Semantic-Aware Validation in Model-Driven Requirements Engineering Using SHACL

Artan Markaj[1][a], Felix Gehlhoff[1][b] and Alexander Fay[2][c]
[1]*Institute of Automation Technology, Helmut-Schmidt-University Hamburg, Hamburg, Germany*
[2]*Chair of Automation, Ruhr University Bochum, Bochum, Germany*

Keywords: Ontologies, Semantic Requirements Engineering, SHACL, Traceability, Validation.

Abstract: The development and implementation of technical system concepts require validation to ensure that stakeholder needs, goals, and requirements are fulfilled. Model-driven requirements engineering focuses on the automatic transformation of requirements into concepts and can be supported by ontologies for semantically unambiguous specifications. However, automated and systematic requirements validation using ontologies remains a challenging process. In this contribution, we propose a concept consisting of a systematic workflow, algorithm, and templates for semantic-aware validation in model-driven requirements engineering using *Shapes Constraint Language*, a formal language for constraint-based ontology validation. The workflow begins with the definition of validation use cases from a requirements model. These use cases are modeled as ontologies using the same metamodel as the requirements. By using Shapes Constraint Language templates, shapes can be generated and enriched with use case-specific information. Lastly, engineering concepts are validated against the requirements by using the defined shapes.

1 INTRODUCTION

Validation is one of the final steps in systems engineering (before operation), ensuring that developed and verified systems meet stakeholder requirements, goals and needs (ISO/IEC/IEEE, 2015). However, this process step can be cumbersome without traceability between the developed systems and their corresponding requirements. Traceability links between requirements and developed concepts enable requirements validation (Cleland-Huang et al., 2012). With model-driven requirements engineering, an automatic model transformation and analysis of requirements are supported by high-level abstractions (Moreira et al., 2022). Ontologies can be employed to ensure that the modeled and transformed requirements are semantically unambiguous. An ontology consists of a vocabulary of terms and an associated specification of their meaning (Uschold, 1998).

Ontologies serve as formalized and structured representations of knowledge (Guarino et al., 2009). This is particularly useful within a specific domain, enabling clear communication among stakeholders (e.g., in requirements engineering). A distinction can be made between stakeholders' implicit and explicit knowledge. Implicit knowledge is tied to individuals as knowledge carriers (i.e., experts), making communication and formalization more challenging. Explicit knowledge, by contrast, can be formalized, communicated, and stored (VDI-Kompetenzfeld Informationstechnik, 2009). Furthermore, the knowledge represented in ontologies can be divided into a TBox and an ABox. The former contains general knowledge about a domain (modeled as classes), while the latter contains specific knowledge represented for a particular use case of the domain (modeled as instances) (Baader et al., 2008).

Knowledge related to requirements, such as assumptions, is often undocumented and therefore remains implicit (Maalej and Thurimella, 2013). Additionally, in requirements engineering, knowledge from various stakeholders and sources is integrated, making ontologies suitable for knowledge representation (Dobson and Sawyer, 2006). Ontologies in requirements engineering have advantages in terms of reducing ambiguities, inconsistencies, and incompleteness in requirements (Dermeval et al., 2016). Ontologies can be used to describe and organize re-

[a] https://orcid.org/0000-0003-1589-9584
[b] https://orcid.org/0000-0002-8383-5323
[c] https://orcid.org/0000-0002-1922-654X

Markaj, A., Gehlhoff, F. and Fay, A.
Semantic-Aware Validation in Model-Driven Requirements Engineering Using SHACL.
DOI: 10.5220/0012985200003838
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 16th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2024) - Volume 2: KEOD*, pages 191-198
ISBN: 978-989-758-716-0; ISSN: 2184-3228
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

quirements (Dobson and Sawyer, 2006). This facilitates the creation of a common terminology across different stakeholders (Riechert et al., 2007). They are applicable at TBox level (e.g., a meta-model describing requirements engineering concepts) and ABox level (e.g., project-specific requirements specifications) (Siegemund et al., 2011).

