OntoRipple: Making Waves in the Knowledge Graph Lifecycle

Diego Conde-Herreros[a], Oscar Corcho[a], David Chaves-Fraga[b]

[a] *Universidad Politécnica de Madrid, Spain*
{diego.conde.herreros,oscar.corcho}@upm.es
[b] *CiTIUS, Universidade de Santiago de Compostela Spain*
david.chaves@usc.es

Abstract

Knowledge Graphs are commonly organised according to the structure of existing ontologies, which define the concepts, relations, and restrictions of the domain of the KG. There are ontology-dependent assets that guide how data from heterogeneous sources is integrated, transformed, validated, and exploited in the KG, such as mapping rules and validation constraints. As ontologies evolve over time, these changes must be consistently reflected in the dependent assets, ensuring that the resulting KG remains aligned with the updated ontology. While ontology evolution has been widely studied, the propagation of changes to dependent artifacts remains an open challenge, requiring manual effort that makes the process slow, error-prone, and costly. In this paper, we present OntoRipple, a set of algorithms integrated into a tool that automates the propagation of ontology changes into RML mappings and SHACL shapes to construct and validate Knowledge Graphs, ensuring consistency with the evolving ontology in a fully declarative workflow.

Keywords: Knowledge Graphs, Ontology Evolution, Mapping Rules, Impact Assessment, Validation Rules

Metadata

Nr.	Code metadata description	Metadata
C1	Current code version	v1.0
C2	Permanent link to code/repository used for this code version	https://github.com/oeg-upm/OntoRipple
C3	Permanent link to Reproducible Capsule	https://doi.org/10.5281/zenodo.17313224
C4	Legal Code License	Apache License, version 2.0.
C5	Code versioning system used	Git
C6	Software code languages, tools, and services used	python
C7	Compilation requirements, operating environments & dependencies	python 3.9 ramel-yaml 0.18.10, rdflib 7.1.1, coloredlog 15.0.1, yatter 2.0
C8	If available Link to developer documentation/manual	https://github.com/oeg-upm/OntoRipple?tab=readme-ov-file
C9	Support email for questions	diego.conde.herreros@upm.es

Table 1: Code metadata

1. Motivation and significance

An ontology is the specification of a shared conceptualization that describes structural and domain-specific knowledge for a variety of tasks, such as data integration, transformation and homogenization processes [1]. Ontologies are implemented using formal declarative languages such as RDF Schema [2], and OWL [3]. Knowledge graphs (KGs) may be created natively or from heterogeneous sources and formats (RDB, CSV, JSON, etc.) using a set of logical rules (or mappings) that establish the underlying relation between the classes and properties from an ontology and the data sources. These mapping rules can be described using languages such as R2RML [4], RML [5], ShExML [6], and SPARQL-Anything [7]. Additionally, the generated KG has to be validated against a set of constraints to ensure that the data is correctly generated. Validation rules can be described by languages such as SHACL [8] and ShEx [9]. