CASCADING FORGETTING IN PRODUCT DEVELOPMENT CHALLENGES AND EVALUATION

Kügler, Patricia (1); Schon, Claudia (2); Schleich, Benjamin (1); Staab, Steffen (2); Wartzack, Sandro (1)

1: Friedrich-Alexander-Universität Erlangen-Nürnberg; 2: University of Koblenz-Landau

ABSTRACT

Vast amounts of information and knowledge is produced and stored within product design projects. Especially for reuse and adaptation there exists no suitable method for product designers to handle this information overload. Due to this, the selection of relevant information in a specific development situation is time-consuming and inefficient. To tackle this issue, the novel approach Intentional Forgetting (IF) is applied for product design, which aims to support reuse and adaptation by reducing the vast amount of information to the relevant. Within this contribution an IF-operator called Cascading Forgetting is introduced and evaluated, which was implemented for forgetting related information elements in ontology knowledge bases. For the evaluation the development process of a test-rig for studying friction and wear behaviour of the cam/tappet contact in combustion engines is analysed. Due to the interdisciplinary task of the evaluation and the characteristics of semantic model, challenges are discussed. In conclusion, the focus of the evaluation is to consider how reliable the Cascading Forgetting works and how intuitive ontology-based representations appear to engineers.

Keywords: Intentional Forgetting, Ontologies, Knowledge management, Semantic data processing

Contact:
Kügler, Patricia
Friedrich-Alexander-Universität Erlangen-Nürnberg
Engineering Design
Germany
kuegler@mfk.fau.de

Cite this article: Kügler, P., Schon, C., Schleich, B., Staab, S., Wartzack, S. (2019) 'Cascading Forgetting in Product Development Challenges and Evaluation', in *Proceedings of the 22nd International Conference on Engineering Design (ICED19),* Delft, The Netherlands, 5-8 August 2019. DOI:10.1017/dsi.2019.259

1 INTRODUCTION

Towards efficiency, productivity and innovation the increasing volume of available data and information in organisations becomes more and more a challenge. Especially in product design a lot of data and information about projects is documented, thus the resulting product knowledge could be reused. This can lead to saving costs and time in development projects on the one hand. On the other hand, individuals become overwhelmed by this vast amount of information. This information overload could reduce performance and affects decision making (Jackson and Farzaneh, 2012). Even if decision making must involve an information basis, however empirical studies proofed that too many information could reduce the quality of decisions because of simplification (Probst *et al.*, 2002). This phenomenon occurs, because we are not able to objectively analyse and evaluate vast amounts of information, thus we have to simplify them (Probst *et al.*, 2002). One strategy of simplification in that case is according to Shenk (2007) the persistence of status quo, which can inhibit fresh ideas and innovation. To prevent that phenomenon, a novel method called Intentional Forgetting (IF) has been applied to product knowledge (Kestel *et al.*, 2017). The main purpose of IF is to reduce the presented amount of information during the product development process to relevant elements. Thus, the product designer is supported by a more efficient and target-oriented reuse of information and knowledge.
This contribution aims at presenting the evaluation of a specific forgetting operator for ontological knowledge bases, namely Cascading Forgetting. This operator is developed with the aim to determine related elements in a knowledge base, which should be rejected together. As the method of IF in the context of product knowledge is an interdisciplinary work between IT and engineers, there occur some challenges. Therefore, besides the evaluation of Cascading Forgetting, the focus of this contribution is also the overcoming of challenges caused by the highly interdisciplinary character of the topic and the evaluation.

2 BACKGROUND ON KNOWLEDGE REPRESENTATION AND UPDATING

A general theory of knowledge reuse is given by Markus (2001), who presents the importance of knowledge and its reuse in organisations and industry in general. There are at least four types of knowledge reusers: shared work producers, shared work practitioners, expertise-seeking novices and secondary knowledge miners (Markus, 2001). For product design especially the first two types are important. With "shared work producers" a group of people is meant, who works together in a team, which produces knowledge for their own later reuse (Markus, 2001). Thus, this reuse situation arises, for instance, by the development of variants or a new generation of existing products. Otherwise, "shared work practitioners" are people, who are doing similar work, but in different settings, thus they produce knowledge for each other's use (Markus, 2001). This is a typical team work situation, where information and knowledge have to be shared and understood by all participants. Thus, for instance, the information about product requirements have to be clear and should be shared among the whole design team. Both reuse situations have some challenge in common: a medium to capture, analyse and share the knowledge is needed and a common understanding of all participants has to be reached. Furthermore, for long-term use, this medium has to tolerance dynamics and changes within the knowledge base. In the following some more information about knowledge representation and reuse, as well as dynamics in knowledge bases, is presented.

