität Erlangen-Nürnberg
kuegler@mfk.fau.de
[2]Engineering Design, Friedrich-Alexander-Universität Erlangen-Nürnberg
schleich@mfk.fau.de
[3]Engineering Design, Friedrich-Alexander-Universität Erlangen-Nürnberg
wartzack@mfk.fau.de

Abstract (300-500 words)

Digitalization and Industry 4.0 are currently trend words in companies and research, which express the striving for digitalized manufacturing environments and autonomous manufacturing processes enabled by cyber-physical production systems. But where do we stand in product development? Modern virtual product development tools and processes allow the definition of product geometry and shape in fully three-dimensional computer-aided design environments. Additionally, even semantic information, for instance about the subsequent manufacturing, can be included employing feature-technology. Moreover, virtual testing of these three-dimensional product models can be performed using sophisticated computer-aided engineering and simulation tools leading to digitally supplied output data. However, CAD and simulation tools mainly support later design activities, while there is a lack of computer support in early design stages. This lack of digitalization in such early design stages eliminates the possibility of retracing the customer requirements in the product at late design activities. In consequence, the decisions made during the design process are hardly comprehensible and therefore existing product knowledge is difficult to reuse. This contribution shows the needs for a consistent digitalization of engineering design with focus on the early stages. This consistent digitalization refers to a representation and linkage of work results in design processes, which allows inference and reasoning between them. Thus, reusing of product knowledge becomes more comprehensible. Moreover, this contribution presents an ontology-based approach for this consistent digitalization, which particularly enables product developers to link early design stages to late design stages and to establish relations between customer requirements and the finished product. The approach thus supports the efficient reuse of relevant design knowledge employing an ontology. Besides this, the paper discusses the general advantages of ontologies for transparently storing and efficiently providing product knowledge. With the novel ontology-based approach proposed in this contribution, the reuse of finished product development processes becomes more efficient and transparent.

Keywords: Ontologies, Knowledge-based Engineering, Requirements Engineering, Information Integration

1 Introduction

"Would you tell me, please, which way I ought to go from here? ", asks Alice the Cheshire Cat. *"That depends a good deal on where you want to get to"*, answers the Cat. *"I don't much care where…"*, Alice responds. *"Then it doesn't matter which way you go."*, the Cat replies. "*...so long as I get somewhere."*, Alice adds as an explanation. *"Oh, you're sure to do that"*, says the Cat, *"if you only walk long enough"*. The small dialogue from LEWIS CARROLL'S novel *Alice's Adventures in Wonderland* illustrates what happens with projects, when it is not clear enough, what the overall target actually is. In product development, the target of a project is defined in the set of requirements the resulting product has to fulfil. Following the wisdom of the Cheshire Cat, if the requirements of a project are not defined clearly, one would probably get *somewhere*, but it will take a long time and probably would not meet the customer's requirements.
Requirement Engineering is an approach to support a clear definition and documentation of requirements and thus to prevent design projects from failure by insufficient requirements management. One main challenge with requirements is the lack of efficient digitalization, which hinders the evaluation whether and how they are finally fulfilled in a finished product. Thus, even if one knows where the project should get to, it is nearly impossible to assess if the target is reached, if the requirements are not related to the technical solutions, which fulfil them. To close this gap this contribution shows a novel approach for digitalizing requirements efficiently and consistently by using ontology-based representations. Furthermore, this approach is a first step to facilitate the connection and integration of the early stages of the product development process to the developed technical solutions.

2 State of the art

In the following, a systematic approach for the management of requirements, called Requirements Engineering (RE), is introduced. One main challenge is the lack of digitalization of this approach, thus the Enterprise Information Integration (EII) is presented as a solution of integrating and digitalizing heterogeneous data and information.

2.1 Requirements Engineering

RE is a systematic approach for establishing, documenting, analysing and managing the requirements while focusing on customer-oriented, technical and economic aspects (Ebert, 2012). Originally, RE was designed for software development, but the core tasks, i.e. establishing, documenting, analysing and managing requirements, are also applied within product development. The main aim of RE is to begin with the end in mind (Ebert, 2012). This end refers to the context and the vision of the project or product, which is formally defined within the requirements. Therefore, the requirements are the common basis for all project participants, like marketing, R&D, customers, etc. For customers they are kind of milestones in the project, while for the suppliers, planning and effort estimation is improved by a conscientious and consistent definition (Ebert, 2012). This also means the requirements list is the binding document for internal and external use and reduces misunderstandings about the product. Thus, requirements are documented in different ways. The typical documentation are specifications, which define the *what* and *why* of a development project (VDI 2519, 2001). Nowadays it is more common to manage requirements individually as database objects within data management systems and structured lists (Mayer-Bachmann, 2007). Several software tools, like DOORS and Soley, facilitate the management of requirements. Nevertheless, a main challenge is the system integration in the overall enterprise software landscape (Mayer-Bachmann, 2007). This means, there is a lack of connection to other software tools a company is working with. Even the libraries provide some references between the stored requirements

and also to documented functions and variants, the networking is insufficient (Mayer-Bachmann, 2007). For an efficient reuse, adaptation and change management it is vital to consistently integrate and relate all data, information and knowledge of a product development process consistently into a joint system environment.

