OntoPad: A Visual Editor for RDF Vocabularies and SHACL Shapes

Natanael Arndt[1][0000−0002−8130−8677], André Valdestilhas[1][0000−0002−0079−2533], Gustavo Publio[1][0000−0002−3853−3588], Andrea Cimmino[3][0000−0002−1823−4484], and Thomas Riechert[1,2]

[1] AKSW Group, Institute for Applied Informatics (InfAI), Leipzig, Germany
{arndt,valdestilhas,gustavo.publio}@informatik.uni-leipzig.de
[2] Leipzig University of Applied Sciences, Germany
thomas.riechert@htwk-leipzig.de
[3] Ontology Engineering Group, Universidad Politécnica de Madrid, Spain
cimmino@fi.upm.es

Abstract. Nowadays, numerous RDF editors exist that allow visualising and editing data. Nevertheless, new technologies like SHACL or the high need for collaboration demand new requirements for those editors to meet. Specifically, allowing users to collaboratively define an RDF terminology and construct SHACL shapes to serve as templates for input forms and to validate instance data. In this paper, an editor for RDF vocabularies and for SHACL shapes based on a visual diagram editor is presented. The editor implements a two step approach for the domain modelling that supports the creation of a reusable RDF terminology in combination with use case specific shape constraints.

Permanent URL: `https://ontopad.aksw.org/`

1 Introduction

The Resource Description Framework (RDF) [9] allows to encode arbitrary data in a graph data model. To exchange the data, several textual serialization formats exist that store the graph as a set of statements (i.e. N-Triples, N-Quads) or in a tree or tree-like structure (RDF/XML, Turtle, JSON-LD). Some experienced users can perform very efficient work with this textual representation of the graph, while for other users and different tasks this representation is not suitable as user interface because it fails to convey the models' graph structure. This has driven the design of various graphical tools to assist users in exploring RDF data. There are tools to visualize RDF graphs [17, 10]. For general graph structures (but not specifically for RDF) a possibility exists to visually explore and manipulate the graph [5, 26], and also systems exist to explore RDF datasets as hypermedia in the browser [8, 2]. The bottom line of such tools is to allow users to visualise their data, i.e., the relationship between instances, and their properties.

On the Semantic Web Layer Cake[4] the RDF serves as common *data interchange* and *representation*. On top of the RDF, vocabularies, ontologies, and rules provide a framework to express and interpret the encoded data using RDF Schema (RDFS) [6] and OWL [30] as well as Rules using RIF [18]. This ontological or schema knowledge is also referred to as the terminological component of a knowledge base and is called *TBox*. In parallel to the TBox, the assertional knowledge is called the *ABox*. Due to the flexibility of the RDF it is possible to encode both, the TBox and the ABox in the same document. In RDF resp. RDFS the TBox consists of classes (instances of `rdfs:Class`) and properties (instances of `rdf:Property`).

Recently, the W3C has promoted the recommendation Shapes Constraint Language (SHACL) [19] to construct schematic blueprints as shapes of RDF data. These shapes can be used to validate data as well as to construct input forms for new RDF data. 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 instances data should look like. Although new systems dealing with SHACL have been presented [7, 11, 22, 21], to our knowledge no visual editor for SHACL exists. As a result, expressing data in RDF, developing its shapes, and the validation must be done separately. On the other hand, the existing RDF editors are meant to be single-user despite the fact that nowadays most of the work is achieved collaboratively, and through a versioning system.

1.1 Contribution

In this paper, we present the overall process we are following to model a domain in a two step process in order to create valid domain data. To underpin this process we present the OntoPad as a tool to define an RDF vocabulary and visually compose SHACL shapes using the vocabulary. The visualization is inspired by the notation of UML class diagrams. Since the process of domain modelling is highly involved with the overall process of working with RDF data, the OntoPad in its current version provides rudimentary possibilities to edit general RDF data. To support the collaborative domain modelling process the OntoPad is devised to work on top of the Quit Store [1]. The Quit Store is a triple store providing a standard SPARQL 1.1 Query and Update interface that tracks all changes to the data in a Git repository and allows to synchronize with remote collaborators.