2.1 Knowledge representation and reuse

For capturing and analysing information and knowledge, several knowledge representation forms are well-known. Rude (1998) introduced, inter alia, rule- and frame-based representation methods, as well as semantic networks. While rule-based systems provide a more or less unstructured database with rules as knowledge, the frame-based approach works with structured knowledge objects (Rude, 1998). Semantic networks are a more logically formalism for representation, as they are graphs, which are built with nodes and labelled edges (Seel, 2012). However, frames and semantic networks have a strong common basis, because they aim both the representation of individuals and their relationships (Baader, 2010). While network-based systems are human-centred and an intuitive way for knowledge representation and especially visualization, they are not satisfactory, because they don't provide a precise semantic characterization (Baader, 2010).

https://doi.org/10.1017/dsi.2019.259 Published online by Cambridge University Press

This leads to the result, that even virtually identical-looking systems behave differently. However, semantic networks and frames are the basis for Description Logic (DL), which is a formal language for representing knowledge and reason about it (Baader, 2010).
A knowledge representation based on DL is usually divided in a TBox and an ABox. The TBox consists of the underlying terminology, which is needed to describe the presented domain. It contains classes and relations between them in a specialized vocabulary describing the ontology. The ABox instead contains assertions about individuals, which belong to the generalized classes of the ontology. The term "ontology" is borrowed from philosophy and stands for an formal and explicit specification of a shared conceptualization (Gruber, 1993). In practice, ontologies can be formulated in the Web Ontology Language (OWL), which corresponds to DL (Baader, 2010). Ontologies are well-suited as a formal knowledge medium for reuse. On the one hand, an ontology provides advantages by analysing the underlying information, thus it is able to reason and infer. Otherwise, an ontology provides a consistent and unique vocabulary, which can be shared and understood by all participants, thus it prevents misunderstanding. Semantic approaches became popular during the last years, especially for reusing knowledge. Due to these advantages, the use of ontologies for collaborative work and knowledge reuse becomes more and more popular either. A general approach for reusing engineering design knowledge was developed by Baxter *et al.* (2007). This methodology integrates best practice reuse, capturing of design rationale and knowledge-based support and supports process, product and task knowledge with one system (Baxter *et al.*, 2007). The focus of this project is the storing and monitoring of information and knowledge, thus the mentioned information overload can inhibit an efficient reuse. Moreover, Li *et al.* (2018) introduced an approach for mapping design and manufacturing knowledge with an ontology. This approach aims to support the designer by providing manufacturing constraints already in the early design stages, thus the manufacturability can be verified. The focus of this approach is on closing the gap between design and manufacturing. With focus on manufacturing another semantic approach for knowledge reuse is given by Camarillo *et al.* (2018), which supports process Failure Mode and Effects Analysis (FMEA) by an ontology. Within additive manufacturing Hagedorn *et al.* (2018) present a method for innovative design. The approach works with an ontology, which captures business and technical knowledge about the innovative use of additive manufacturing. As already mentioned, the approaches focus more on the capturing and representing aspect, not an efficient retrieval. Moreover, dynamics and evolutions in the built knowledge bases are not considered.

2.2 Changing and evolving ontological knowledge bases

Once a knowledge base is built up, the information and knowledge is usually not static. Thus, when it comes to changes and dynamics, the content has to be revised. Within the classical belief revision theory according to Alchourrón *et al.* (1985), which is also called AGM model, there are three change operations: expansion, revision and contraction The simplest form of change is the expansion, which adds a new axiom to a given knowledge base (hopefully) without violating the consistency. Contraction stands for the rejection of axioms, which were further in the knowledge base. Contraction goes hand in hand with the challenge of determining the axioms, which have to be rejected in order to make sure that the contracted axiom is not entailed anymore (Alchourrón *et al.*, 1985). Revision is an operation where axioms are added, which are inconsistent with the given knowledge base, thus it could belong with some contraction operation. Finally, inconsistencies caused by changes in ontologies are a main challenge for evolving and dynamic knowledge bases. How these inconsistencies arise by updates in knowledge bases, can be seen in the following example of a revision operation from Gärdenfors (1992). Suppose there is a knowledge base, which contains the axioms from Table 1.

Table 1. Axioms of an example knowledge base (Gärdenfors, 1992)

Axiom	**Description**
α	All European swans are white.
β	The bird caught in the trap is a swan.
γ	The bird caught in the trap comes from Sweden.
δ	Sweden is part of Europe.
Inferred axiom	
ε	The bird caught in the trap is white.

https://doi.org/10.1017/dsi.2019.259 Published online by Cambridge University Press

Furthermore, the last axiom ε is derived from the facts α-δ. If the bird in the trap turns out to be black, the existing knowledge base has to be updated and becomes inconsistent, because now not every European swan is white, as axiom α is violated. Consequently, a revision has to be applied. The question is, which of the axioms have to be rejected. This is not trivial, because every statement in the knowledge base has consequences and effects caused by inference. Thus, the decision is, which of these consequences to retain and which to retract (Gärdenfors, 1992). For instance, there is the decision between the two revisions of axiom α (see Table 2).