Various methods can be used to validate systems against requirements. *Testing* is a suitable validation technique that can be applied at this stage (ISO/IEC/IEEE, 2015). In model-driven requirements engineering, ontologies can assist in formalizing and analyzing requirements during the testing stage. Semantic validation enables requirements to be implemented correctly and to be interoperable (see, for example, the semantic validation of the correct use of specifications from industrial standards (Bareedu et al., 2023)). However, a *systematic* and *automated* semantic validation requires the generation, execution, and analysis of test cases. Despite recent advancements, there remains a gap in generating, executing, and analyzing test cases in a manner that is aligned with the formal semantics of ontologies. Addressing this issue would streamline the validation process, reduce human error, and enhance the consistency of requirement verification in complex systems. Thus, the primary research question addressed in this paper is:

How can systematic and automated semantic validation be achieved in the testing phase of model-driven requirements engineering using ontologies?

The paper is structured as follows: Section 2 introduces the *Shapes Constraint Language* (SHACL) as a possible solution for semantic-aware validation and explains its limitations as well as the research goal for this contribution. Section 3 provides an overview of related approaches to semantic requirements engineering and validation. In section 4 a workflow, an algorithm, and exemplary templates are introduced. Section 5 provides a prototypical implementation of the algorithm. In Section 6 an exemplary application for a robot concept is shown. Lastly, section 8 concludes the paper with a summary and outlook.

2 SEMANTIC-AWARE VALIDATION WITH SHACL

The *Shapes Constraint Language* (SHACL) (W3C, 2017) helps tackle several challenges in using ontologies for model-driven requirements validation. It provides a formal language for defining constraints, thereby enabling the validation ontology instances. This is achieved by defining SHACL shapes that represent the expected structure of the ontology, which is automatically validated. It is designed to be expressive while defining complex constraints and conditions (e.g., cardinality constraints). In contrast to *Object Constraint Language* (OCL) (OMG, 2014), which is designed for *Unified Modeling Language* (UML) models, SHACL can be used for ontologies. As a result, it integrates with semantic reasoning and is specifically tailored for use in semantic web applications. SHACL is applicable in iterative development stages, where requirements may change and evolve, allowing for early detection of requirement violations by concepts.

Although SHACL provides a suitable language for ontology validation, it lacks a systematic approach for applying it to semantic requirements validation in model-driven requirements engineering. Thus, the objective of this contribution is to provide a concept, consisting of a workflow, algorithm, and templates for model-driven and semantic-aware requirements validation by using automatically generated SHACL shapes. Achieving systematic and automated validation would lead to greater consistency in engineering, saving time and reducing errors caused by iteration loops in testing. This leads to higher quality in the requirements models and better communication among stakeholders due to a common understanding of requirements. The templates can be used across different projects thus promoting standardization and reusability. Lastly, the concept enables traceability from modeling to testing requirements.

3 RELATED WORK

Semantic web technologies such as OWL, SPARQL, and SWRL for modeling, querying, and reasoning have been applied in requirements engineering for several years (see, for example, (Mayank et al., 2004; Runde and Fay, 2011)). As emphasized in section 1, potential uses of ontologies include requirement representation and structuring as well as the application of domain knowledge (Dobson and Sawyer, 2006). Mappings between meta models play a crucial role in avoiding semantically inconsistent models (Saeki, 2010). Various ontologies have been created for semantic requirements engineering, for example, SWORE (Riechert et al., 2007), CORE (Jureta et al., 2009) or GORO (Bernabé et al., 2019). These models focus on the semantic representation of requirement concepts and represent a fundamental base for semantic validation.

Several systematic methods for creating semantic requirements models exist. In (Veleda and Cysneiros, 2019; Guizzardi et al., 2023), methods for ontology-based requirements elicitation are presented. Boilerplates can be used to integrate natural language components into ontology-based requirements engineering (as shown in (Antoniou and Bassiliades, 2024)). Using semantic web technologies such as SWRL, rules can be used to check completeness and consistency in requirements models (Siegemund et al., 2011). In (Banerjee and Sarkar, 2022) validation rules for validating consistency, unambiguity, and traceability in requirements specifications are proposed. However, the approach has predefined rules that can be adapted to other requirement meta-models only with considerable effort. Furthermore, the rules validate requirements in requirements specifications and not developed concepts against requirements.

