Reasoning on Engineering Knowledge: Applications and Desired Features

Constantin Hildebrandt[1(✉)], Matthias Glawe[1], Andreas W. Müller[2], and Alexander Fay[1]

[1] Institute of Automation Technology, Helmut-Schmidt-University, Hamburg, Germany
{c.hildebrandt,matthias.glawe,alexander.fay}@hsu-hh.de
[2] Data Architecture and Frameworks, Schaeffler Technologies AG & Co. KG, Herzogenaurach, Germany
andreas_w.mueller@schaeffler.com

Abstract. The development and operation of highly flexible automated systems for discrete manufacturing, which can quickly adapt to changing products, has become a major research field in industrial automation. Adapting a manufacturing system to a new product for instance requires comparing the systems functionality against the requirements imposed by the changed product. With an increasing frequency of product changes, this comparison should be automated. Unfortunately, there is no standard way to model the functionality of a manufacturing system, which is an obstacle to automation. The engineer still has to analyze all documents provided by engineering tools like 3D-CAD data, electrical CAD data or controller code. In order to support this time consuming process, it is necessary to model the so-called skills of a manufacturing system. A skill represents certain features an engineer has to check during the adaption of a manufacturing system, e.g. the kinematic of an assembly or the maximum load for a gripper. Semantic Web Technologies (SWT) provide a feasible solution for modeling and reasoning on the knowledge of these features. This paper provides the results of a project that focused on modeling the kinematic skills of assemblies. The overall approach as well as further requirements are shown. Since not all expectations on reasoning functionality could be met by available reasoners, the paper focuses on desired reasoning features that would support the further use of SWT in the engineering domain.

1 Introduction

Decreasing life-cycles and lot sizes [1], increasing numbers of product variants [2] as well as more decentralized manufacturing systems [3] have become a business standard in the manufacturing industry. These circumstances create a dynamic environment, which state-of-the-art manufacturing systems struggle with, since current manufacturing systems are not flexible enough yet to handle this dynamic environment [4]. Applying Semantic Web technologies (SWT) in the engineering domain has become a promising approach in order to gain flexibility in the industrial manufacturing domain. Especially for repetitive tasks, the use of knowledge-based systems provides advantages in terms

E. Blomqvist et al. (Eds.): ESWC 2017, Part II, LNCS 10250, pp. 65–78, 2017.
DOI: 10.1007/978-3-319-58451-5_5

of effort savings, time savings and quality assurance [5–7]. Concerning the operations planning of a manufacturing system, ontologies may be used for modeling the knowledge about the planning domain [8–10]. Once the domain knowledge has been modeled, it can be used for resource allocation, planning and scheduling. As shown in these and other contributions, modeling of manufacturing domain knowledge can be accomplished successfully by using ontologies. On top of such ontologies, rules or queries can be defined to infer additional knowledge for the manufacturing system, e.g. for maintenance [11, 12], security [13], validation [14] or process planning [15]. Typical results of reasoning could be a possible process plan, a security threat or a necessary maintenance task. Especially for these applications, reasoning support is crucial [16] and determines the success of using SWT in the engineering domain. Unfortunately, the use of SWT for reasoning on engineering knowledge seems to lack in support of some features which are necessary in the engineering domain and are addressed by this paper. Therefore, an application from the engineering domain is described in this paper which requires certain reasoning features. The availability of these reasoning features in SWT would significantly support further applicability and, thus, industrial acceptance of SWT in the engineering domain. The structure of this paper is as follows. Section 2 introduces the goal and concept of the underlying research project. Section 3 shows how the knowledge base was built in OWL, while Sect. 4 describes the requirements and desired features on reasoning that have been identified and which aspects are not fulfilled up to now. Section 5 provides a summary and an outlook.

2 Concept of Extracting Knowledge About Skills

According to [17] a skill is the ability of a resource to implement a certain type of manufacturing, logistic or other production related process. In order to represent the skill of a manufacturing system with SWT, a concept for the automatic generation of a skill description based on engineering data has been created by the authors. In this concept, depicted in Fig. 1, an engineer performs design work as usual with a 3D-CAD tool. In the CAD tool, design results are stored in a native file that contains information usable for inferring the mechanical skills of the system. Information stored by the CAD tool are e.g. the modeled kinematic and the material of the mechanical components. From the modeled kinematic, the potentially reachable positions can be inferred, and the chosen material is important to determine the maximum load of a gripping unit. This CAD file, which is usually in a proprietary tool format, is imported by a so-called "Mapping Component" (MAP). Since CAD files may contain lots of information, the purpose of the MAP is to obtain the necessary information for inferring a skill from the CAD file in a semantically proper way and inserting it into a target ontology, which contains all needed concepts and properties. This mapping is based on a concept that was initially introduced by [13] for mapping Computer Aided Engineering Exchange (CAEX) data into an ontological structure, and works as follows. The MAP imports a set of SWRL [21] rules that describe which entities or relations in the CAD file should be mapped to the corresponding structures in the target ontology. The rule's antecedent describes the entities and relation that need to be found in the CAD file, the consequent