Table 2. Possible revised axioms (Gärdenfors, 1992)

Axiom	Description
α′	All European swans except the one caught in the trap are white.
α″	All European swans except some of the Swedish are white

Dealing with such updates can be done in different ways, thus different semantics are applied to prevent or debug the inconsistencies in updated knowledge bases. Schon and Staab (2017) present an approach for using query-driven updates with SPARQL update on instance-level. Thus, they formulate semantics for update operations within the ABox. SPARQL query-language (Harris *et al.*, 2013) was developed for accessing the information in OWL ontologies. Due to changes represented knowledge is usually not static, thus the SPARQL update language (Gearon *et al.*, 2013) was developed to meet the requirements for dynamics in knowledge bases. As it could be seen in the shown example, dynamics in knowledge bases is not just the single deletion of one database entry. Therefore, the challenge for applying IF to engineering design knowledge bases is the development of operators to ensure the forgetting of the desired elements.

3 CASCADING FORGETTING IN PRODUCT DEVELOPMENT

As mentioned in the previous section, there are different ways for dealing with updates in ontological knowledge bases. In product development every piece of information is interdependent, thus for instance, geometry and design is interdependent from requirements and function. For forgetting operations, it is important to be consistent with regard to the dependencies, thus if one piece of information is not needed anymore, related information is rejected as well. Therefore, the IF-operator Cascading Forgetting is developed and implemented. For evaluation purposes, an example ontology is introduced, which contains the semantic model of design information about a test-rig.

3.1 An example: Setting up an ontology as a semantic model of a test-rig

For evaluating the Cascading Forgetting, an example ontology was built as a semantic model of a test-rig, which aims studying the wear behaviour and lubrication conditions of the cam/tappet contact in combustion engines. This demonstrator was chosen, because its development is highly based on reuse and adaptation of an earlier test-rig development, which is for studying friction behaviour. For more information about the test-rigs and the studies further read the contribution of Marian *et al.* (2018). Both development processes are well documented, thus design adaptations are traceable. For instance, measuring friction forces was realized with piezoelectric sensors within the first test-rig, which are expensive and not robust enough for long-term studies. Due to this, the second test-rig was modified and the friction force is no longer detected by piezoelectric sensors. Those adaptions provide some unintentionally used IF mechanisms, like Cascading Forgetting, thus it was used to inspire the scenarios for IF. Figure 1 shows an excerpt of the graphical visualized semantic model of the test-rig, which was firstly introduced in Kügler *et al.* (2018). In the left down corner the core structure of the ontology describes the product development process with classes (yellow) and relations (blue). The ontology for the test-rig is partly automatically generated. While requirements, functions and solution principles have to be inserted manually, in Kügler *et al.* (2018) an automated approach is introduced for generating individuals from CAD-data and assign them to classes of the core ontology, as well as relate them. This approach uses text-mining methods like Information Extraction for extracting classes, individuals and relations from CAD reference lists. In Figure 1 some individuals (purple), which are assigned to the ontology, and their relations (arrows) to each other are shown. The semantic of the relations is equal to the class system and is not explicit shown in the figure because of transparency and comprehensibility.

https://doi.org/10.1017/dsi.2019.259 Published online by Cambridge University Press