The rest of this paper is structured as follows. Section 1.2 motivates the problem we are facing and provides some basic requirements to a visual SHACL editor; Section 2 reports an analysis of the proposals from the literature; Section 3 describes the process of domain modeling and vocabulary engineering that is implemented in our tool; Section 4 provides some implementation details and demonstrates the OntoPad; finally, Section 5 presents our findings and conclusions.

[4] A collection of several Semantic Web layer cake diagrams: `https://natanael.arndt.xyz/notes/semantic-web-layer-cake`

1.2 Motivation

In various use cases it is important that people get the opportunity to create vocabularies to describe their RDF data. This can be to describe the data contained in a data set or data source but also to model a specific domain to express the common understanding of domain experts. Currently, domain experts are not able to model the terms and relations of their domain on their own. They are sometimes scared by the RDF data model, the RDF serialization formats, or might get confused by the exact usage of the SHACL classes and properties (e.g. `sh:NodeShape`, `sh:targetClass`, `sh:PropertyShape`, `sh:path`). While the process of modelling a domain is a highly collaborative process that involves discussion of the domain experts they currently need to be supported by an additional data scientist resp. Semantic Web expert. The result expressed in SHACL is then also hard to validate by the 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 define the target area of this paper we define a set of dimensions. We recognize that there are several ways to author and maintain RDF data in general (interactive dimension): (I1) in a serialization format, (I2) with a graphical user interface, and (I3) by changing a visual graph diagram. On the expressiveness dimension[5] there are tools to work with (E1) plain RDF data (i.e. triples and resources), (E2) with an RDF terminology (classes and properties), (E3.1) with ontologies using ontology languages like OWL, and (E3.2) with RDF shapes using techniques like SHACL or ShEx. The expressiveness dimension is not linear. You can of course work with an RDF Schema, OWL, or SHACL shapes by just using plain RDF data. Further, also OWL ontologies overlap with RDFS (RDFS is even a Fragment of OWL Full, but not of OWL DL and OWL Lite). To build SHACL Shapes in turn you already need an RDF terminology to build on. On a third dimension, there are tools that allow to (R) read, explore, and visualize the data while other tools (RW) allow the change and edit the data. Our aim is at a tool that provides two steps. First, it should provide a possibility to create and edit an RDF terminology in a graphical user interface (I2,E2,RW). Second, it should provide an editor based on a visual graph diagram of SHACL Shapes (I3,E3.2,RW).

2 Related Work

Looking at the related work we found many tools that contribute to solving the problem targeted in this paper. Some systems even provide a visual way to interact with OWL ontologies. Nevertheless, we could find no system that fulfills our requirements for a visual SHACL editor. In the following we show the most relevant work related to our approach.

[5] Please note that the numbers should just denote categories and not stand for a level of expressiveness.

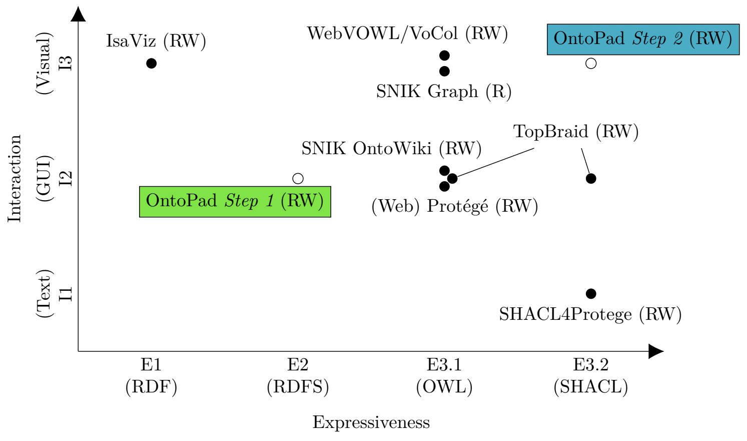


Fig. 1. The categorization of the related work systems[5].

