A Visual SHACL Shapes Editor Based On OntoPad

Natanael Arndt[1], André Valdestilhas[1], Gustavo Publio[1], Andrea Cimmino[2], Konrad Höffner[3] and Thomas Riechert[1,4]

[1]*AKSW Group, Institute for Applied Informatics (InfAI), Leipzig, Germany*
[2]*Ontology Engineering Group, Universidad Politécnica de Madrid, Spain*
[3]*Institute for Medical Informatics, Statistics and Epidemiology, Germany*
[4]*Leipzig University of Applied Sciences, Germany*

Abstract
On the Semantic Web, vocabularies and ontologies play a fundamental role to express the terminology and rules of certain domains. New technologies like SHACL provide the possibility to express data schemata specific to certain data sets, applications, and domains. However, the domain modeling process is collaborative and when using RDF, it requires technical knowledge. In this paper, we present a tool to support a two-step-process to model a terminology and a schema with a combined graphical RDF Schema editor and visual SHACL editor. This tool allows domain experts to create a terminology and schema without the need for a deep understanding of RDF Schema or SHACL.

Demo URL: https://ontopad.aksw.org/

1. Introduction

The W3C has promoted the Shapes Constraint Language (SHACL) [6] as recommendation to construct schematic blueprints as shapes of RDF data. These shapes can be used to validate data, to construct input forms to author new RDF data, and to express a domain model. In this way, they provide a pragmatic and flexible way to express how the individual terms in a vocabulary (classes and properties) relate to each other and how the instance data should look like. To model a domain means to understand and express its language and rules. To formally express this domain model with means of the Semantic Web is a technical process. The overall process of domain modeling is a collaborative process that requires the involvement of domain experts. Providing a graphical tool that allows to interact with SHACL shapes by using a visual diagram component would allow to make the RDF layer transparent to its users and provide a visual language to interact with the data model. To support the collaborative domain modeling process, visual editors could help to increase the involvement of domain

SEMANTiCS 2021, September 6–9, 2021, Amsterdam, NL
arndt@informatik.uni-leipzig.de (N. Arndt); valdestilhas@informatik.uni-leipzig.de (A. Valdestilhas); gustavo.publio@informatik.uni-leipzig.de (G. Publio); cimmino@fi.upm.es (A. Cimmino); konrad.hoeffner@uni-leipzig.de (K. Höffner); thomas.riechert@htwk-leipzig.de (T. Riechert)
https://aksw.org/NatanaelArndt (N. Arndt); https://aksw.org/AndreValdestilhas (A. Valdestilhas); https://aksw.org/GustavoPublio (G. Publio); https://aksw.org/KonradHoeffner (K. Höffner); https://aksw.org/ThomasRiechert (T. Riechert)
0000-0002-8130-8677 (N. Arndt); 0000-0002-0079-2533 (A. Valdestilhas); 0000-0002-3853-3588 (G. Publio); 0000-0002-1823-4484 (A. Cimmino); 0000-0001-7358-3217 (K. Höffner); 0000-0003-2053-5347 (T. Riechert)

experts into the process. Several systems to deal with SHACL shapes have been proposed (cf. section 2). So far, some of these tools provide a visualization of SHACL shapes and only one system has a prototypical interactive visual editor for SHACL shapes. Consequently, our far goal is to establish a joined distributed collaborative domain modeling process, based on the Quit Store [1], that involves all stakeholders. In this paper we present a tool, based on OntoPad, to visually support the domain modeling process. The domain modeling process is a two-step-process, in which we (1) define an RDF vocabulary and (2) visually compose SHACL shapes using the vocabulary. The visualization is inspired by the notation of UML class diagrams.

This paper is structured as follows. Section 2 provides an overview on the literature; Section 3 shows some implementation details and demonstrates the tool; finally, Section 4 presents the conclusions and a prospect to future work.

2. Related Work

There are many tools that contribute to creating a visual vocabulary and ontology editors. For the sake of brevity of this demo paper we have created a comparison in the Open Research Knowledge Graph (ORKG) [3] that represents the related work [2]. It is available at https://doi.org/10.48366/R113089.

3. Implementation and Demonstration

The OntoPad is implemented as a web application using JavaScript and the VueJS framework. The source code is available as FLOSS licensed under the terms of the GPL-3.0 on GitHub (https://github.com/AKSW/OntoPad). This allows to use state-of-the-art technology for the user interfaces and provide a responsive interaction independent of the users client platform. Figure 1 shows an overview of the domain modeling interface of the OntoPad.

To store the data, we send SPARQL Update requests to a SPARQL endpoint. In order to support the highly collaborative process of domain modeling, we use the Quit Store [1] as SPARQL endpoint. The Quit Store tracks all changes performed with SPARQL Update requests in a Git repository and allows to synchronize the repository with remote collaborators. This allows each collaborator to setup their own independent workspace while still participating in a common distributed collaborative workspace.