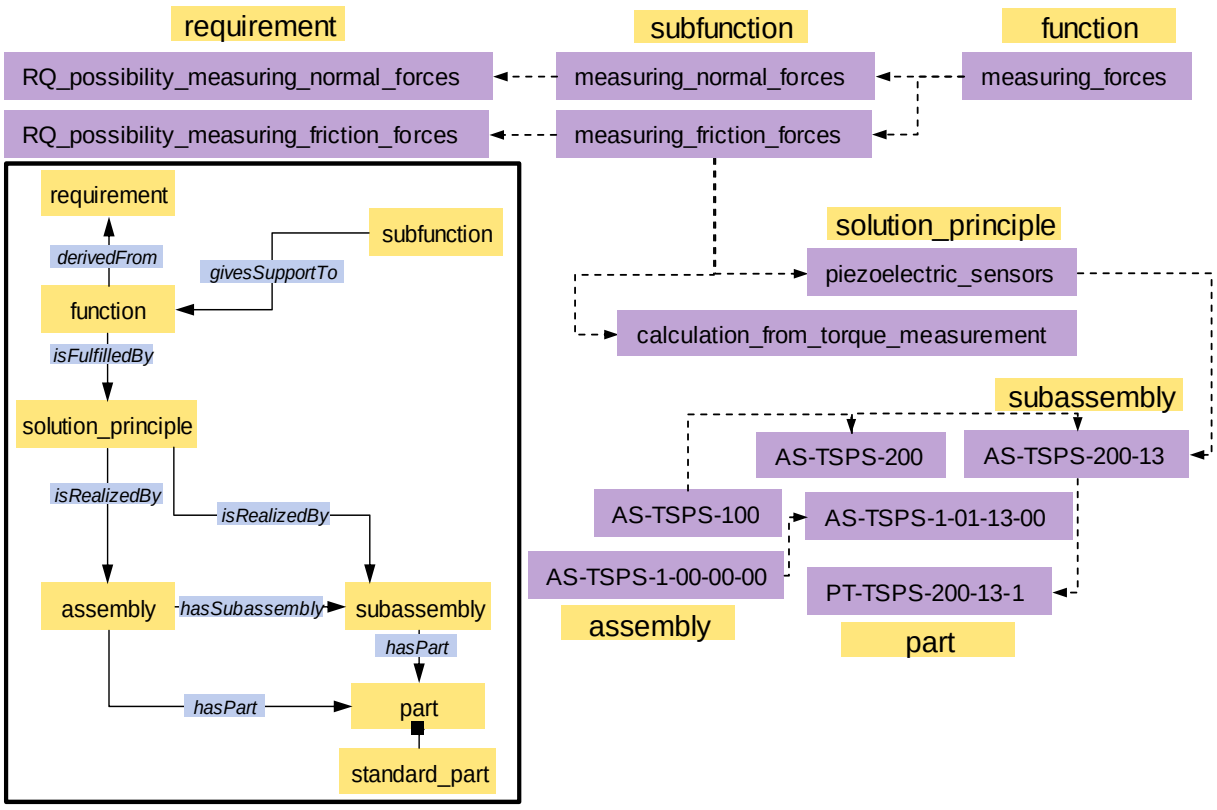

Figure 1. Graphical section from semantic model of the test-rig

As it can be seen the individuals are highly interdependent. For instance, the **function** "measuring_friction_forces" and the **requirement** "RQ_possibility_measuring_friction_forces" are connected due to the relation "*derivedFrom*". Beyond the advantages of the test-rig as a demonstrator, like traceability of adaptation, it causes some challenges through the interdisciplinary character of the project. While for the engineers the adaptations of some components and functions caused by new requirements are clear and understandable, for the project partners of the IT side this was hardly transparent. Vice versa, the engineers hardly understood the effects of semantics for updating, which are used for the IF operators. Due to this, the communication between the partners required a common basis, which provides the possibility to deliver the forgetting scenarios from the test-rig development and the other way round to discuss the implementation of semantics for IF. Therefore, the ontology provides this basis, because the engineers can describe the forgetting scenarios using the formal logic of the ontology in the form shown in Table 3. It is not important for the IT-experts to understand the technical interdependencies, as long they understand the underlying structure of classes and relations from the ontology. Vice versa, which effects metaproperties cause can be described in the same way.

3.2 Using metaproperties and SPARQL queries for Cascading Forgetting

The Cascading Forgetting operation is a meta-property-guided deletion, which exploits the metaproperties rigidity and dependency to guide the deletion and the desired cascading behaviour (Schon *et al.*, 2018). Metaproperties provide information about classes and their relationship to one another. The metaproperties are added manually as annotations to the classes in the TBox of the test-rig model by the engineers, using the ontology as communication medium for the implementation of the Cascading Forgetting by the IT-experts. The first metaproperty is rigidity. This metaproperty can be set for classes, which are essential to all its individuals. For instance, usually a class person is rigid, because one cannot stop being a person. However, rigidity can be exploited for Cascading Forgetting, because rejecting the fact that an individual belongs to a rigid class removes such a fundamental property of this individual that it should be entirely removed from the ABox. As seen in Figure 2 the classes **requirement** and **function** are set rigid. Suppose the ABox contains the following axioms (see Table 3). If the **function** "measuring_forces" (1) is rejected, also axiom (3) is deleted, because it contains the individual of the rigid class. Furthermore, the deletion of axiom (2) is caused by the dependency-metaproperty **subfunction** dependsOn **function** and no individual of **function** is left. (see Figure 2).

https://doi.org/10.1017/dsi.2019.259 Published online by Cambridge University Press

Table 3. Example ABox of the test-rig

(1) **function** (measuring_forces)
(2) **subfunction** (measuring_friction_forces)
(3) *givesSupportTo* (measuring_friction_forces, measuring_forces)

This shows how the dependency supports the cascading behaviour of the operation. Due to the fact, that some class depends on another class, the individuals of that class will be rejected with the individuals from the class it depends on, if there is no individual left belonging to the class. The Cascading Forgetting operator combines the dependency- and rigidity-guided deletions, allowing interactions between rigid concepts and dependencies, which can lead to further forgetting (Schon *et al.*, 2018). A dependency-guided deletion can lead to the deletion of a rigid assertion, which will cause the deletion of all related axioms, leading to the violation of dependencies (Schon *et al.*, 2018).