2.2 Enterprise Information Integration

Enterprise Information Integration (EII) is an approach to provide uniform access to multiple and different data sources by integrating it in a uniform information representation (Halevy et al., 2005). We are talking about data, since the information sources initially contain a pure listing of facts. In the context of the concerning product, the data is structured to information. Within a product development process, different intern departments, such as R&D, marketing or manufacturing, as well as extern stakeholders are involved. This leads to the need to communicate and exchange and synchronize data and information from different sources. EII provides an integration without loading all data into a warehouse. A typical integration scenario begins with identifying the data sources, which should participate in the application, and building a virtual or mediated schema, which can be queried by users or applications (Halevy et al., 2005). Queries, that are posed over the virtual schema, are reformulated and processed through so-called wrappers and thus can query the local schemas of the different data sources. In this way, different data sources can be merged and queried through a uniform platform. One main challenge is the development of the mediated schema, which has to provide an intuitive access to the data and can handle the wrappers.
When dealing with complex heterogeneous data sources, semantic web technologies offer advantages. These technologies can integrate existing applications and data and deal also with an appropriate exchange of information and documents (Bernadi, Holz, Maus, & van Elst, 2006). One main aim is the integration of decentralized data and information in a way, that it is retrievable and understandable by all parties involved and that it is furthermore machine-processable. Significant for these aims are a uniform terminology and thus a common understanding of the data and information (Bernadi et al., 2006). Therefore, ontology-based approaches are especially appropriate to achieve these goals. While there exists no exact definition of an ontology, the best known comes from GRUBER (1993), who defines an ontology as an *explicit specification* of a *(shared) conceptualization*. This essentially refers to the concepts, relations and other distinctions that are relevant for the respective domain and *explicit* defined within the ontology. Furthermore, the *conceptualization* is in a form of representational vocabulary (classes, relations, instances), which is understood and accepted by all participants (Gruber, 2009). The class hierarchy contains the terminological knowledge, while the individuals (also called instances) are single expressions, which are generalized in the classes. Through the relations between classes or also between individuals, axioms are generated. In Figure 1, a class hierarchy is shown on the left side, which consists of three superclasses (*human, animal, community*) and corresponding subclasses (*woman, man, cat, mouse*).

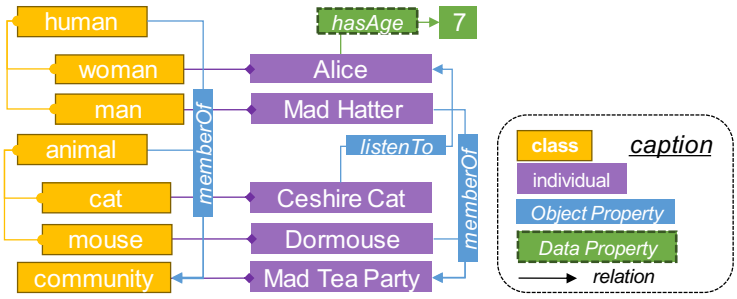

Figure 1. Classes, individuals and relations of an ontology with an example from LEWIS CARROLL'S novel "*Alice's Adventures in Wonderland*"

The subclasses inherit the properties of their superclass, thus in this case every woman is a human or every cat is an animal (Figure 1). On the right side of Figure 1 there are some individuals belonging to the classes (*Alice, Mad Hatter, Ceshire Cat*, etc.). This shows the generalizing character of an ontology. The individual "*Alice*" can be generalized as a woman, while the "*Cheshire Cat*" is generalized as a cat, while a cat is more generalized as an animal. Two classes or two individuals can be related to each other with so-called object properties (*memberOf, listenTo*). Another type of relations are data properties, which connect individuals and specific data values (*hasAge*). With these relations axioms like "*Mad Hatter memberOf Mad Tea Party*" or "*Alice hasAge 7*" are generated. These axioms enable specific queries, such as "*Who is member of the Mad Tea Party?*", which is answered with "*Mad Hatter*" and "*Dormouse*".