WebVOWL [32] is a web application to visualize ontologies which implements the Visual Notation for OWL Ontologies (VOWL) [23]. The approach is entirely based on open web standards and provides a web editor that is device independent, allowing the modelling of ontologies visually. The user interface considers different modes of operation, including mouse and touch interactions. We categorize the system as (I3,E3.1,RW). While the system is very rich in its visual interaction and the possibility to work with OWL ontologies it does not include support for SHACL. The visual representation framework of WebVOWL was also published as a separate framework called **GizMO**[31]. The GizMO framework allows to further customize the visual representation.

The **VoCol** [15] system is an integrated environment that supports the development of vocabularies using Version Control Systems, which is based on a fundamental model of vocabulary development, consisting of the three core activities modeling, population, and testing. The implementation uses a loose coupling of validation, querying, analytics, visualization, and documentation generation components on top of a standard Git repository. All components, including the version-controlled repository, can be configured and replaced. With its integration of WebVOWL it can also be categorized as (I3,E3.1,RW) while it provides no specific support for SHACL shapes.

The **Protégé** [24] and its web version **Web Protégé** [28] is a platform that provides a suite of tools to construct domain models and knowledge-based applications with ontologies. The desktop version offers plugins in order to extend its functionalities. The SHACL4Protege plugin [11] provides a text-based editor for SHACL constraints and is also capable of validating data against the shapes inside the Protégé platform. The plugin does not offer a visual interface, so explicitly writing SHACL in an RDF serialization through the editor is necessary. We categorize as (I2,E3.1,RW) and with the SHACL4Protege plugin as (I1,E3.2,RW).

The **TopBraid** system by TopQuadrant[6] has a SHACL library included that was also release as open source[7]. It provides a command-line interface where the user can specify the data file and the SHACL file to be checked against each other. As the TopBraid tools are not Open Source we were not able to further analyse the interaction with SHACL shapes. As of our understanding the tool has a well integrated support for SHACL but no visual diagram based SHACL modeling interface[8]. We categorize thas system as (I2,E3.1/E3.2,RW).

The **SNIK Graph**[16] provides a tool to visualize medical informatics ontologies, combines knowledge from different literature sources dealing with the management of hospital information systems (HIS). Concepts and relations were extracted from literature, modeled as an ontology, and visualized as a graph on a website. It has the capability to visualize very large ontologies (currently 2641 nodes and 4904 edges). The approach is able to visualize ontologies from SPARQL endpoints. It does not support SHACL shapes. With the underlying OntoWiki [13, 14] it is able to create and edit OWL and RDF Schema terms. It can be categorized as (I3,E3.1,R) and (I2,E3.1,RW).

The **IsaViz** [25] is a visual environment for browsing and authoring RDF models, represented as directed graphs. Resources and literals are the nodes of the graph (ellipses and rectangles respectively), with properties represented as the edges linking these nodes. Since version 2.0, IsaViz supports GSS (Graph Stylesheets), a stylesheet language derived from CSS and SVG for styling models represented as node-link diagrams. The difference here is also on the support for SHACL definitions on the RDF-Schema terms. We categorize as (I3,E1,RW).

In fig. 1 the systems of the related work are visualized according to their categorization.

3 The Process of Authoring and Validating Data

The vocabulary engineering and SHACL shape creation is embedded into the overall life-cycle of linked data. The general life-cycle that is performed when working with linked data was researched in the LOD2 project and is captured in the linked data life-cycle [4]. For our problem, especially the steps *Quality Analysis* that includes data validation, the data creation steps *Extraction* and *Manual revision/Authoring*, and also the *Evolution/Repair* are relevant. These life-cycle steps are depicted in the right hand and bottom part of fig. 2. The result of the data management process shall be *valid domain data*. To support this endeavor, in preparation a collaborative domain modelling process needs to be performed.