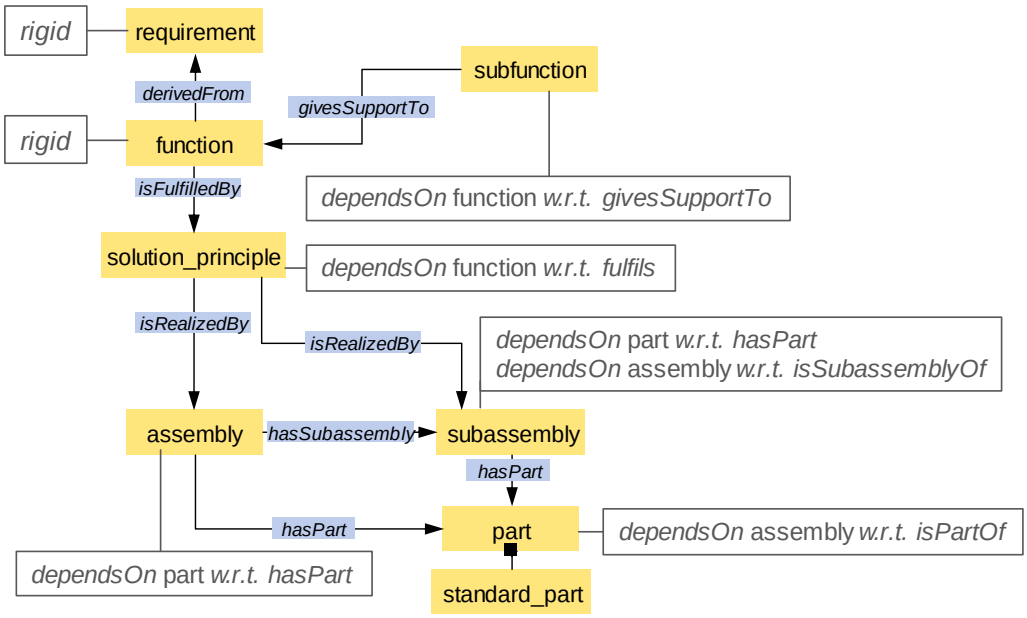

Figure 2. Metaproperties rigidity and dependency class annotation

For the formal definition of dependency- and rigidity-guided deletion further read Schon *et al.* (2018). The use of metaproperty-guided deletion seems to be well-suited for Cascading Forgetting of engineering design knowledge, because it specifically rejects the highly interdependent elements. The evaluation of the implemented operators will be part of the next section.

4 INTERDISCIPLINARY PERFORMANCE EVALUATION OF CASCADING FORGETTING

The evaluation of the Cascading Forgetting operator was conducted by the IT partners of the project, who also developed the operator. The knowledge base of the test-rig was developed by the engineers within the project, who supported the evaluation with additional material for the test persons. In the following the focus is on the conduction with the engineers.

4.1 Evaluation set up

The project-partners from the IT developed a questionnaire with 40 questions in the form shown in Figure 3. Seven experts from the engineering side were chosen for the evaluation. Four of them were not familiar with the development of the test-rigs, neither with the semantic model. One was an expert of the test-rigs and one has developed the ontology and is author of this paper. The scheme is the same in every question, thus the engineers had to decide if they want to delete the second assertion with the first one.

https://doi.org/10.1017/dsi.2019.259 Published online by Cambridge University Press

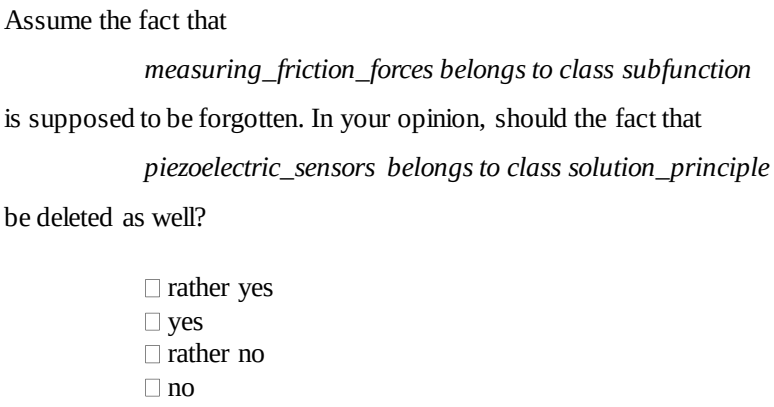

Figure 3. Example question of the questionnaire developed by Institute WeST

This means for the example in Figure 3 the engineers had to decide if they want to delete the **solution principle** piezoelectric_sensors with the **subfunction** measuring_friction_forces or not. Due to that decision the engineers had to know about the relations and hierarchical structure of the test-rig. The main objective of the evaluation was finding out, how well the Cascading Forgetting operator matches the desired and expected forgetting operations of the engineers. Different knowledge states of the test persons are important for the engineering perspective of evaluation. The evaluation was used to improve ontology modelling and to learn how the forgetting is perceived by the test persons. Also a question of interest was, if there are significant differences between the answers persons related to different knowledge states. For preparation and conduction of the evaluation there arise two main challenges:

1. Understanding the semantics

The challenge with the final questionnaire was, that the used semantics, and thus ontologies in general, are not a very common knowledge representation form in engineering design yet. Thus, the main question from engineering perspective at this point was, how well the test persons deal with these semantics. Even if the vocabulary of the ontology was developed in cooperation with an expert for the test-rig, the notation and structure of an ontology is sometimes difficult to understand. Therefore, the following question should be answered by the evaluation from engineering perspective: Do one need a deeper background knowledge by dealing with ontology representations or is it enough to know the structure of the ontology?