3 Ontology-based approach for a consistent digitalization of RE

As introduced in section 2, approaches for the management of requirements and also for the integration of heterogeneous data and information already exist. To close the described lack of digitalization within RE, the EII provides potentials, especially with ontology-based approaches. In the following a concept for such an approach is introduced for digitalizing RE-data from existing and new development projects. The ontology, shown in this contribution, is developed using the open-source ontology-editor Protégé (https://protege.stanford.edu/).

3.1 Concept and workflow

A consistent digitalization of product data refers to the linkage between all work results generated in the development process (e. g. requirements, functions, CAD data). Especially for reuse and adaptation this is vital to enable conclusions about decisions and developed solutions in closed projects. Moreover, efficient linked requirements facilitate conclusions about customer's needs and how they can be fulfilled within functional boundaries. This can help developing more sophisticated and successful products. Figure 2 shows the concept to integrate existing and new requirement data using an ontology as an mediated schema.

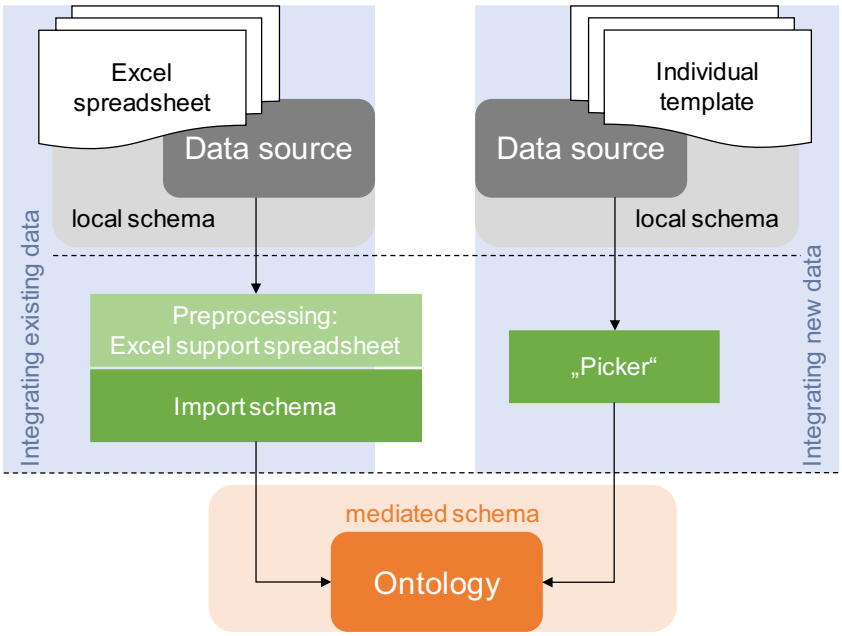

Figure 2. Concept of integrating existing and new requirement data with an ontology

Following the typical integration process, first the data source has to be identified. Usually requirements are stored in lists and thus held in Excel spreadsheet files. Even though the requirements are managed with applications like DOORS, it is common to export Excel spreadsheets for documentation purposes and information sharing. The Protégé-Plugin Cellfie supports translating axioms from Excel workbooks. To provide an easier and automatically implementation of individuals from Excel workbooks, it can be helpful to preprocess the original spreadsheet and work with a support spreadsheet (Section 3.2). The import schema defines which data is integrated in the ontology and how it is integrated (Section 3.2). Integrating existing data is always potentially problematic, because the form and structure of the held data is often not corresponding to the integration. Moreover, the data can be insufficient and thus the manual effort for data preparation for integration is high. Working with automated sorting methods and support spreadsheets reduces the manual effort. By integrating new data, the potential should be used to generate immediately "clean" data. Therefore, individual templates are developed to facilitate an optimal structuring for the integration in the ontology through a user interface or even by creating efficient structures in applications like DOORS. In this way, a time-consuming preprocessing of the data is redundant, only a "picker" is vital to ensure only the relevant data is integrated in an efficient way (Section 3.3). The ontology itself provides the mediated schema within the classes, individuals and relations, which enables querying and inference. The inference mechanisms are one of the main advantages of ontologies, because they provide discovering new relationships in existing data.

3.2 Integration of existing requirement data

As mentioned above, the typical data source of requirements are Excel files. Figure 3 (1) shows an excerpt of a requirements list from a test rig, which is considered closer in Section 4. In this requirements list there is information about the date, change index, responsibility and the requirement description itself. The formatted spreadsheet is preprocessed, such as the relevant data is transferred in a support spreadsheet through sorting methods.