describes the individuals, object properties and data properties that need to be created in the ontology, whenever the entities in the antecedent are found. Hence, the mapping rules provide a possibility to define the mapping procedure on an abstract semantic level and are, thus, independent of the CAD data format used.

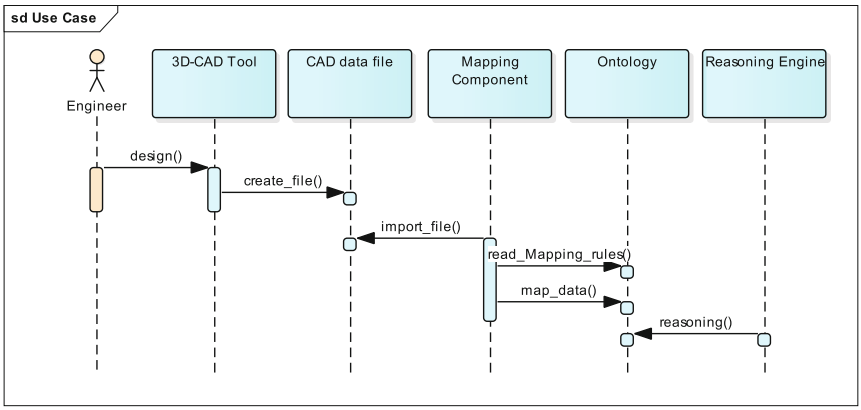

Fig. 1. Sequence diagram of the mapping process

An example of a SWRL rule for the MAP is shown in Listing 1. As the example illustrates, if the MAP finds an entity `?x` in the CAD file, that is part of an entity `?y`, the object property `consistsOf` should link the two respective individuals in the ontology. While the antecedent does not have any inner meaning to the ontology, it triggers certain methods in the MAP, which represents a more or less complex analysis on the CAD file.

Listing 1. example of a mapping rule

```
is_CADPartOf(?x, ?y) -> consistsOf(?y, ?x)
```

After finalizing the mapping procedure, the ontology contains all relevant information modeled in the CAD file, which is needed for further inference. The representation of this information is shown in Sect. 3.

3 Representing Engineering Data in an OWL Ontology

This section shows the modeling of engineering data in the underlying project. As an excerpt, the use-case on the left side of Fig. 2 is introduced as an example for a kinematic skill. The modeled object represents an assembly that contains two linear actors and a gripper. The following information is explicitly included in the exemplary 3D CAD file:

(I1) Information about the existence of objects, i.e. components and assemblies
(I2) Information about the hierarchy of objects, i.e. the parts an assembly consists of

(I3) Geometric information about the shape of objects
(I4) Geometric information about the position of objects
(I5) Information about movement restrictions of parts, i.e. how a part can or cannot move
(I6) Further details, e.g. an assumed *Tool Center Point* or material information

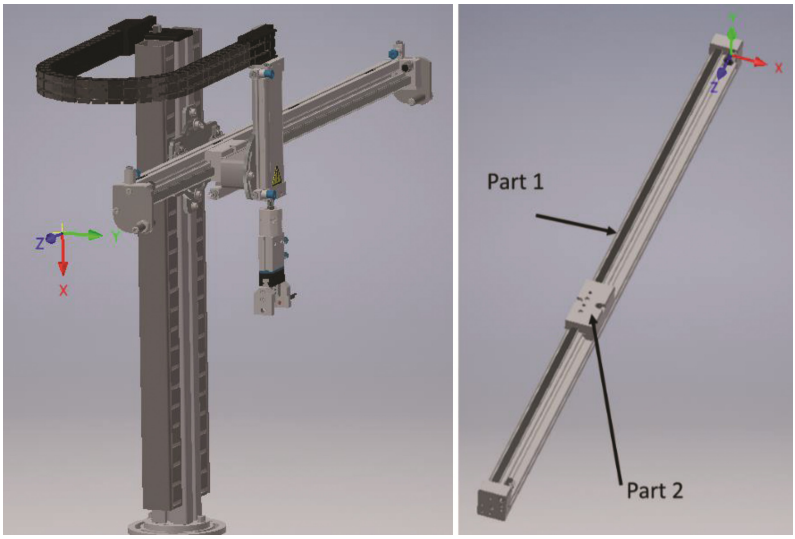

Fig. 2. 3D-CAD complete assembly (left) and linear actor assembly (right)