With the OntoPad approach we focus on the collaborative domain modelling process. The management of the vocabulary and schema, which involve the *definition of the terminology* and the *definition of the application* as depicted in

[6] The TopQuadrant web page: `https://www.topquadrant.com/`

[7] The TopQuadrant SHACL API: `https://github.com/TopQuadrant/shacl`

[8] TopQuadrant SHACL Tutorial: `https://www.topquadrant.com/technology/shacl/tutorial/`

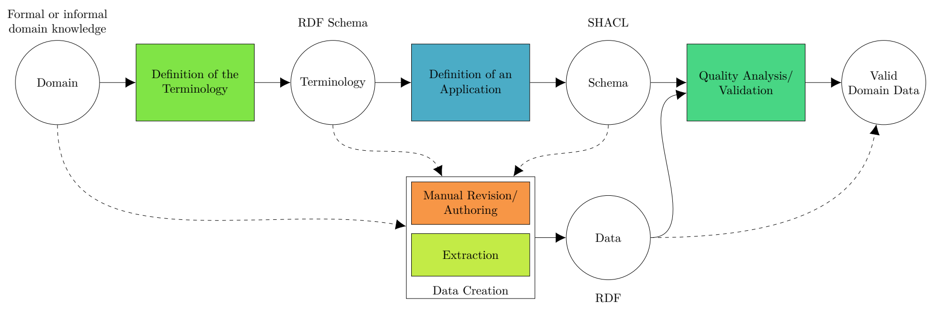

Formal or informal domain knowledge
Domain
Definition of the Terminology
RDF Schema
Terminology
Definition of an Application
SHACL
Schema
Quality Analysis/ Validation
Valid Domain Data
Manual Revision/ Authoring
Extraction
Data Creation
Data
RDF

Fig. 2. Generating Valid Domain Data

fig. 2. The process is divided into two steps, in the first step the *terminology* if the domain is captured and defined, in the second step the terms are put into relation in order to model the intended use case or application. This process can be seen in parallel to the parts that need to be learned with a natural language. At first it is necessary to understand important terms and learn the vocabulary of the language resp. domain, next one needs to learn how to combine the terms in a way to make sense which is expressed in the grammar resp. schema.

The knowledge about what data means and how it is expressed is defined within a specific domain of application. The domain (meta) knowledge is encoded into the terminology as instances of `rdfs:Class` and `rdfs:Property` and the schema expressed as SHACL shape. Data is then either manually created or extracted from unstructured sources, and validated according to this schema knowledge. The resulting output is valid domain data, which not necessarily indicates that the data is true, but that it conforms to its schema constraints.

To support this overall process of managing domain knowledge the proper handing of the vocabulary engineering process is of paramount importance. We argue that within this process the separation of the two steps of the *definition of the terminology* and the *definition of the application* need to be organized in to separated steps. The terminology needs to describe the concepts of a domain in a general and reusable way in order to be able to reuse the terms across multiple use cases and applications within a domain. On the other hand, the definition of an application needs to provide the possibility to combine the terms and define strict restrictions to allow well guided data creation processes and a pragmatic data management.

4 Implementation and Demonstration

The OntoPad is implemented as browser 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[9]. This allows to use state of the art technol-

[9] The source code of OntoPad: `https://github.com/AKSW/OntoPad`

ogy for user interfaces and provide a responsive interaction independent of the users client platform. Figure 3 shows an overview of the vocabulary engineering interface of the OntoPad. To store the data we of course use the RDF and send SPARQL Update requests to the back-end SPARQL endpoint. In order to support the highly collaborative process of the vocabulary engineering we use the Quit Store as SPARQL endpoint [1]. 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. Both the OntoPad as well as the Quit Store are available as virtual container image using the docker technology[10], easing its use and deployment in data infrastructures [3].