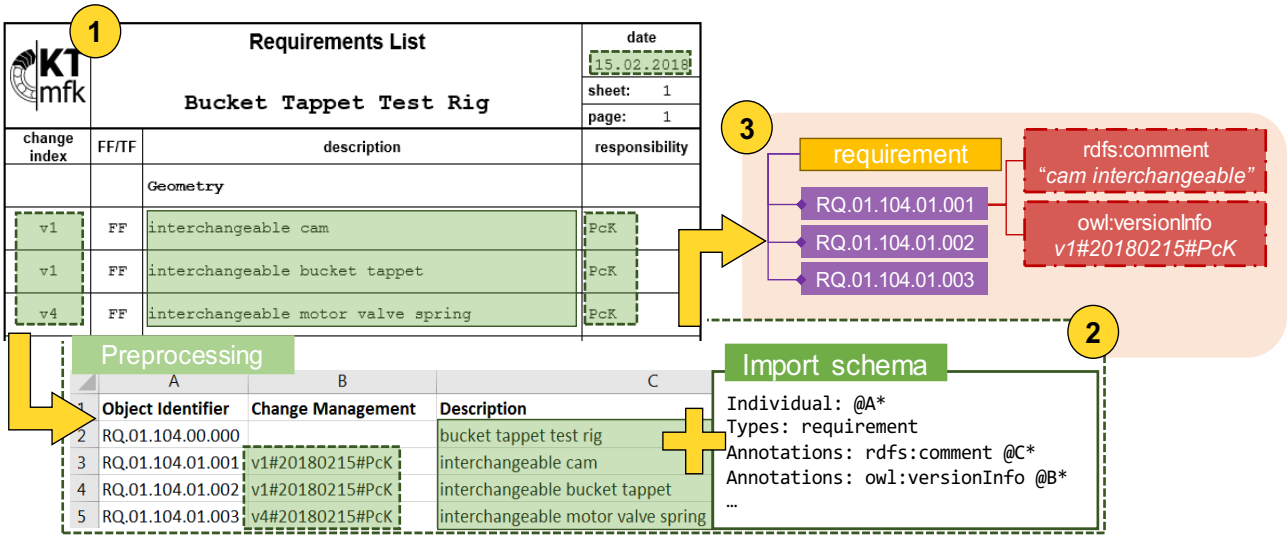

Figure 3. Integration of existing data with preprocessing and import schema

In Figure 3 (2) the description is listed in column C starting with the headline of the requirements list. For change management, for instance, a combination from change index, date and responsibility can be useful information for tracking when and who has changed some requirement. Within ontologies the uniqueness of names is vital, thus if there is no identifier for each requirement in the original requirements list, a unique number is defined in the support list. Starting with the ID number for the superordinate object (e.g. Figure 3 (1) "*Bucket Tappet*

Test Rig") and then increment the number with the filled rows in the file. In order to import the "clean" data from the support spreadsheet, an import schema is required. Therefore, the Plugin Cellfie provides the possibility to load the spreadsheet and then generate axioms from the schema written in Manchester Syntax (Horridge & Patel-Schneider, 2012). In Figure 3 (2) an example for generating individuals from the list data is shown. The first two lines define that every entry from column A is an individual (*Individual: @A**) of the class requirement (*Types: requirement*). In this case, the three individuals of the class requirement are generated (3). Special objects within ontologies are annotations, which can be defined with different data types. The type "*rdfs:comment*" provides the possibility of a human-readable description of an individual, for instance "cam interchangeable" (3), which is the requirements description of "*RQ.01.104.01.001*". Moreover, for change management, the "*owl:versionInfo*" provides a hook for versioning systems or can be exploited by list versioning annotations, like "*v1#20180215#PcK*" (3). Considering the import schema, it becomes clear why the preprocessing is vital. The preprocessing enables the use of the asterisk (*@A**) to express that the entire column A starting with a particular row is used as individuals. However, it is possible to define import schemas for single cells, but the manual effort increases enormously. Using preprocessing, which is customary in the context of data mining, and Cellfie, new individuals of existing requirements lists can be generated and enriched with additional information from the spreadsheets without high manual effort.