The ontology used as the target for the mapping was modeled in Protégé. Three main concepts were used in order to model the domain information (see left side of Fig. 3). The concept `Description` subsumes concepts that represent mathematical and physical descriptions of real world entities, e.g. vector descriptions. The concept `ProductionSystem` classifies the hierarchical structure of a production system, while the concept `FunctionalSkill` is used for classifying skills of a production system. The object properties that were used for modeling relations between objects are shown on the right side of Fig. 3, while the data properties are shown in Fig. 4.

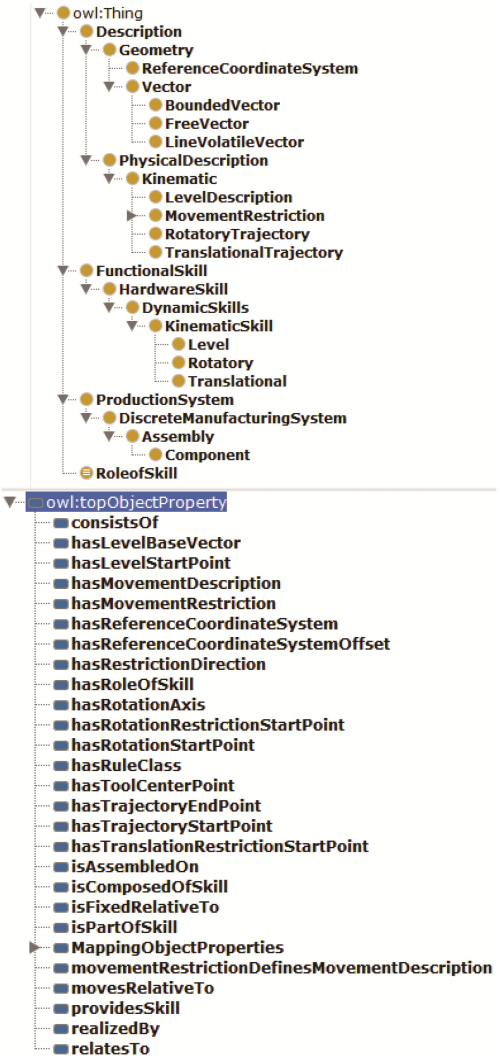

Fig. 3. Classes of the ontology (left), object properties of the ontology (right)

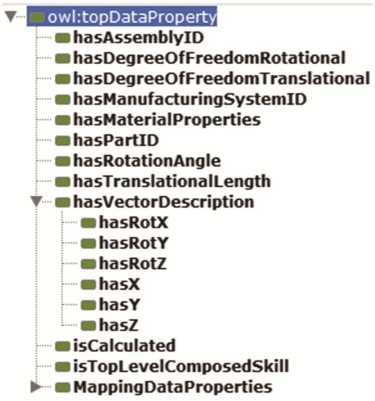

Fig. 4. Data properties of the ontology

The assembly shown in Fig. 2 consists of two components `Part1` and `Part2`. This corresponds to information of categories I1 and I2 according to the enumeration above. Furthermore, `Part2` possesses movement restrictions, where one movement restriction describes explicitly how the part can move along a single axis (I5). In the depicted case it is the individual `MovementRest_Trans_1`, which is member of the class `MovementRestriction`.

In order to represent the respective movement restrictions, it is necessary to use individuals for representing vectors. Hence, every individual `?x` that is a member of a vector class (i.e. `BoundedVector(?x)` or `FreeVector(?x)` or `LineVolatileVector(?x)`) (see Fig. 3 left side) has at least three data properties. These data properties are `hasX(?x,float)`, `hasY(?x,float)`, `hasZ(?x,float)`. Since every vector has to have a reference coordinate system (RCS) for its interpretation, every part or assembly has a RCS (I4). The Tool Center Point, representing the desired point to attach a tool or another assembly to the actual assembly, is also represented by a vector description (`Vector_TCP_P2`) (I6). A representation of the geometrical shape of the CAD objects in the ontology was not necessary, since the focus was on modeling kinematics.