2. Presenting the relations within the test-rigs

Some additional material was provided for the test persons, thus they know the relevant hierarchical structure of the test-rigs and the interdependencies within the whole ontology. A main challenge for providing this additional material was, that the information has to be clear enough to understand the relationships within the semantic model of the test rig, but not influence the test persons by decisions of forgetting. Therefore, an relevant excerpt from the basic ontology structure (see Figure 1, below left) and an excerpt of a model tree representation (like in CAD software) was presented. Moreover, a table sheet was added, which shows the relationships between requirements, functions and solution principles.

4.2 Results and Discussion

For evaluating the quality of the Cascading Forgetting operator the metrics precision and recall were created. Precision is a metric for showing how many matches between the answers of the engineers and the Cascading Forgetting are achieved. Recall stands for how many of the desired forgetting operations, the Cascading Forgetting operator has performed. The precision of the Cascading Forgetting was 1, which indicates, that the engineers agreed with all deletions the operator has performed and there is no case the operator deletes to much (Schon *et al.*, 2018). The recall was comparatively low with 0.48, which means the engineers expected more forgetting operations, which are not performed by the operator (Schon *et al.*, 2018). For more detail on how the evaluation was conducted further read Schon *et al.* (2018). An interesting difference within the interpretation of these metrics arises during the discussion of the results between the interdisciplinary project partners: For the IT-experts the low recall was interpreted more negative for the performance of the operator, as it was for the engineers.

https://doi.org/10.1017/dsi.2019.259 Published online by Cambridge University Press

A main reason for that is, that "forgetting" is associated with loss of information, which is a highly sensitive topic in product design. Therefore, a so-to-say cautiously behaving forgetting operator is perceived more positive from an engineering perspective. However, for a more efficient use of Cascading Forgetting the recall should be a little bit higher. As described in Schon *et al.* (2018), by analysing the questions, where the majority of the engineers wanted to delete more than the Cascading Forgetting operator, some causes have been revealed. Some metaproperties were not set in the ontology. This is a problem of the currently manual set of the metaproperties, which will supported by spreadsheets in the future (see section 4.3). Moreover, some modelling errors occurred during the evaluation. For instance, the class subfunction was not set as a subclass of function. Schon *et al.* (2018) also mention that some kind of counting dependencies are needed, thus assertions like something is only a subassembly if it has at least two parts, can be mapped. Another aspect are mutual dependencies, which means, that the dependency was set for a property, but not for the one, which is symmetric to it (Schon *et al.*, 2018). These kind of dependencies are not supported by the current Cascading Forgetting.
From the engineering perspective there were some further results beside the quantification of the quality of the Cascading Forgetting operator. All participants were largely in agreement about the forgetting operations, which is an indicator, that the modelling of the ontology and the forgetting operations are intuitive and largely understandable to the engineers. Thus, IF scenarios can be generalized, for instance the starting point is often a function or a requirement. Moreover, the largely agreement of the different participants is interpreted, that the semantics are understandable for experts and novices of a specific technical system, when they have an engineering background. Therefore, the ontology itself can be used for the presentation of the knowledge. However, some questions arise during the evaluation regarding the modelling of the ontology. Some individuals were attached to more than one class, for instance, some subassemblies are also classified as assemblies, because every subassembly is also an assembly. Such double classifications lead to confusion when deciding on forgetting operations. Therefore, this issue will be considered in future semantic models. The ontology was optimized, thus the above mentioned issues were resolved. The missing metaproperties were set and the modelling errors were fixed, thus the recall rises to 0.7, while the precision remain unchanged (Schon *et al.*, 2018).

4.3 Automating the setting of metaproperties for Cascading Forgetting

For a better handling of Cascading Forgetting in product design, the dependencies between the classes are stored in a spreadsheet (see Table 4). Due to the questionnaire the forgetting pairs are extracted, which were deleted together by the engineers, even if they are not all yet supported by the Cascading Forgetting operator. Therefore, Table 4 shows the pairs, which are connected through the dependency metaproperty. As described in Section 3.2 an individual is rejected with another individual, on which it depends. For instance, an individual x of class **function** is deleted, if the individual y of class **requirement** is rejected, on which x depends with respect to (w.r.t) the relation *derivedFrom*. These kind of rules are true for the entries (2) to (6) of Table 4. The pairs (7) and (8) have a special role, because a dependency metaproperty is just set, if there is no **subfunction** or rather **subassembly**, which is connected to the **solution_principle** or rather to the part.