3.3 Integration of new requirement data

Supporting the integration of new data in the ontology, a template can help to manage the requirements data. Moreover, by structuring single requirements as objects in a unified form this provides a common basis and enables validation in RE (Ebert, 2012). There are existing methods to structure requirements lists, for instance the introduced schema shown in Figure 4 (Feldhusen & Grote, 2013). Following this schema, requirements lists provide four sectors. The content includes the description, categories and priorities of the requirements. Within the sector identification, IDs for the list itself and the single requirements as well as the version is included. Date, the product or project name and the organisation or department are part of the organisation sector. Tracking contains change management data, for instance the change date of a single requirement, the responsibility, thus who works on a special requirement and the source, for instance it is a customer's or legal requirements, etc.

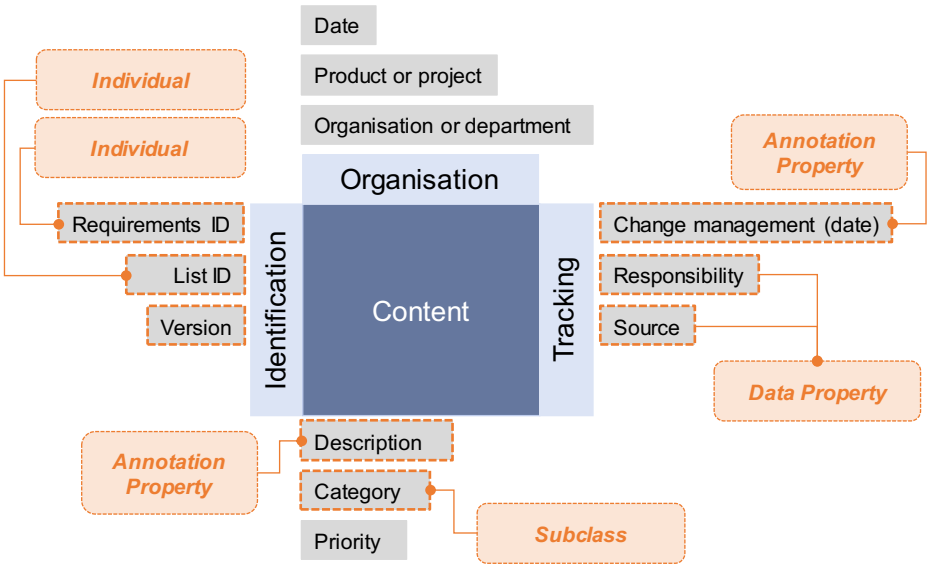

Figure 4. Content of the template and relevant data for integration (Feldhusen & Grote, 2013)

Through the "picker" only the relevant content of the template is imported in the ontology. The ID numbers of the single requirements and even the whole list is used as individuals (Figure 4), similar to the existing data integration (section 3.2). Equally, the description and the change management data is integrated with annotations, as described above. The categories of the requirements are integrated as subclasses of requirement and the single IDs can be instantiated as individuals of these subclasses (Figure 4). This provides a differentiated hierarchy of the requirements. The responsibility and source are generated as data properties. These are properties of a specific data type, like integer or string or even name. The main advantage of new data integrated through a uniform template is that no preprocessing of the data is needed, thus this is less error-prone and generates "clean" data directly. Moreover, in this way the requirements are managed as single objects, thus this enables analysing and processing.

4 Ontology-based requirements representation of a test rig

Previously, the general workflow for integrating existing and new data was shown. In the following, an example is presented to demonstrate the content and structure of an ontology for integrating requirements data. As mentioned in Section 3.2, a test rig for studying the *wear behaviour* and *lubrication conditions* of the cam/tappet contact in combustion engines, is used as a demonstrator. This test rig was designed highly based on reuse and adaptation from an earlier test rig, which was designed for studying *friction* behaviour. Therefore, the requirements changed. The test rig is documented in more detail in SCHULZ (2013). On the basis of the documentation, especially the requirements list, it was first decided which elements and information are relevant and how they can be structured within the ontology (Figure 5). The requirements are documented in a formatted Excel spreadsheet. The list contains information about the description, geometry, date, change index and responsibility of the single requirements. First, the general class hierarchy is implemented consisting of the classes *requirementsList* and *requirement* in Protégé. These two classes build with the object property (relation) *hasContent* the axiom "*requirementsList hasContent requirement*" (Figure 5). Based on this small structure the categories are automatically built as subclasses of the class requirement from a template or from spreadsheets.