3.1. Step 1: Definition of the Terminology

In the first step, the terminology of a domain is defined. For this purpose the OntoPad provides a terminology component as can be seen in the left column of fig. 1. In the top it lists all explicit instances of rdfs:Class, as well as inferred instances of rdfs:Class by considering objects of rdf:type-triples and instances of owl:Class, that are found in the current graph. By clicking on the plus-button a new class term can be added. The interface to add a new class is shown in fig. 2. The user can specify the IRI as well as a label using the property rdfs:label and a comment using rdfs:comment. To edit a class, the user can select the respective term and switch to the edit tab. The edit tab provides a triple view on the class that allows to change the existing

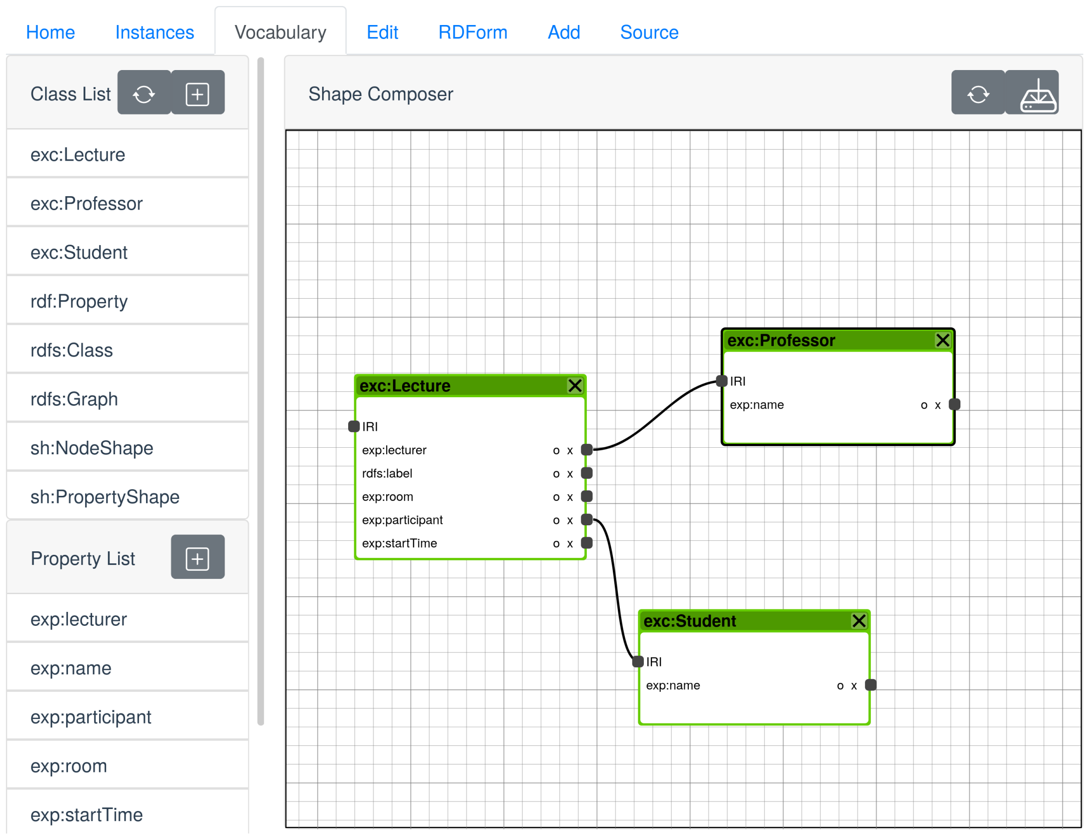
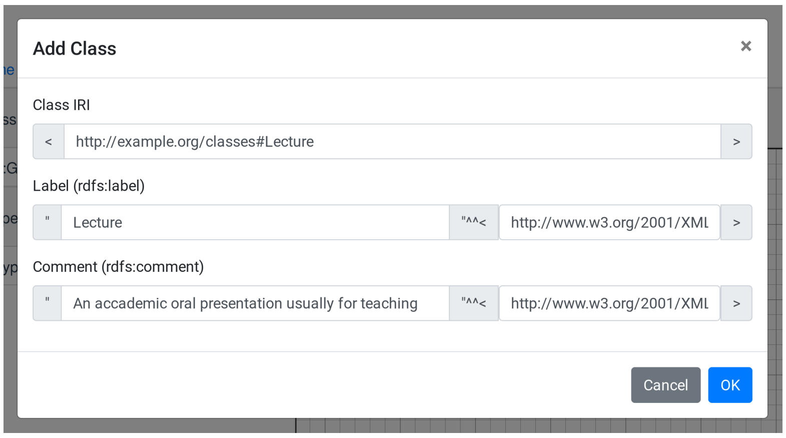


Figure 1: User interface of the visual representation of SHACL.

statements and also to add new statements. Additionally a source-tab is available to edit the resource using the Turtle syntax. In the lower part of the terminology component, properties can be defined and edited as instances of rdf:Property in the same way as for class terms. The term creation process is kept at a bare minimum to provide a very simple user interface that does not distract the domain experts from the discursive process.