Table 4. Table of metaproperties for Cascading Forgetting in product design

(1)	**requirement**	rigid	
(2)	**function**	rigid	dependsOn **requirement** w.r.t. derivedFrom
(3)	**subfunction**		dependsOn **function** w.r.t. givesSupportTo
(4)	**solution_principle**		dependsOn **subfunction** w.r.t. fulfils
(5)	**subassembly**		dependsOn **assembly** w.r.t. isSubassemblyOf
(6)	**part**		dependsOn **subassembly** w.r.t. isPartOf
(7)	**solution_principle**		dependsOn **function** w.r.t. fulfils
(8)	**part**		dependsOn **assembly** w.r.t. isPartOf
(9)	**subassembly**		consists of: [at least 1 part and 1 subassembly] or 2 parts or 2 subassemblies
(10)	**asssembly**		consists of: at least 1 part and 1 subassembly or 2 parts or 2 subassemblies

https://doi.org/10.1017/dsi.2019.259 Published online by Cambridge University Press

For the pairs (9), and (10) we need a special dependency, which can count, because an assembly is only an assembly, if it contains at least two parts or two subassemblies or a subassembly and a part. This is equal to the subassemblies. Those counting dependencies are not possible with Cascading Forgetting yet, but part of future research. The "rigid" property is added in a separate column to the rigid classes. Due to the Protégé-Plugin Cellfie[1], the spreadsheet can be automatically loaded as annotations to the existing classes using rules in Manchester Syntax (Horridge and Patel-Schneider, 2012). The control by spreadsheets is much easier to handle and the usability can be improved by a template, thus only the class pairs, which depend on each other, and the relation have to be chosen from a list manually.

5 CONCLUSION AND FUTURE WORK

Due to the corporate performance of the evaluation, three main perceptions are gained. First of all, the evaluation shows, that metaproperty-guided Cascading Forgetting rejects related information reliably, even it is a little bit too cautious for now and needs some more dependencies, e. g. counting dependencies. Second, ontologies are a powerful communication base for bridging the gap between interdisciplinary caused misunderstandings, as long all participants have a basic understanding of the logical structure of ontologies. This makes ontologies a quite good tool for knowledge engineers to design uniform and universally understandable knowledge and information structures that can be used for analysis. At least, Cascading Forgetting and the ontological notion of product design information is intuitive to engineers, even if they did not work with ontologies before. In the questionnaire, Cascading Forgetting was evaluated only with regard to pairs of statements. However, the operator for Cascading Forgetting also deletes larger amounts of assertions, depending on the specified metaproperties. These intelligent deletions provide a great support to product designers, by supporting the rejection of related elements, especially if the connections are not obvious.
However, as mentioned before, some modelling errors occurred during the evaluation. Therefore, the optimization of the basic ontology structure is part of future research. For a more robust design of a product design ontology, which integrates and connects all relevant information for reuse and adaptation, suitable Ontology Design Patterns (ODPs) will be searched. These are reusable patterns and small ontology excerpts, which are validated for specific tasks, thus they reduce the occurrence of modelling errors. Due to an ontological integration of reusable product design knowledge and the use of Intentional Forgetting, reuse and adaptation becomes more efficient and transparent.

REFERENCES

Alchourrón, C.E., Gärdenfors, P. and Makinson, D. (1985), "On the logic of theory change. Partial meet contraction and revision functions", *The journal of symbolic logic*, Vol. 50 No. 2, pp. 510–530. http://dx.doi.org/10.2307/2274239.

Baader, F. (2010), *The description logic handbook: Theory, implementation, and applications*, Cambridge University Press, Cambridge. http://dx.doi.org/10.1017/CBO9780511711787.

Baxter, D., Gao, J., Case, K., Harding, J., Young, B., Cochrane, S. and Dani, S. (2007), "An engineering design knowledge reuse methodology using process modelling", *Research in Engineering Design*, Vol. 18 No. 1, pp. 37–48. http://dx.doi.org/10.1007/s00163-007-0028-8.

Camarillo, A., Ríos, J. and Althoff, K.-D. (2018), "Knowledge-based multi-agent system for manufacturing problem solving process in production plants", *Journal of Manufacturing Systems*, Vol. 47, pp. 115–127. http://dx.doi.org/10.1016/j.jmsy.2018.04.002.

Gärdenfors, P. (1992), "Belief revision. An introduction", In Gärdenfors, P. (Ed.), *Belief Revision*, Cambridge University Press, Cambridge, pp. 1–28. http://dx.doi.org/10.1017/CBO9780511526664.001.