In conclusion, the semantic validation of system concepts against requirements has not yet been fully addressed. While the analysis, specification, and elicitation of requirements are supported by ontologies, suitable methods for validation are lacking (Valaski et al., 2016). First, semantic validation requires definition and preferably automated generation of validation test cases. Furthermore, a common terminology must be established to automatically validate system concepts against requirements. Finally, a method for semantic validation must be independent of any specific requirements meta-model to be applicable across different domains.

4 CONCEPT

The concept presented in this contribution consists of a workflow for the systematic generation and application of test cases for validating requirements using SHACL, an algorithm for automatically generating SHACL shapes, and exemplary templates to support the algorithm.

4.1 Workflow

Figure 1 depicts the workflow for semantic-aware validation in model-driven requirements engineering. The steps have specific input artifacts (e.g., an ontology) and specific output artifacts (e.g., SHACL shapes).

4.1.1 Definition of Validation Use Cases

The first step of the workflow focuses on identifying validation requirements by formulating validation use

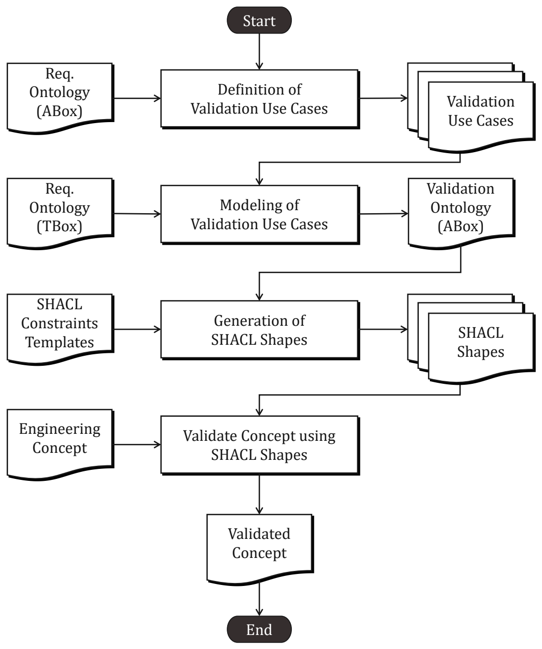
Start
Req. Ontology (ABox)
Definition of Validation Use Cases
Validation Use Cases
Req. Ontology (TBox)
Modeling of Validation Use Cases
Validation Ontology (ABox)
SHACL Constraints Templates
Generation of SHACL Shapes
SHACL Shapes
Engineering Concept
Validate Concept using SHACL Shapes
Validated Concept
End

Figure 1: Workflow for semantic-aware validation of engineering concepts against requirements in model-driven requirements engineering.

cases. The input for this step is a requirements ontology, specifically the ABox part, which encapsulates concrete instances and their relationships within the domain. Exemplary instances could be `Robot Arm Length` with a certain value, or `Maximum Pressure` required for a specific process step. This requirements ontology was created at the very beginning of an engineering process.

Analyzing this ontology identifies various representative scenarios and conditions under which system validation is necessary. These validation use cases will guide the subsequent phases of the validation process. The result of this step is a comprehensive set of validation use cases that capture the different facets of the validation requirements.

For improved documentation, validation use cases can be described using textual descriptions or semi-formal means of description such as SysML use case diagrams. In model-driven engineering, SysML facilitates consistent modeling of use cases down to requirements. Extending this conventional model-driven engineering with ontologies, as pursued in our approach, allows for semantic validation of requirements.

4.1.2 Modeling of Validation Use Cases

After identifying and defining validation use cases, the focus shifts to their formal representation through the creation of a validation ontology. This step uses the TBox part of the *same* requirement ontology as input, which specifies the terminology and conceptual framework. This ensures that requirements and validation use cases are described using the same vocabulary and semantics. Using the same TBox allows for automated analyses without the need for mapping between the two different TBoxes. Furthermore, this allows people from the same domain to model the requirements and subsequently test their fulfillment.

An exemplary validation use case could be a functionality test, for example, validating if a robot can reach objects within a certain range in a certain rotating angle. This is validated by checking the parameters `Robot Arm Length` and `Rotation Angle`.

This validation ontology, which contains specific instances of validation use cases, serves as a structured model that encapsulates relevant concepts, relationships, and constraints necessary for automatic validation.