3.2. Step 2: Definition of an Application

In the right part of the domain modeling interface, as depicted in fig. 1, the visual diagram component is provided. The graphical notation of the diagram is inspired by the notation of UML class diagrams as used in software engineering. The modeling tool allows the user to create new shapes by dragging an rdfs:Class from the left hand side and dropping it in the right hand shape composer. This action automatically creates a new instance of sh:NodeShape with the class specified as sh:targetClass. In the same way the user can drag an rdf:Property from the left hand list of terms and drop it inside a node shape. This automatically creates a new instance of sh:PropertyShape with the property specified as sh:path. The sh:PropertyShape is attached to the sh:NodeShape with the sh:property property. To specify a shape as constraint


Add Class

Class IRI

< http://example.org/classes#Lecture >

Label (rdfs:label)

" Lecture "^^< http://www.w3.org/2001/XML >

Comment (rdfs:comment)

" An accademic oral presentation usually for teaching "^^< http://www.w3.org/2001/XML >

Figure 2: An example of add a class and property.

for the value nodes of a property the user can click the square behind the property and draw a line to the IRI input port of the respective node shape. Also the sh:PropertyShape and sh:NodeShape resources can be viewed and edited with the triple editor or with the source-tab. By clicking the save button in the top right corner, the created node shapes and property shapes are stored to the underlying Quit Store. When opening the view, the diagram is initialized with already existing instances of sh:NodeShape and their linked sh:PropertyShape instances.

4. Conclusion and Future Work

We presented to our knowledge the first system that allows the user to create and edit an RDF terminology in conjunction with a visual diagram based editor for SHACL shapes. The OntoPad supports the two step vocabulary creation process: it allows to create and edit an RDF terminology in a graphical user interface and it provides an editor based on a visual graph diagram of SHACL Shapes. Our system is complemented by an editor for the RDF serialization and a graphical user interface to edit triples. OntoPad is built on top of Quit Store which provides a versioned RDF triple store, allowing the user to keep track of the creation process, provides the possibility for distributed collaboration, and tracks the relevant provenance information.

To actually make the vision of an interlinked vocabulary space in the Semantic Web true, we want to extend the system to also import existing vocabularies from vocabulary repositories like Linked Open Vocabularies [9] or DBpedia Archivo [5]. This should allow to foster the reuse of existing terminology and provide a mix-and-match environment to construct SHACL shapes for specific use cases and applications. As future work, we will further implement more of the SHACL constraints, like cardinality, datatypes, and string patterns, into the editor to support use cases like the creation of input forms with RDForm (https://github.com/simeonackermann/RDForm/). Even though the underlying Quit Store provides many unparalleled features, we want to untangle the dependency on this specific triple store to be able to use the OntoPad in combination with any endpoint that provides the standard SPARQL 1.1 Query and Update interface. To support the full linked data life-cycle the system shall also be tested in combination with and integrated with systems like ASTREA [4] in order to enable the automatic generation of SHACL shapes and test an ontology using formally pre-defined guidelines or custom SHACL tests with SHARK [8] or RDFUnit [7].

Acknowledgements

This work was partly supported by the Bundesanstalt für Materialforschung und -prüfung (BAM) and by a grant from the German Federal Ministry for Economic Affairs and Energy (BMWi) for the PlatonaM project (01MT19005A).

References

[1] Natanael Arndt et al. "Decentralized Collaborative Knowledge Management using Git". In: *Journal of Web Semantics* (2018). DOI: 10.1016/j.websem.2018.08.002.

[2] Natanael Arndt et al. *Related Work for the paper "A Visual SHACL Shapes Editor Based On OntoPad"*. en. 2021. DOI: 10.48366/R113089. URL: https://www.orkg.org/orkg/comparison/R113089.

[3] S. Auer et al. "Improving Access to Scientific Literature with Knowledge Graphs". In: *Bibliothek Forschung und Praxis* 44 (2020), pp. 516–529. DOI: 10.1515/bfp-2020-2042.

[4] Andrea Cimmino, Alba Fernández-Izquierdo, and Raúl García-Castro. "Astrea: Automatic Generation of SHACL Shapes from Ontologies". In: *European Semantic Web Conference*. Springer. 2020, pp. 497–513.

[5] Johannes Frey et al. "DBpedia Archivo - A Web-Scale Interface for Ontology Archiving under Consumer-oriented Aspects". In: *SEMANTiCS 2020*. 2020.

[6] Holger Knublauch and Dimitris Kontokostas. *Shapes Constraint Language (SHACL)*. Recommendation. W3C, July 2017. URL: https://www.w3.org/TR/2017/REC-shacl-20170720/.

[7] Dimitris Kontokostas et al. "Test-driven Evaluation of Linked Data Quality". In: *WWW '14*. Seoul, Korea, 2014. DOI: 10.1145/2566486.2568002.

[8] Gustavo Correa Publio. "SHARK: A Test-Driven Framework for Design and Evolution of Ontologies". In: *European Semantic Web Conference*. Springer. 2018, pp. 314–324.

[9] Pierre-Yves Vandenbussche et al. "Linked Open Vocabularies (LOV): A gateway to reusable semantic vocabularies on the Web". In: *Semantic Web Journal* 8.3 (2017). DOI: 10.3233/SW-160213.