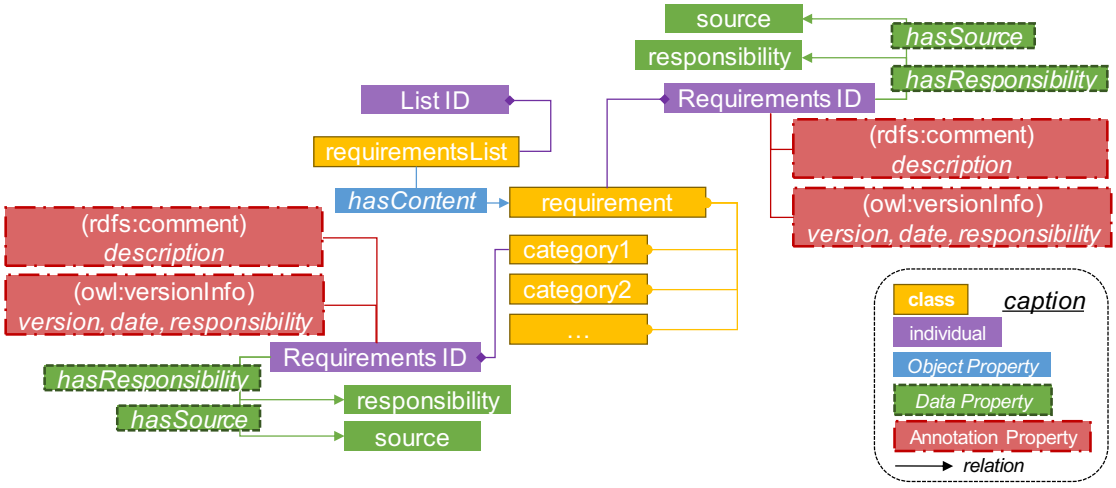

Figure 5. General concept of an ontology-based representation of requirements

Besides an object property, another relation is the data property, which connects individuals with data values. This property is exploited for defining the responsibility and the source of a requirement. In Figure 7 the acronym "*PcK*" marks the responsible person for a specific requirement. For sources standard definitions are defined, like "Customers RQ" or "*Functional RQ*" (Figure 7), to retrace why the requirement was included.

The automatic import of the list data with Cellfie in an ontological representation in Protégé is shown in Figure 6. The import schema of Cellfie has to be developed corresponding to the list data, which should be imported.