4.1.3 Generation of SHACL Shapes

In the next step, the validation process continues with the generation of SHACL shapes that encode validation constraints derived from the validation ontology. The input for this step consists of pre-defined SHACL constraint templates that contain the constraints to be applied during the validation process. Utilizing the information encoded in the validation ontology, SHACL shapes are generated automatically. A generic template with placeholders is used to generate the SHACL shapes. Specific information is then retrieved from the validation ontology and inserted into the placeholders of the shapes. These shapes formally represent the validation criteria, facilitating an automated validation process. The output of this step consists of a comprehensive set of SHACL shapes.

4.1.4 Validate Concepts Using SHACL Shapes

In the final step of the workflow, engineering concepts are validated against the constraints encoded in the SHACL shapes. The input for this step consists of the engineering concept or artifact to be validated. Using the generated SHACL shapes, the automated validation of the engineering concept to the specified validation criteria can be performed. The output of this step is a validated concept that indicates the extent to which it conforms to predefined validation requirements. The output includes a validation report and results indicating which constraints are not fulfilled. Similar to test case validation in software engineering, the expected results can be checked. If the test fails, the engineering concept does not meet the desired criteria.

4.2 Algorithm

To generate SHACL shapes from a validation ontology, an algorithm is required to perform this step automatically. Algorithm 1 illustrates possible pseudocode for an automatic generation of SHACL shapes.

Algorithm 1: Generation of SHACL Shapes from Ontologies.

```
Input  : Validation Use Cases in Validation
         Ontology $V \in \mathcal{V}$
         SHACL Constraints Templates
$T \in \mathcal{T}$
Output: SHACL Shapes $S \in \mathcal{S}$
1  $M$ – **Create** or **Import** Mapping between
     OWL and SHACL concepts
2  **foreach** $V \in \mathcal{V}$ **do**
3      $t_V$ – Identify all Triples in $V$
4      **foreach** $t_V \in V$ **do**
5          **Select** Template $T$ for $t_V$ based on $M$
6          **Insert** $T$ into $S$
7          **Insert** Triple-specific information
             from $t_V$ into $T$
8      **end**
9      **Append** $S$ to $\mathcal{S}$
10 **end**
11 **return** $\mathcal{S}$
```

As shown in Figure 1, the step *Generation of SHACL Shapes* takes validation uses cases from the validation ontology $V \in \mathcal{V}$ and SHACL constraints templates $T \in \mathcal{T}$ as inputs, returning SHACL shapes $S \in \mathcal{S}$. First, mappings between OWL and SHACL must be created (**Line 1**). These mappings can also be predefined and imported. The mappings specify which concepts from OWL (and RDFS) can be mapped to SHACL concepts (e.g., `owl:Class` to `sh:targetClass`). For each validation use case, all triples (subject-predicate-object or subject-predicate-literal) t_V are identified by querying the ontology using SPARQL (**Lines 2-3**). The triples specify the constraints that the engineering concept must satisfy to pass the validation test. For each triple, suitable templates are selected and inserted into the SHACL shape. These templates are enriched with triple-specific information (**Lines 4-8**). The selection of a suitable template is based on the provided mappings. Assume there is a triple stating that a robot arm should cover an angle of 270 degrees around its axis. If the mapping includes relational operators with SHACL property pair constraints (e.g., sh:equals), suitable templates containing these constraints can be searched for. The triple-specific information (subject = `RobotArmLengthAngle`, predicate = `hasValue`, literal = `270°`) is inserted intro the template. Exemplary templates will be provided in the upcoming subsection. The shape is appended to the set of all SHACL shapes, which are returned at the end (**Line 9-11**).

Various approaches for the transformation into SHACL shapes exist, including (Pandit et al., 2018; Cimmino et al., 2020; Delva et al., 2021). In this contribution, we consider using the mapping from (Cimmino et al., 2020), because it automatically extracts SHACL shapes from ontologies and provides a suitable open-source implementation. The mappings between OWL and SHACL are predefined and embedded into Algorithm 1.

4.3 Templates

To generate SHACL shapes from OWL ontologies, templates for specific constraint types are used. These templates define a structure in which the triples are embedded and enhanced with additional restrictions.

An initial set of templates for SHACL constraints is developed. One example of a constraint is the *Cardinality Constraint*, which enforces that a certain element (subject) may be connected to more, fewer, or exactly a specified number of other elements or literals (objects) via an *ObjectProperty* or *DatatypeProperty* (predicate). The corresponding code is shown in Listing 1. The ABox-specific content of the validation ontology is integrated into the placeholders (represented by the brackets $< >$).

```
ex:CardinalityConstraintShape
  a sh:NodeShape ;
  sh:targetClass ex:<CLASS> ;
  sh:property [
    sh:path ex:<PROPERTY> ;
    sh:maxCount <INTEGER> ;
    sh:minCount <INTEGER> ;
    sh:severity sh:Violation ;
    ] .
```

Listing 1: Cardinality Constraint Template.

Another example is the *Negation Constraint*, which specifies conditions under which certain elements should not hold. Listing 2 provides the corresponding code. It specifies that a specific class should not have one or more properties of a particular type. If this occurs, the concept violates the requirement.

```
ex:NegationConstraintShape
  a sh:NodeShape ;
  sh:targetClass ex:<CLASS> ;
  sh:property [
    sh:not [
      sh:property [
        sh:path ex:<PROPERTY> ;
        sh:minCount 1 ;
      ] ;
    ] ;
  ] ;
  sh:severity sh:Violation .
```

Listing 2: Negation Constraint Template.

These templates enable the automated creation of SHACL shapes and allow them to be populated with content from the ontology.

5 IMPLEMENTATION

Figure 2 shows a simplified architecture of the prototype implementation that supports the generation of SHACL shapes and the execution of the algorithm. The components of the *SHACL Validator Tool* are illustrated in the architecture. The `Methods` component describes the methods that are used to import, manipulate, and execute SHACL shapes. A total of five methods are employed. First, the `importMappingTable()` method imports the `Mapping` table from an Excel file. The two ontologies, `Requirements Ontology` and `Validation Ontology`, are also imported. The `getAllTriples()` method is used to retrieve all relevant triples in the validation ontology using SPARQL. The two methods `selectSHACLTemplate` and `fillSHACLTemplate` are executed to select SHACL templates and parameterize them using content from the triples. The result is a SHACL shape, which can be executed using `executeSHACLShape`.

A mapping table is used to establish a relationship between OWL concepts and SHACL concepts. Based on this table, the triples can be examined to determine which SHACL templates are relevant and how the template parameterization is carried out. Currently, the selection of a suitable SHACL shape is performed manually, but it should be implemented automatically in the future development of the prototypical implementation. For this selection, the mapping table from (Cimmino et al., 2020) will be utilized. Table 1 shows an excerpt of the mapping table.

The prototypical implementation is realized in Python. In particular, the library *owlready2*[1] is used to import, query, and manipulate ontologies. Furthermore, *re*[2] is used to replace text snippets in the SHACL templates with the contents of the triples using regular expressions. Finally, *pyshacl*[3] can be used to execute the SHACL shapes directly in Python. This is not yet available in the current prototypical implementation but will be included in future work.

[1]https://pypi.org/project/owlready2/

[2]https://docs.python.org/3/library/re.html

[3]https://pypi.org/project/pyshacl/

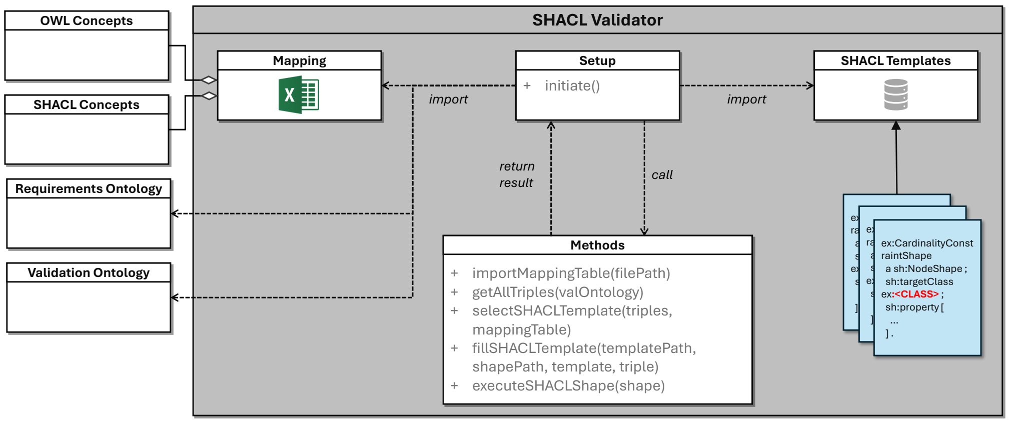
OWL Concepts
SHACL Concepts
Requirements Ontology
Validation Ontology
SHACL Validator
Mapping
Setup
+ initiate()
SHACL Templates
import
import
return result
call
Methods
+ importMappingTable(filePath)
+ getAllTriples(valOntology)
+ selectSHACLTemplate(triples, mappingTable)
+ fillSHACLTemplate(templatePath, shapePath, template, triple)
+ executeSHACLShape(shape)
ex:CardinalityConstraintShape
a sh:NodeShape ;
sh:targetClass
ex:<CLASS> ;
sh:property [
...
] .