After the mapping procedure is finished, the ontology contains the necessary information for the reasoning process, which was beforehand modeled in the CAD file. A small example of mapped individuals is shown in Fig. 5.

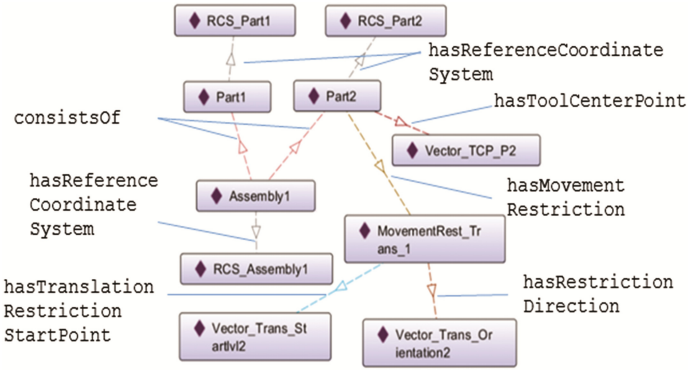

Fig. 5. Example of mapped individuals

In order to support an engineer in checking the functionality of the system, e.g. checking the kinematic, it is now necessary to infer information about the combined movement descriptions of the contained parts. This is essential, since there is no direct information in the CAD file about the combination of single parts movements. The knowledge on how to derive combined kinematics from basic descriptions has been modeled with SWRL. Query-based inference like SPARQL Protocol and RDF Query Language combined with SPARQL Inferencing Notation (as was done by [12] for instance) were also considered. However, the goal was to maintain continuous logical expressivity throughout the entire process of deriving skill descriptions. Thus, SWRL was chosen as it allows for handling and processing knowledge at the OWL level.

For the purpose of mapping and inferencing, a total of 17 rules has been defined. Nine rules are used for mapping and eight rules are used for inferring knowledge about the kinematic skill. Due to the paper's extend, the next section will only introduce a few rule examples, while the next section focuses on desired reasoning features in this context.

4 Reasoning on Engineering Information

As it is pointed out in [16] the reasoning support is a very important requirement when it comes to choosing the modeling language. This section describes desired features in reasoning from an engineering point of view, as found in the project described in Sects. 2 and 3. Beforehand, a basic requirement is introduced as R0 that is based on recent initiatives in the engineering domain.

4.1 R0: Standardized Rule Functionality

Manufacturing systems are becoming more flexible by using data for smart concepts like diagnosis systems [11] or even context-aware maintenance systems [12]. However, as soon as systems need to collaborate with other systems, it is essential for the systems to share a common understanding of concepts, i.e. a common semantic [18]. If a

manufacturing execution system for instance needs to analyze the skills of different manufacturing systems, all manufacturing systems have to use the same semantic about the modeled knowledge of the skill. If rules are used for knowledge modeling, the rule functionality has to be standard-based as well. Otherwise, there is no common understanding about the modeled knowledge. Thus, the first requirement is about standard rule functionality. If there is no standard "off-the-shelf" reasoner available for a reasoning task, then the interpretation of concepts is becoming an issue for engineering applications. This receives attention by initiatives like the German Industry 4.0 [18] or the American Industrial Internet of Things initiative [19], where physical production systems and information technology grow together. The authors of [20] for instance present an approach for orchestrating the resources of small companies according to an order. The authors used SWRL and a rule engine in order to define additional functionality. Sharing the knowledge about orchestration is becoming an issue with ongoing decentralization of systems without a common understanding about rule functionalities.

Since SWRL based on [21] is a de-facto standard for rules in Semantic Web, it was chosen to state the rules for extracting kinematic skills. The target skill to be inferred by the reasoning process, is visualized as a plane in Fig. 6 (left side). Since there are two linear actors that do not move in the same direction, the resulting kinematic skill is represented by a plane that has a certain height and width. This plane description relates to the RCS of the highest assembly in the object hierarchy. This is the desired result, which has to be inferred through rule-based inference from the information input. The information input is shown on the right side of Fig. 6. To ensure comparable results in different applications, a common reasoner should be used to infer the information needed to represent the desired skill.