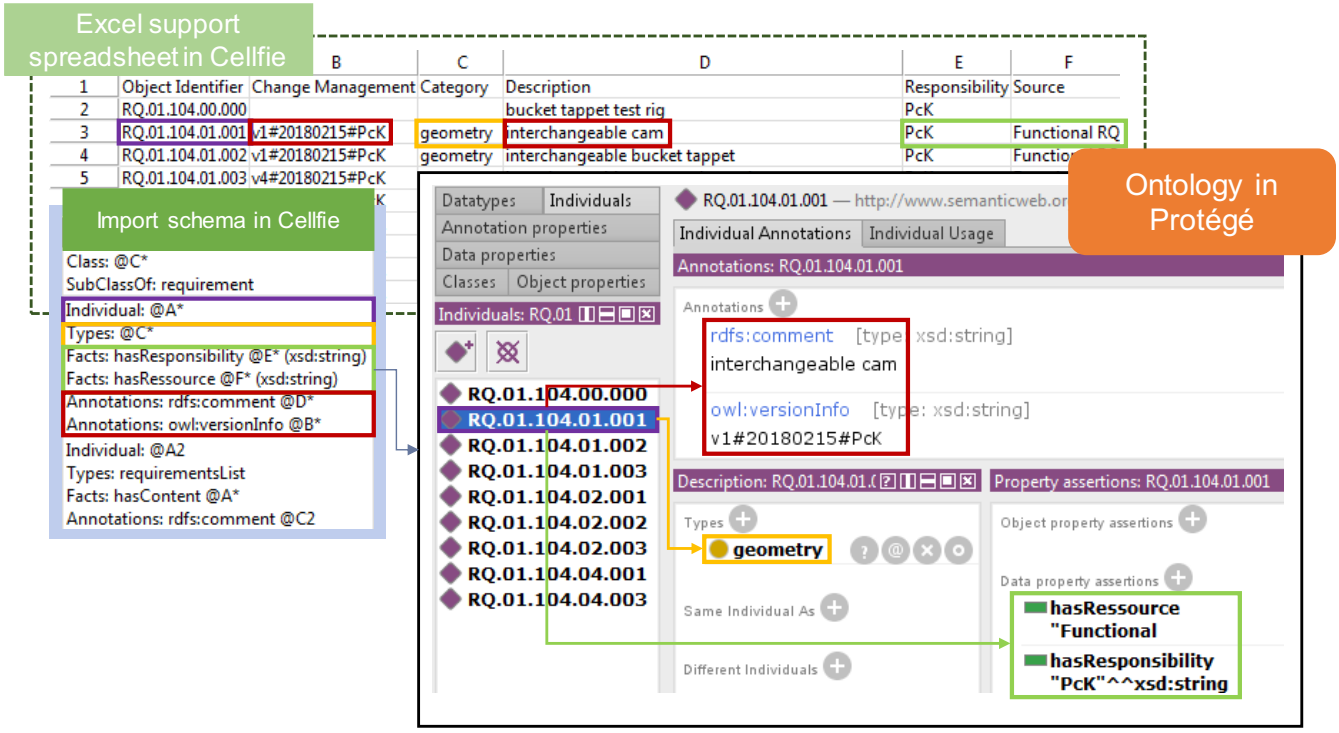

Figure 6. Integrating the test rig requirements data with Cellfie and Protégé

The first two lines of the import schema in Figure 6 import and generate subclasses (*geometry*, *kinematics*, *conditions* and *measuring*) of the class *requirement* from column C. Due to the class hierarchy, the subclasses inherit the properties of the class *requirement*. The import rule for column A generates individuals of these subclasses, for instance *geometry* (Figure 6). The individuals of the requirements list (*RQ.01.104.00.000*) and the requirements (*RQ.01.104.01.00x*) are equally related by *hasContent* like the classes through the rule "*Facts: hasContent @A**" (Figure 6 and Figure 7). Thus, the subclasses are also automatically connected to class *requirementsList* with regard to the relation *hasContent*. The columns E and F provide more information about the source and the responsibility, thus the axioms *"RQ.01.104.0y.00x hasRessource Functional RQ"* and *"RQ.01.104.0y.00x hasResponsibility PcK"* are generated.

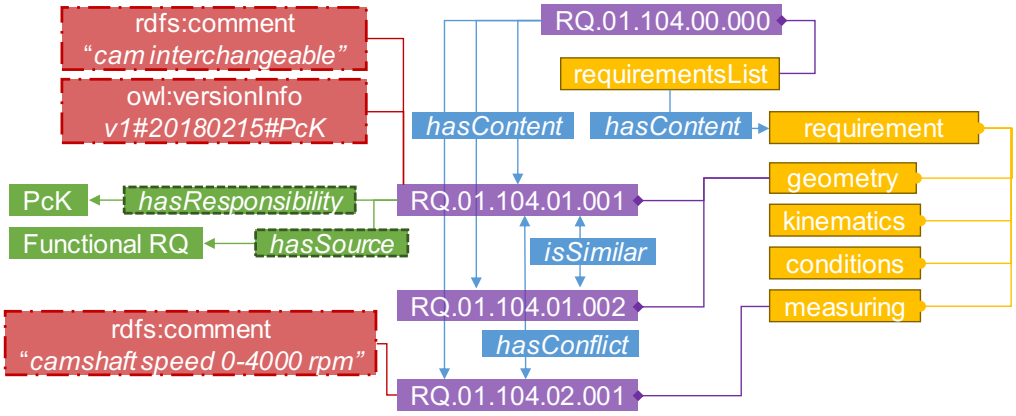

Figure 7. Excerpt from the ontology of the test rig

Due to the character of an ontology, interconnections between the single requirements are built as object properties. These relations can describe, for instance, if two requirements are similar (*isSimilar*) or stand in a conflict (*hasConflict*) to each other. In the example in Figure 7 the requirement "*cam interchangeable*" stands potentially in conflict with the requirement "*camshaft speed 0-4000rpm*", because on the one hand the cam should be fixed in a way that makes changing easy, on the other hand the mount of the cam has to enable the transmission of 4000 rpm speed. Such conflicts can be analysed and documented within the ontology, which enables a direct querying about conflicts. Thus, for example, all requirements, which stay in conflict with another requirement can be queried and listed. The interconnections can be built automatically for new projects through checkboxes in the template or for existing project documentations, if there are documented analyses about these relations (e. g. in text form), which can be preprocessed by Text Mining methods, like Information Extraction and then automatically integrated from list forms through Cellfie.