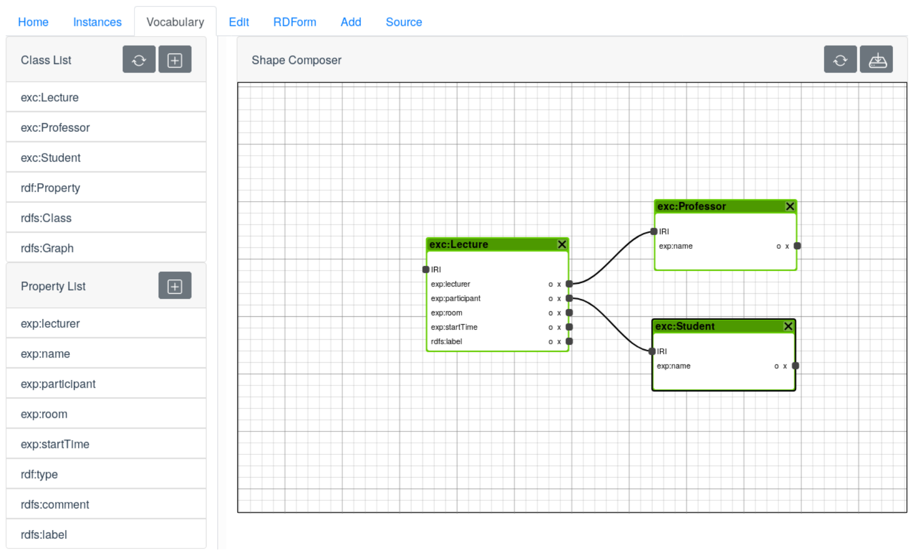
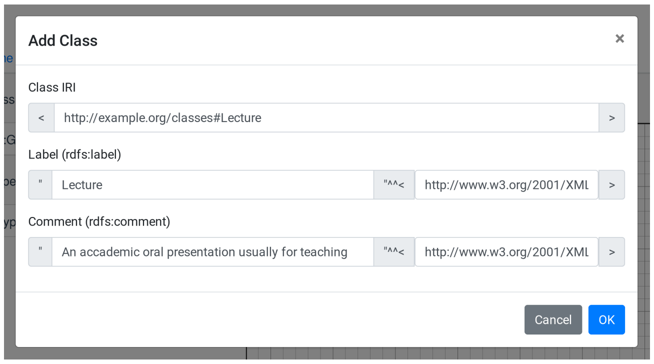


Fig. 3. User interface of the visual representation of SHACL

4.1 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. 3. In the top it lists all instances of `rdfs:Class`, as well as inferred instances by considering object of `rdf:type` and instances of `owl:Class`, that are found in the current graph. By clicking on plus-button a new class-term can be added.

[10] The OntoPad docker image: `https://hub.docker.com/r/aksw/ontopad`, the Quit Store docker image: `https://hub.docker.com/r/aksw/quitstore`



Add Class ×

Class IRI

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

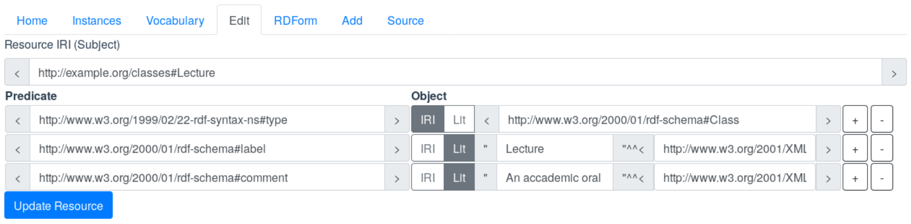
Label (rdfs:label)

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

Comment (rdfs:comment)

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

Fig. 4. An example of add a class and property.



Resource IRI (Subject)

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

Predicate	Object
http://www.w3.org/1999/02/22-rdf-syntax-ns#type	IRI: http://www.w3.org/2000/01/rdf-schema#Class
http://www.w3.org/2000/01/rdf-schema#label	Lit: " Lecture "^^< http://www.w3.org/2001/XML
http://www.w3.org/2000/01/rdf-schema#comment	Lit: " An accademic oral "^^< http://www.w3.org/2001/XML