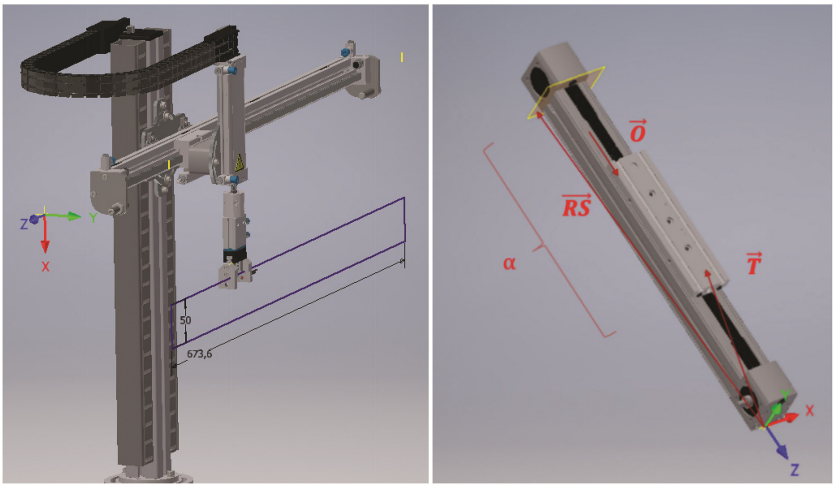

Fig. 6. Kinematic skill of the assembly (left side), geometric description of a linear actor (right side)

4.2 D1: Creating New Individuals Representing Generated Knowledge

The first desired feature (D1) relates to creating individuals in order to represent inferred knowledge. Since the information about a skill is something that was not explicitly created by an engineer, a possibility is needed to create individuals that represent this inferred skill information. For instance (see Listing 2), if there is an assembly `?a` that has a RCS `?RCSa` and consists of a part `?c` that has at least one degree of freedom, the movement restrictions of this component `?c` define a kinematic `?Kin` that was not known to the model before. Therefore, an individual to represent this kinematic has to be created and further detailed. For creating individuals, the SWRL syntax provides an extension through the SWRL Extensions built-in library[1] called `swrlx:makeOWL-Thing(?a,?b)`, where `?a` defines the variable that binds the individual and `?b` defines the number of individuals to be created.

Listing 2. Creating new individuals with SWRL extension

```
Assembly(?a), hasReferenceCoordinateSystem(?a,?RCSa),
consistsOf(?a,?c), Component(?c),
hasDegreeOfFreedomTranslational(?c,?DoFT),
hasDegreeOfFreedomRotational(?c,?DoFR),
swrlb:add(?DoFSum,?DoFT,?DoFR),
swrlb:greaterThan(?DoFSum,0),
swrlx:makeOWLThing(?Kin,1) ->
movesRelativeTo(?c,?RCSa),
movementRestrictionDefinesMovementDescription(?c,?Kin),
Kinematic(?Kin)
```

As stated in [22] this is not supported by DL Reasoners of Protégé, although the Protégé SWRL tab supports the syntax. For this reason, it became necessary to use an external rule engine in order to define the functionality for this extension. Unfortunately, this violates requirement R0. In general, many engineering applications require the generation of new knowledge and therefore the creation of new individuals when using an ontology. Therefore, creating individuals with a defined functionality through the use of a maintained reasoner as analyzed in [23] is a desired feature in engineering.

4.3 D2: Calculations

Knowledge in the engineering domain is often based on mathematical descriptions like vectors for instance. In cases where this knowledge is processed or evaluated in a new context, mathematical operations need to be performed. For the extraction of kinematic skills numerous operations are necessary, e.g. calculation of RCS offsets, combination of translational/rotational kinematics or combination of planes or even solid body descriptions. For SWRL only basic mathematical operations are defined by the Built-In

[1] http://swrl.stanford.edu/ontologies/built-ins/3.3/swrlx.owl.

library [21]. Due to the fact that even increasing a vector length by one leads to the use of three `swrlb:add` operators, the rules are becoming very large. This in turn leads to problems in creating or maintaining the rules. For instance, creating a translational kinematic skill out of a part's movement restrictions requires the following calculation (see also right side of Fig. 6):

$$\overrightarrow{sv} = \vec{T} + \left(\frac{\vec{O} \cdot \left(\overrightarrow{RS} - \vec{T} \right)}{\vec{O} \cdot \vec{O}} \right) \vec{O}; \overrightarrow{ev} = \vec{S} + \alpha \vec{O}$$

In this calculation, $\overrightarrow{sv}$ would point to the start and $\overrightarrow{ev}$ to the end of the translation. In order to define this as a rule, it requires 16 basic math operations and the plane calculation in Fig. 6 requires even more. This number of course does not include the rest of the antecedent and the consequent that have to be defined. In order to solve the problem, a rule engine for defining easier-to-handle math operators had to be used. A desirable solution would be a Protégé plug-in, where complex math operations, e.g. vector algebra, can be created through combination of basic math operations so that R0 is still satisfied and using math operations in SWRL is becoming more flexible. To the best of the authors' knowledge, there are no ways to create such plugins in an easy-to-use way and to implement them in maintained reasoners.