Figure 2: Logical software architecture illustrating the components of the SHACL Validator Tool.

Table 1: Excerpt from mapping table showing exemplary mappings between OWL constructs and SHACL constructs (Cimmino et al., 2020).

OWL Construct Topic	OWL Construct	SHACL Construct	SHACL Construct Topic
Class definition	owl:Class	sh:NodeShape, sh:targetClass	Logical Constraint Constraints
Cardinality	owl:Class, rdfs:subClassOf, owl:Restriction, owl:cardinality, owl:onProperty	sh:NodeShape, sh:property, sh:PropertyShape, sh:maxCount, sh:minCount	Cardinality Constraints
Object Property definition	owl:ObjectProperty	sh:PropertyShape, sh:nodeKind, sh:BlankNodeOrIRI	Value Type Constraints
...	...	...	...

6 EXEMPLARY APPLICATION

To illustrate the method presented in this article, we use an exemplary application. A robotic arm is to be configured and developed to fulfill certain requirements. These requirements are modeled in a requirements ontology.

Consider the requirement `Req:Reaching-Capability` which defines the reaching capability of a robot. It should have a value of at least 1 meter (`Req:hasValue` $\geqslant 1m$). The validation use case defined for this could be *"Validate if arm length of developed robot concept is at minimum 1 meter"*.

The corresponding validation ontology instance `Val:Length` has a value of at least 1 meter (`Req:hasValue` $\geqslant 1m$). It uses the same ObjectProperty from the requirements ontology.

The generated SHACL shape is shown in Listing 3. Further details, such as the target class, can be added (e.g., `sh:targetClass ex:RobotArmLength`) if known.

```
ex:LengthShape ;
  a sh:NodeShape ;
  sh:property [
    sh:path ex:hasValue ;
    sh:datatype xsd:decimal ;
    sh:minInclusive 1.0 ;
    ] .
```

Listing 3: Exemplary SHACL shape - Arm Length.

Another requirement `Req:Mobility` addresses the mobility of a robot. A possible validation use case could be *"Validate if the robot concept has six degrees of freedom"*. The corresponding validation ontology instance `Val:DegreesOfFreedom` has a value of 6 (`Req:hasValue` = 6). The generated SHACL shape is shown in Listing 4.

```
ex:DegreesOfFreedomShape
    a sh:NodeShape ;
    sh:property [
        sh:path ex:hasValue ;
        sh:datatype xsd:integer ;
        sh:hasValue 8 ;
        ] .
```

Listing 4: Exemplary SHACL shape - Degrees of Freedom.

Furthermore, the robot should fulfill existing connectivity requirements to be integrated into the company's IT/OT system (`Req:Connectivity`). A possible validation use case could be *"Validate if the robot is capable of connecting via OPC UA and MQTT"*. The generated SHACL shape is shown in Listing 5.

```
ex:OPCUAandMQTTShape
    a sh:NodeShape ;
    sh:property [
        sh:path ex:hasValue ;
        sh:in ("OPCUA" "MQTT") ;
    ] .
```

Listing 5: Exemplary SHACL shape - Connectivity.

7 DISCUSSION

Using semantic web technologies such as SHACL for semantic-aware validation has several implications for model-driven requirements engineering. Formal representations improve interoperability, traceability, and consistency in model-driven requirements engineering. However, the presented approach requires an ontology-based modeling of requirements and engineering concepts. This is still challenging, as ontologies are still not fully adopted in an industrial context. This process can be supported by model-driven approaches that transform ontologies into discipline-specific data exchange formats like XML. As shown in (Köcher et al., 2022), a mapping language and mapping algorithm called *RDFex* can achieve this transformation. We will explore how RDFex can be integrated into the current implementation and used to generate and enrich SHACL shapes from ontologies.