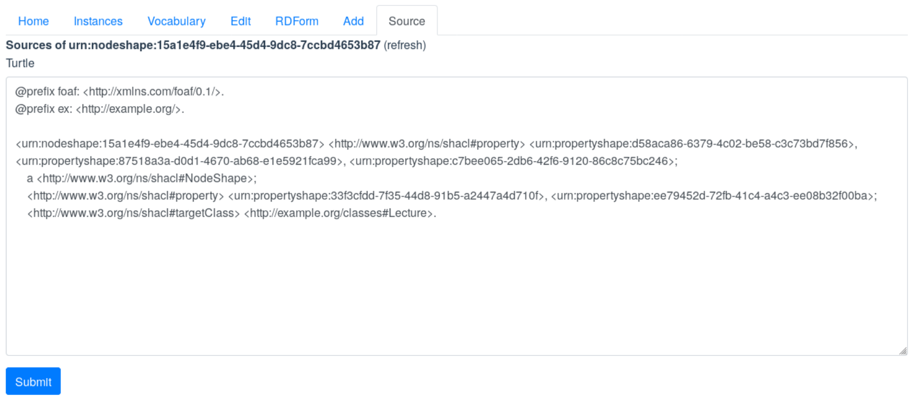
Fig. 5. An example of editing a resource.

The interface to add a new class is shown in fig. 4. The user can specify the IRI as well as a label using the property `rdfs:label` and a comment `rdfs:comment`. To edit a class the user can select the respective term and switch to the edit-tab. The edit-tab, as depicted in fig. 5 provides a triple view on the class that allows to change the existing statements and also to add new statements. Additionally a source-tab is available to edit the resource using the Turtle syntax as depicted in fig. 6. In the lower part of the terminology component properties can be defined and edited as instances of `rdf:Property` in the same 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.

4.2 Step 2: Definition of an Application

In the right part of the vocabulary engineering interface, as depicted in fig. 3, the visual diagram component is provided. The graphical notation of the diagram is inspired by the notation of UML class diagrams as they are using in the software engineering. The OntoPad 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 will automatically create 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 will automatically create a new instance of `sh:PropertyShape` with the property specified as `sh:path`. The `sh:ProeprtyShape` is attached to the `sh:NodeShape` with the `sh:property` property. To specify a shape as constraint 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 for the shapes it is possible to view the resources in the triple editor as depicted in fig. 5 or use the source-tab to with with the Turtle syntax fig. 6. By clicking the save-button in the top right corner the created node shapes and property shapes are stored to the underlying Quit Store. The diagram loads existing instances of `sh:NodeShape` with their linked `sh:PropertyShape` instances and visualizes them if present in the store already.



Sources of urn:nodeshape:15a1e4f9-ebe4-45d4-9dc8-7ccbd4653b87 (refresh)

Turtle

```
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
@prefix ex: <http://example.org/>.

<urn:nodeshape:15a1e4f9-ebe4-45d4-9dc8-7ccbd4653b87> <http://www.w3.org/ns/shacl#property> <urn:propertyshape:d58aca86-6379-4c02-be58-c3c73bd7f856>,
<urn:propertyshape:87518a3a-d0d1-4670-ab68-e1e5921fca99>, <urn:propertyshape:c7bee065-2db6-42f6-9120-86c8c75bc246>;
  a <http://www.w3.org/ns/shacl#NodeShape>;
  <http://www.w3.org/ns/shacl#property> <urn:propertyshape:33f3cfdd-7f35-44d8-91b5-a2447a4d710f>, <urn:propertyshape:ee79452d-72fb-41c4-a4c3-ee08b32f00ba>;
  <http://www.w3.org/ns/shacl#targetClass> <http://example.org/classes#Lecture>.
```

Fig. 6. An example output of a SHACL shape source.

5 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 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 (category (I2,E2,RW) section 1.2) and it provides an editor based on a visual graph diagram of SHACL Shapes (category (I3,E3.2,RW)). Our system is complemented by an editor for the RDF serialization and a graphical user interface for the triple view (categories (I1,E1,RW) and (I2,E1,RW)). The OntoPad is built on top of the Quit Store with provides a versioned RDF triple store, allowing the user to keep track of the creation process, provide 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 LOV or DBpedia Archivo [29, 12]. This should allow to foster the reuse of existing terminology and provide a mix'n'match environment to construct SHACL shapes for specific use cases and applications. As future works we will further implement more of the SHACL constraints into the editor to fully support use cases like the creation of input forms with RDForm[11]. Even though the underlying Quit Store provides many unparalleled features we want to untangle the dependency on this specific triple store in to be able to used 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[7] 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[27] or RDFUnit [20].