4.4 D3: Defeasible Reasoning

Most engineering methods are based on an iterative procedure, where knowledge about objects is created, updated and sometimes even fully retracted afterwards. In [24] for instance, the whole engineering process from the first idea to the detailed model may be in a feedback loop. The reason for this is that the engineering process is sliced into phases where sometimes assumptions have to be made in order to proceed. These assumptions about an object may be retracted as soon as new knowledge is accessible or processed. Revisiting the extraction of the kinematic skill it is possible to define a rule for the combination of two translational skills of the two linear actors in Fig. 6 (left side). This is shown in Listing 3. If an assembly `?a3` exists, which contains two different assemblies `?a1` and `?a2` that provide a calculated skill, then these two assemblies `?a1` and `?a2` provide a new kinematic skill `?s1`. This new kinematic skill is shown in Fig. 6 (left side). In order to represent this skill as the top-level skill, so that it can be found by querying for instance, the data property `isTopLevelComposedSkill(?s1,true)` should be assigned to the skill `?s1`. As soon as another overlying assembly (e.g. the gripper) is found during the process, the value of the data property needs to be changed to false since `?s1` does not represent the top-level skill anymore. Furthermore, retracting knowledge is also necessary for calculation. If a skill should be calculated by combining two skills, it is necessary to indicate whether it has already been calculated before (`isCalculated(?s1,true)`). Otherwise, stepwise calculations under open world assumption would not be possible, because the actual calculation step could not be determined.

Listing 3. Declaring the state of knowledge after applying a rule

```
Assembly(?a1), Assembly(?a2), Assembly(?a3),
consistsOf(?a1,?a2), consistsOf(?a1,?a3),
providesSkill(?a2,?s2), providesSkill(?a3,?s3),
swrlb:notEqual(?a2,?a3), isCalculated(?s2,true),
isCalculated(?s3,true), consistsOf(?a2,?c2),
Component(?c2), consistsOf(?a3,?c3), Component(?c3),
hasToolCenterPoint(?c3,?TCPc3),
isAssembledOn(?c2,?TCPc3), swrlx:makeOWLThing(?s1,1),
swrlx:makeOWLThing(?Kina1,1) -> KinematicSkill(?s1),
providesSkill(?a1,?s1), hasRoleOfSkill(?s1,Combined),
isComposedOfSkill(?s1,?s2), isComposedOfSkill(?s1,?s3),
isCalculated(?s1,false),
hasMovementDescription(?s1,?Kina1), Kinematic(?Kina1),
isTopLevelComposedSkill(?s1,true)
```

Even though this example is simple, retracting knowledge occurs at various points during the engineering process. Therefore, reasoning on engineering knowledge requires defeasible reasoning features, so that facts or assumptions can be changed through inference whenever downstream facts make this necessary.

It was expected that declaring the data property that needs to be changed (e.g. `isCalculated`) as a functional data property would help to solve this problem. However, no reasoner supports checking the characteristics of data properties while processing the rules. In order to solve this issue, an own rule engine had to be used, which checks whether an existing data property is functional and in this case changes the value of the data property. Again, this leads to a violation of R0. Even though there has been recent research interest on defeasible reasoning [25–27], none of the reasoners that are analyzed by [23] provides this feature yet.

4.5 Implementation

A rule engine for executing the mapping rules was created based on the OWL API.NET (https://owlapinet.codeplex.com/). This rule engine was designed to infer the mapping rules and to create the required OWL individuals. Due to the limitations described before, the authors decided to extend this rule engine to infer the non-mapping rules as well. The rule engine´s implementation was focused on solving the actual problem. To do so, methods to create new individuals (using `swrlx:makeOWLThing` operator) and to execute calculations (using self defined SWRL math operators) were implemented. To support the needed aspect of defeasible reasoning the rule engine interpretes data properties with functional characteristics like flags. If an inferred data property exists, the data property´s value is changed to the actual inferred value (for an example see Listing 3).