Gearon, P., Passant, A. and Polleres, A. (2013), *SPARQL 1.1 Update*. [online] W3C, Available at: http://www.w3.org/TR/2013/REC-sparql11-update-20130321/ (accessed 12 November 2018).

Gruber, T.R. (1993), "A translation approach to portable ontology specifications", *Knowledge Acquisition*, Vol. 5 No. 2, pp. 199–220. http://dx.doi.org/10.1006/knac.1993.1008.

Hagedorn, T.J., Krishnamurty, S. and Grosse, I.R. (2018), "A Knowledge-Based Method for Innovative Design for Additive Manufacturing Supported by Modular Ontologies", *Journal of Computing and Information Science in Engineering*, Vol. 18 No. 2, p. 21009. http://dx.doi.org/10.1115/1.4039455.

Harris, S., Seaborne, A. and Prud'hommeaux, E. (2013), *SPARQL 1.1 query language*. [online] W3C, Available at: http://www.w3.org/TR/2013/REC-sparql11-query-20130321/ (accessed 12 November 2018).

1 https://github.com/protegeproject/cellfie-plugin

https://doi.org/10.1017/dsi.2019.259 Published online by Cambridge University Press

Horridge, M. and Patel-Schneider, P.F. (2012), "*OWL 2 Web Ontology Language*". *Manchester Syntax (Second Edition)*, W3C Working Group Note. [online] W3C, Available at: https://www.w3.org/TR/2012/NOTE-owl2-manchester-syntax-20121211/ (accessed 19 February 2018).
Jackson, T.W. and Farzaneh, P. (2012), "Theory-based model of factors affecting information overload", *International Journal of Information Management*, Vol. 32 No. 6, pp. 523–532. http://dx.doi.org/10.1007/s00163-007-0028-8.
Kestel, P., Luft, T., Schon, C., Kügler, P., Bayer T, Schleich, B. and Wartzack, S. (2017), "Konzept zur zielgerichteten, ontologiebasierten Wiederverwendung von Produktmodellen". *Design for X, Bamberg*, 04.-05. Oktober 2017, TuTech, Hamburg, pp. 241–252.
Kügler, P., Kestel, P., Schon, C., Marian, M., Schleich, B., Staab, S. and Wartzack, S. (2018), "Ontology-based approach for the use of Intentional Forgetting in product development". *15th International Design Conference*, May, 21-24, 2018, The Design Society, Glasgow, pp. 1595–1606. http://dx.doi.org/10.21278/idc.2018.0402.
Li, Z., Zhou, X., Wang, W.M., Huang, G., Tian, Z. and Huang, S. (2018), "An ontology-based product design framework for manufacturability verification and knowledge reuse", *The International Journal of Advanced Manufacturing Technology*, Vol. 26 No. 1, p. 139. http://dx.doi.org/10.1007/s00170-018-2099-2.
Marian, M., Tremmel, S. and Wartzack, S. (2018), "Microtextured surfaces in higher loaded rolling-sliding EHL line-contacts", *Tribology International*, Vol. 127, pp. 420–432. http://dx.doi.org/10.1016/j.triboint.2018.06.024.
Markus, L.M. (2001), "Toward a theory of knowledge reuse. Types of knowledge reuse situations and factors in reuse success", *Journal of management information systems*, Vol. 18 No. 1, pp. 57–93. http://dx.doi.org/10.1080/07421222.2001.11045671.
Probst, G., Raub, S. and Deussen, A. (2002), *Kompetenz-Management: Wie Individuen und Organisationen Kompetenz entwickeln*, Gabler, Wiesbaden.
Rude, S. (1998), *Wissensbasiertes Konstruieren*, Shaker, Herzogenrath.
Schon, C. and Staab, S. (2017), "Towards SPARQL instance-level Update in the Presence of OWL-DL TBoxes". *Joint Ontology Workshop*, Bozen-Bolzano, CEUR-WS.org.
Schon, C., Staab, S., Kügler, P., Kestel, P., Schleich, B. and Wartzack, S. (2018), "Metaproperty-guided deletion from the instance-level of a knowledge base". *21st International Conference on Knowledge Engineering and Knowledge Management*, Nancy, 12.-16. 11.2018, Springer, Cham, pp. 407–423. http://dx.doi.org/10.1007/978-3-030-03667-6_26.
Seel, N.M. (2012), *Encyclopedia of the Sciences of Learning*, Springer US, Boston. http://dx.doi.org/10.1007/978-1-4419-1428-6.
Shenk, D. (2007), *Data smog: Surviving the information glut*, Rev.ed., HarperCollins, New York.

ACKNOWLEDGEMENT

The authors gratefully acknowledge the financial support of project WA 2913/22-1 and STA 572/15-1 within the Priority Program 1921, by the German Research Foundation (DFG).

https://doi.org/10.1017/dsi.2019.259 Published online by Cambridge University Press