Acknowledgements

This work was partly supported 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`. URL: `https://arxiv.org/pdf/1805.03721`.

[2] Natanael Arndt et al. “Jekyll RDF: Template-Based Linked Data Publication with Minimized Effort and Maximum Scalability”. In: *19th International Conference on Web Engineering (ICWE 2019)*. Vol. 11496. LNCS. Daejeon, Korea, June 2019. DOI: `10.1007/978-3-030-19274-7_24`.

[3] Natanael Arndt et al. “Knowledge Base Shipping to the Linked Open Data Cloud”. In: *SEMANTICS '15: Proceedings of the 11th International Conference on Semantic Systems*. Ed. by Sebastian Hellmann, Josiane Xavier Parreira, and Axel Polleres. Vienna, Austria, Sept. 2015, pp. 73–80. DOI: `10.1145/2814864.2814885`.

[4] Sören Auer et al. “Managing the life-cycle of Linked Data with the LOD2 Stack”. In: *ISWC 2012: The Semantic Web – ISWC 2012*. Ed. by Philippe Cudré-Mauroux et al. Vol. 7650. LNCS. Berlin, Heidelberg, Germany: Springer, Nov. 2012. ISBN: 978-3-642-35172-3. DOI: `10.1007/978-3-642-35173-0_1`.

[5] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. “Gephi: an open source software for exploring and manipulating networks”. In: *Third international AAAI conference on weblogs and social media*. 2009.

[11] The project page for RDForm: `https://github.com/simeonackermann/RDForm/`

[6] Dan Brickley and R.V. Guha. *RDF Schema 1.1*. Recommendation. W3C, Feb. 2014. URL: https://www.w3.org/TR/2014/REC-rdf-schema-20140225/.

[7] 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.

[8] Richard Cyganiak and Chris Bizer. "Pubby-a linked data frontend for sparql endpoints". In: *Url: http://wifo5-03. informatik. uni-mannheim. de/pubby/.(Acesso: 19-11-2014)* (2008).

[9] Richard Cyganiak, David Wood, and Markus Lanthaler. *RDF 1.1 Concepts and Abstract Syntax*. Recommendation. W3C, Feb. 2014. URL: https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.

[10] Leonidas Deligiannidis, Krys J Kochut, and Amit P Sheth. "RDF data exploration and visualization". In: *Proceedings of the ACM first workshop on CyberInfrastructure: information management in eScience*. 2007, pp. 39–46.

[11] Fajar J Ekaputra and Xiashuo Lin. "SHACL4P: SHACL constraints validation within Protégé ontology editor". In: *2016 International Conference on Data and Software Engineering (ICoDSE)*. IEEE. 2016, pp. 1–6.

[12] Johannes Frey et al. "DBpedia Archivo - A Web-Scale Interface for Ontology Archiving under Consumer-oriented Aspects". In: *Semantic Systems. The Power of AI and Knowledge Graphs*. Vol. 16. Lecture Notes in Computer Science. Springer, 2020. URL: https://svn.aksw.org/papers/2020/semantics_archivo/public.pdf.

[13] Philipp Frischmuth, Natanael Arndt, and Michael Martin. "OntoWiki 1.0: 10 Years of Development - What's New in OntoWiki". In: *Joint Proceedings of the Posters and Demos Track of the 12th International Conference on Semantic Systems - SEMANTiCS2016 and the 1st International Workshop on Semantic Change & Evolving Semantics (SuCCESS'16)*. Vol. 1695. CEUR Workshop Proceedings. Leipzig, Germany, Sept. 2016. URL: http://ceur-ws.org/Vol-1695/paper11.pdf.

[14] Philipp Frischmuth et al. "OntoWiki - An Authoring, Publication and Visualization Interface for the Data Web". In: *Semantic Web Journal* 6.3 (2015), pp. 215–240. ISSN: 1570-0844. DOI: 10.3233/SW-140145.