5 Discussion

Ontologies support the understanding and sharing of domain knowledge through defining a common terminology, which is human-readable and also machine-processable. The ontology-based integration of RE data leads to a consistent management of requirements by structuring and connecting relevant information about the requirements. Within the ontology-based data storing and management, querying and inference is possible, thus for instance, potentials for conflicts in the requirements can be presented. This leads to advantages in change management. Thus, if one requirement is related to another, they have to be changed together. The approach shown in this contribution enables the integration of existing and new data within the ontology. The integration of existing data is more challenging, because the data is usually not in the form, which is indispensably for the integration. Therefore, a preprocessing and import schema is introduced, which enables the import of list data from Excel spreadsheets. The consistent integration by means of an ontology as a mediated schema provides a basis to access data and information in a uniform manner. Therefore, the ontology-based representation, from which an excerpt is shown in this contribution, enables so-called *Intentional Forgetting* (IF). This approach supports an efficient reuse and adaptation by using mechanisms to only provide the elements of product models and processes, which are relevant and adapted to the development situation and is further introduced in KESTEL ET AL. (2017). A main challenge by developing the mediated schema for the implementation of IF is the generation of the relations within the ontological representation. On the one hand, this is due to the fact that the relations between the single work results are not documented and therefore cannot be extracted by Text and Data Mining methods. On the other hand, these relations are often based on intuitive decisions and experience of product development experts, thus they are only implicitly available.

6 Outlook and future work

For consistently storing and re-using product development knowledge, the ontology-based integration of requirements data is one first step. Moreover, ontologies enable the integration of heterogeneous data sources, thus it is vital to store also function structures or CAD representations ontology-based. This enables relations between the requirements and the technical solutions, thus the traceability of decisions in development processes is possible. Moreover, an ontology-based representation supports analysing and processing, like the mentioned Intentional Forgetting (Section 5). For this purpose, it is part of current research to implement the automatic ontology-based integration and synchronization of the various sources of product documentation.

The consistent integration and relation of requirements and the relation to technical solution provide the possibility to retrace how similar products and solutions were developed and thus learn from closed projects. Therefore, following the wisdom of the Cheshire Cat from the introduction, the way to go depends on where one wants to go. Conversely, this means knowing and understanding the target reveals the way to reach it.

Acknowledgement

The authors gratefully acknowledge the financial support of project WA 2913/22-1 within the Priority Program 1921, by the German Research Foundation (DFG).

Citations and References

Bernadi, A., Holz, H., Maus, H., & van Elst, L. (2006). Komplexe Arbeitswelten in der Wissensgesellschaft. In T. Pellegrini & A. Blumauer (Eds.), *X.media.press. Semantic Web: Wege zur vernetzten Wissensgesellschaft* (1st ed., pp. 27–45). Berlin: Springer.

Ebert, C. (2012). *Systematisches Requirements Engineering: Anforderungen ermitteln, spezifizieren, analysieren und verwalten* (4th ed.). Heidelberg: Dpunkt.verlag.

Feldhusen, J., & Grote, K.-H. (Eds.). (2013). *Pahl/Beitz Konstruktionslehre*. Berlin, Heidelberg: Springer.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, *5*(2), 199–220. https://doi.org/10.1006/knac.1993.1008

Gruber, T. R. (2009). Ontology. In L. Liu & M. T. Özsu (Eds.), *Encyclopedia of database systems* (pp. 1963–1965). Boston, MA: Springer.

Halevy, A. Y., Ashish, N., Bitton, D., Carey, M., Draper, D., Pollock, J., Rosenthal, A., & Sikka, V. (2005). Enterprise information integration. In F. Ozcan (Ed.), *Proceedings of the 2005 ACM SIGMOD international conference on Management of data - SIGMOD '05* (pp. 778–787). New York, USA: ACM Press. https://doi.org/10.1145/1066157.1066246

Horridge, M., & Patel-Schneider, P. F. (2012). OWL 2 Web Ontology Language: Manchester Syntax (Second Edition). W3C Working Group Note. Retrieved from https://www.w3.org/TR/2012/NOTE-owl2-manchester-syntax-20121211/

Kestel, P., Luft, T., Schon, C., Kügler, P., Bayer Thomas, Schleich, B., & Wartzack, S. (2017). Konzept zur zielgerichteten, ontologiebasierten Wiederverwendung von Produktmodellen. In D. Krause, K. Paetzold, & S. Wartzack (Eds.), *Beiträge zum 28. DfX-Symposium.* Hamburg: TuTech.

Mayer-Bachmann, R. (2007). Integratives Anforderungsmanagement: Konzept und Anforderungsmodell am Beispiel der Fahrzeugentwicklung (Dissertation). Universität Karlsruhe (TH), Karlsruhe.

Schulz, E. (2013). Wissensbasierte Vorhersage der Reibung im komplexen tribologischen Systemen am Beispiel des Kontakts Nockenwelle/beschichteter Tassensstößel (PhD). Friedrich-Alexander-Universität, Erlangen-Nuremberg, Germany.

VDI 2519 (2001). *Blatt 1, Vorgehensweise bei der Erstellung von Lasten-/Pflichtenheften.* Berlin: Beuth.