Although the rule engine was able to solve the given problem, it is very proprietary and implements only those parts that were needed to solve the given use case.

Furthermore, constant maintenance and enhancement cannot be ensured. For transferring such problems to a wider scale the implementation of the described desired features within an open and maintained reasoner is be needed.

5 Summary, Conclusion and Outlook

In this paper, an application of Semantic Web technologies in the engineering domain was introduced. The use-case of the application is a system that should support an engineer in checking the functionality (e.g. the kinematics) of a manufacturing system in order to adapt it to new environments. Therefore, a solution for modeling 3D-CAD information in an OWL ontology was shown, while the CAD data was automatically mapped into the OWL ontology using a mapping component based on standard technology. Since the needed information about the kinematic of a manufacturing system is not contained explicitly in the CAD data, it had to be made explicit through rule-based inference. Due to the fact, that a standardized semantic is a basic requirement for Industry 4.0 or similar initiatives, the utilization of both an "off-the-shelf" reasoner and a standardized rule language were defined as a project goal. Unfortunately, some compromises about this basic requirement had to be made, since available reasoners did not support all necessary features for this application. In order to support the use of Semantic Web technologies in the engineering domain, three desired features were derived in this paper. Including these features in an actively maintained reasoner would support the use of Semantic Web technologies in the engineering domain, since the use of project-specific rule functionality defined with rule engines would not be necessary anymore.

References

1. Wiendahl, H.-P., ElMaraghy, H.A., Nyhuis, P., Zäh, M.F., Wiendahl, H.-H., Duffie, N., Brieke, M.: Changeable manufacturing - classification, design and operation. CIRP Ann. Manufact. Technol. **56**(2), 783–809 (2007). doi:10.1016/j.cirp.2007.10.003
2. Hu, S.J., Zhu, X., Wang, H., Koren, Y.: Product variety and manufacturing complexity in assembly systems and supply chains. CIRP Ann. Manufact. Technol. **57**(1), 45–48 (2008). doi:10.1016/j.cirp.2008.03.138
3. Mourtzis, D., Doukas, M.: Decentralized manufacturing systems review challenges and outlook. Logist. Res. **5**(3–4), 113–121 (2012). doi:10.1007/s12159-012-0085-x
4. Vogel-Heuser, B., Diedrich, C., Fay, A., Jeschke, S., Kowalewski, S., Wollschlaeger, M., Göhner, P.: Challenges for software engineering in automation. JSEA **07**(05), 440–451 (2014). doi:10.4236/jsea.2014.75041
5. Strube, M., Runde, S., Figalist, H., Fay, A.: Risk minimization in modernization projects of plant automation — a knowledge-based approach by means of semantic web technologies. In: Factory Automation (ETFA 2011), Toulouse, France, pp. 1–8 (2011)
6. Runde, S., Fay, A.: Software support for building automation requirements engineering—an application of semantic web technologies in automation. IEEE Trans. Ind. Inf. **7**(4), 723–730 (2011). doi:10.1109/TII.2011.2166784
7. Legat, C.: Knowledge-based technologies for future factory engineering and control. IFAC Proc. Volumes **45**(6), 44–48 (2012). doi:10.3182/20120523-3-RO-2023.00447