[15] Lavdim Halilaj et al. "VoCol: An Integrated Environment to Support Version-Controlled Vocabulary Development". In: *EKAW 2016: Knowledge Engineering and Knowledge Management*. Ed. by Eva Blomqvist et al. Vol. 10024. LNCS. Bologna, Italy: Springer, Nov. 2016, pp. 303–319. ISBN: 978-3-319-49003-8. DOI: 10.1007/978-3-319-49004-5_20.

[16] Konrad Höffner et al. "Technical Environment for Developing the SNIK Ontology of Information Management in Hospitals." In: *GMDS*. 2017, pp. 122–126.

[17] Karwan Jacksi, Subhi Zeebaree, and Nazife Dimililer. "Design and Implementation of LOD Explorer: A LOD Exploration and Visualization

Model". In: *Journal of Applied Science and Technology Trends* 1.2 (2020), pp. 31–39.

[18] Michael Kifer and Harold Boley. *RIF Overview.* Working Group Note. W3C, Feb. 2013. URL: https://www.w3.org/TR/2013/NOTE-rif-overview-20130205/.

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

[20] Dimitris Kontokostas et al. "Test-driven Evaluation of Linked Data Quality". In: *WWW '14: Proceedings of the 23rd international conference on World wide web.* Seoul, Korea: Association for Computing Machinery, 2014, pp. 747–758. ISBN: 978-1-4503-2744-2. DOI: 10.1145/2566486.2568002.

[21] Jose Emilio Labra Gayo, Daniel Fernández-Álvarez, and Herminio Garcıa-González. "RDFShape: An RDF playground based on Shapes". In: *Proceedings of ISWC.* 2018.

[22] Martin Leinberger et al. "Type Checking Program Code Using SHACL". In: *International Semantic Web Conference.* Springer. 2019, pp. 399–417.

[23] Steffen Lohmann et al. "Visualizing Ontologies with VOWL". In: *Semantic Web* 7.4 (2016), pp. 399–419. DOI: 10.3233/SW-150200. URL: http://dx.doi.org/10.3233/SW-150200.

[24] Natalya F Noy et al. "Creating semantic web contents with protege-2000". In: *IEEE intelligent systems* 16.2 (2001), pp. 60–71.

[25] Emmanuel Pietriga. "Isaviz, a visual environment for browsing and authoring rdf models". In: *Eleventh International World Wide Web Conference Developers Day, 2002.* 2002. URL: https://www.w3.org/2001/11/IsaViz/.

[26] Laura Po et al. *Linked Data Visualization: Techniques, Tools, and Big Data.* Morgan & Claypool Publishers, 2020.

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

[28] Tania Tudorache, Jennifer Vendetti, and Natalya Fridman Noy. "Web-Protege: A Lightweight OWL Ontology Editor for the Web." In: *OWLED.* Vol. 432. Citeseer. 2008, p. 2009.

[29] Pierre-Yves Vandenbussche et al. "Linked Open Vocabularies (LOV): A gateway to reusable semantic vocabularies on the Web". In: *Semantic Web* 8.3 (2017), pp. 437–452. DOI: 10.3233/SW-160213. URL: https://doi.org/10.3233/SW-160213.

[30] W3C OWL Working Group. *OWL 2 Web Ontology Language Document Overview.* Recommendation. W3C, Dec. 2012. URL: https://www.w3.org/TR/2012/REC-owl2-overview-20121211/.

[31] Vitalis Wiens, Steffen Lohmann, and Sören Auer. "GizMO–A Customizable Representation Model for Graph-Based Visualizations of Ontologies". In: *Proceedings of the 10th International Conference on Knowledge Capture.* 2019, pp. 163–170.

[32] Vitalis Wiens, Steffen Lohmann, and Sören Auer. "WebVOWL Editor: Device-Independent Visual Ontology Modeling". In: *Proceedings of the ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks*. Vol. 2180. CEUR Workshop Proceedings. CEUR-WS.org, 2018.