In comparison to other approaches discussed in Section 3, we present a systematic workflow for semantic validation. While requirements engineering ontologies such as SWORE (Riechert et al., 2007), CORE (Jureta et al., 2009), and GORO (Bernabé et al., 2019) primarily focus on semantic representation, our approach integrates these ontologies into the validation phase of engineering. Several approaches, including (Veleda and Cysneiros, 2019) and (Guizzardi et al., 2023), address requirements elicitation but omit validation. In (Banerjee and Sarkar, 2022), predefined validation rules for consistency, clarity, and traceability in requirements specifications are proposed. However, these rules are difficult to adapt to other meta-models and do not validate developed concepts against the requirements, that our approach aims to achieve.

Two limitations of the current approach should be addressed in future work. First, while the method aims to support semantic validation of engineering concepts against requirements, several steps are still performed manually, making the process time-consuming. To fully realize the benefits of semantic-aware validation, greater automation is essential to improve efficiency and increase the return on modeling effort. Second, scalability remains a challenge for this approach. The method has so far only been validated on a small example, and its performance on larger, more complex use cases has not yet been tested. Without sufficient automation, scaling to larger use cases may result in significantly increased validation time, undermining the economic viability of the approach. To address these limitations, the reuse of templates and the automation of use case modeling will be necessary. These measures can help minimize modeling efforts and duplication across projects. While some steps of the method already incorporate templates and partial automation, we plan to extend these features in future work.

8 CONCLUSION

In this contribution, we presented a concept that addresses the research question by offering a solution for systematic and automated semantic validation during the testing phase of model-driven requirements engineering. The approach includes a structured workflow, algorithm, and templates specifically designed for semantic-aware validation using SHACL. The workflow comprises four consecutive steps, beginning with the definition of validation use cases, followed by their modeling as an ontology, the generation of SHACL shapes using SHACL constraints templates, and the validation of engineering concepts.

The presented approach is still a work in progress and currently lacks suitable evaluation. In future contributions, we intend to evaluate the concept using specific industrial examples from the process industry, such as oil, gas, and water separation, where semantic validation is crucial for compliance and safety. These industrial use cases can be used to test the scalability of the approach. Additionally, the prototypical implementation will be improved to support all automated steps, including the automatic generation of SHACL shapes from existing requirements and the integration of a user-friendly interface for a more streamlined validation workflow. This will allow users to effortlessly define validation use cases, model them as ontologies, and execute the validation process without manual intervention.

ACKNOWLEDGEMENTS

This research is funded by dtec.bw – Digitalization and Technology Research Center of the Bundeswehr. dtec.bw is funded by the European Union – NextGenerationEU.

REFERENCES

Antoniou, C. and Bassiliades, N. (2024). A tool for requirements engineering using ontologies and boilerplates. *Automated Software Engineering*, 31(5).

Baader, F., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., editors (2008). *The Description Logic Handbook*. Cambridge Univ. Press.

Banerjee, S. and Sarkar, A. (2022). Domain-specific requirements analysis framework: ontology-driven approach. *International Journal of Computers and Applications*, 44(1):23–47.

Bareedu, Y., Frühwirth, T., Niedermeier, C., Sabou, M., Steindl, G., Thuluva, A., Tsaneva, S., and Ozkaya, N. (2023). Deriving semantic validation rules from industrial standards: An opc ua study. *Semantic Web*, 15:1–38.

Bernabé, C. H., Silva Souza, V. E., de Almeida Falbo, R., Guizzardi, R. S. S., and Silva, C. (2019). Goro 2.0: Evolving an ontology for goal-oriented requirements engineering. In Guizzardi, G., Gailly, F., and Suzana Pitangueira Maciel, R., editors, *Advances in Conceptual Modeling*, volume 11787, pages 169–179. Springer International Publishing.