8. Puttonen, J., Lobov, A., Lastra, M.: Semantics-based composition of factory automation processes encapsulated by web services. IEEE Trans. Ind. Inf. **9**(4), 2349–2359 (2013). doi: 10.1109/TII.2012.2220554
9. Legat, C., Schütz, D., Vogel-Heuser, B.: Automatic generation of field control strategies for supporting (re-)engineering of manufacturing systems. J. Intell. Manuf. **25**(5), 1101–1111 (2014). doi:10.1007/s10845-013-0744-z
10. Harcuba, O., Vrba, P.: Ontologies for flexible production systems. In: 20th Conference on Emerging Technologies and Factory Automation (ETFA). International Conference on Emerging Technologies & Factory Automation,. Luxembourg, Luxembourg. IEEE, Institute of Electrical and Electronics Engineers, Piscataway (2015)
11. Bunte, A., Diedrich, A., Niggemann, O.: Integrating semantics for diagnosis of manufacturing systems. In: 21st IEEE International Conference on Emerging Technologies and Factory Automation, ETFA, Berlin, Germany, 6–9 September 2016. IEEE, Institute of Electrical and Electronics Engineers (2016)
12. Aarnio, P., Vyatkin, V., Hästbacka, D.: Context modeling with situation rules for industrial maintenance. In: 21st IEEE International Conference on Emerging Technologies and Factory Automation, ETFA. Berlin, Germany, 6–9 September 2016. IEEE, Institute of Electrical and Electronics Engineers (2016)
13. Glawe, M., Tebbe, C., Fay, A., Niemann, K.-H.: Knowledge-based engineering of automation systems using ontologies and engineering data. In: 7th International Conference on Knowledge Engineering and Ontology Development, Lisbon, Portugal, 12–14 November 2015
14. Abele, L., Legat, C., Grimm, S., Muller, A.W.: Ontology-based validation of plant models. In: IEEE 11th International Conference on Industrial Informatics (INDIN), Bochum, Germany, pp. 236–241 (2015)
15. Ramis Ferrer, B., Ahmad, B., Vera, D., Lobov, A., Harrison, R., Martínez Lastra, J.L.: Product, process and resource model coupling for knowledge-driven assembly automation. Automatisierungstechnik **64**(3), 231–243 (2016). doi:10.1515/auto-2015-0073
16. Negri, E., Fumagalli, L., Garetti, M., Tanca, L.: Requirements and languages for the semantic representation of manufacturing systems. Comput. Ind. **81**, 55–66 (2016). doi:10.1016/j.compind.2015.10.009
17. Pfrommer, J., Stogl, D., Aleksandrov, K., Escaida Navarro, S., Hein, B., Beyerer, J.: Plug & produce by modelling skills and service-oriented orchestration of reconfigurable manufacturing systems. Automatisierungstechnik **63**(10), 790–800 (2015). doi:10.1515/auto-2014-1157
18. Usländer, T., Epple, U.: Reference model of industrie 4.0 service architectures. Automatisierungstechnik **63**(10), 858–866 (2015). doi:10.1515/auto-2015-0017
19. Industrial Internet Consortium: Industrial Internet Reference Architecture (2015). https://www.iiconsortium.org/IIRA-1-7-ajs.pdf
20. Jules, G.D., Saadat, M., Li, N.: On designing a unified ontology for holonic manufacturing networks. In: Fathi, M. (ed.) Integration of Practice-Oriented Knowledge Technology: Trends and Prospectives, pp. 207–220. Springer, Heidelberg (2013)
21. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: a semantic web rule language combining OWL and ruleML. In: W3C-World Wide Web Consortium. https://www.w3.org/Submission/SWRL/#8, zuletzt geprüft am 16 September 2016

22. Roda, F., Zanni-Merk, C.: An intelligent data analysis framework for supporting perception of geospatial phenomena. In: Ferrario, R., Kuhn, W. (eds.) Formal ontology in information systems. In: Proceedings of the 9th International Conference (FOIS 2016). Frontiers in artificial intelligence and applications, vol. 283. IOS Press, Amsterdam (2016)
23. Matentzoglu, N., Leo, J., Hudhra, V., Parsia, B., Sattler, U.: A survey of current, stand-alone OWL Reasoners. In: Informal Proceedings of the 4th International Workshop on OWL Reasoner Evaluation (2015)
24. Pahl, G., Beitz, W., Blessing, L., Feldhusen, J., Grote, K.-H., Wallace, K. (eds.): Engineering Design. A Systematic Approach, 3rd edn. Springer, London (2007). http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10230457
25. Basseda, R., Gao, T., Kifer, M., Greenspan, S., Chell, C.: Representing flexible role-based access control policies using objects and defeasible reasoning. In: Bassiliades, N., Gottlob, G., Sadri, F., Paschke, A., Roman, D. (eds.) RuleML 2015. LNCS, vol. 9202, pp. 376–387. Springer, Cham (2015). doi:10.1007/978-3-319-21542-6_24
26. Rakib, A., Haque, H.M.U.: Modeling and verifying context-aware non-monotonic reasoning agents. In: ACM/IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE), Austin, TX, USA, pp. 61–69 (2015)
27. Casini, G., Meyer, T., Moodley, K., Sattler, U., Varzinczak, I.: Introducing defeasibility into OWL ontologies. In: Arenas, M., et al. (eds.) ISWC 2015. LNCS, vol. 9367, pp. 409–426. Springer, Cham (2015). doi:10.1007/978-3-319-25010-6_27