Cimmino, A., Fernández-Izquierdo, A., and García-Castro, R. (2020). Astrea: Automatic generation of shacl shapes from ontologies. In Harth, A., Kirrane, S., Ngonga Ngomo, A.-C., Paulheim, H., Rula, A., Gentile, A. L., Haase, P., and Cochez, M., editors, *The Semantic Web*, pages 497–513. Springer International Publishing.

Cleland-Huang, J., Zisman, A., and Gotel, O. (2012). *Software and Systems Traceability*. Springer.

Delva, T., Smedt, B., Oo, S., Assche, D., Lieber, S., and Dimou, A. (2021). Rml2shacl: Rdf generation taking shape. *Proceedings of the 11th Knowledge Capture Conference*, pages 153–160.

Dermeval, D., Vilela, J., Bittencourt, I. I., Castro, J., Isotani, S., Brito, P., and Silva, A. (2016). Applications of ontologies in requirements engineering: a systematic review of the literature. *Requirements Engineering*, 21(4):405–437.

Dobson, G. and Sawyer, P. (2006). Revisiting ontology-based requirements engineering in the age of the semantic web. 2006.

Guarino, N., Oberle, D., and Staab, S. (2009). What is an ontology? In Staab, S. and Studer, R., editors, *Handbook on ontologies*, International handbooks on information systems, pages 1–17. Springer.

Guizzardi, R., Amaral, G., Guizzardi, G., and Mylopoulos, J. (2023). An ontology-based approach to engineering ethicality requirements. *Software and Systems Modeling*, 22.

ISO/IEC/IEEE (2015). ISO/IEC/IEEE 15288. Systems and software engineering. System life cycle processes. (15288).

Jureta, I. J., Mylopoulos, J., and Faulkner, S. (2009). A core ontology for requirements. *Applied Ontology*, 4(3-4):169–244.

Köcher, A., Markaj, A., and Fay, A. (2022). Toward a generic mapping language for transformations between rdf and data interchange formats. In *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–4. IEEE.

Maalej, W. and Thurimella, A. K., editors (2013). *Managing requirements knowledge*. Springer.

Mayank, V., Kositsyna, N., and Austin, M. (2004). Requirements engineering and the semantic web, part ii. representaion, management, and validation of requirements and system-level architectures.

Moreira, A., Mussbacher, G., Araújo, J., and Sánchez, P. (2022). Theme section on model-driven requirements engineering. *Software and Systems Modeling*, 21.

OMG (2014). Object constraint language (ocl). https://www.omg.org/spec/ocl/2.4/pdf. accessed 2024-06-13.

Pandit, H. J., O'Sullivan, D., and Lewis, D. (2018). Using ontology design patterns to define shacl shapes. In *WOP@ ISWC*, pages 67–71.

Riechert, T., Lauenroth, K., Lehmann, J., and Auer, S. (2007). Towards semantic based requirements engineering. *Proceedings of the 7th International Conference on Knowledge Management*.

Runde, S. and Fay, A. (2011). Software support for building automation requirements engineering—an application of semantic web technologies in automation. *IEEE Transactions on Industrial Informatics*, 7(4):723–730.

Saeki, M. (2010). Semantic requirements engineering. In Nurcan, S., Salinesi, C., Souveyet, C., and Ralyté, J., editors, *Intentional Perspectives on Information Systems Engineering*. Springer Berlin Heidelberg.

Siegemund, K., Thomas, E. J., Zhao, Y., Pan, J., and Assmann, U. (2011). Towards ontology-driven requirements engineering. *The 10th International Semantic Web Conference*.

Uschold, M. (1998). Knowledge level modelling: concepts and terminology. *The Knowledge Engineering Review*, 13(1):5–29.

Valaski, J., Reinehr, S., and Malucelli, A. (2016). Which roles ontologies play on software requirements engineering? a systematic review.

VDI-Kompetenzfeld Informationstechnik (2009). VDI 5610-1. Knowledge management for engineering - Fundamentals, concepts, approach.

Veleda, R. and Cysneiros, L. M. (2019). Towards an ontology-based approach for eliciting possible solutions to non-functional requirements. In Giorgini, P. and Weber, B., editors, *Advanced Information Systems Engineering*, pages 145–161. Springer International Publishing.

W3C (2017). Shapes Constraint Language (SHACL). https://www.w3.org/TR/2017/REC-shacl-20170720/. Accessed 2024-03-19.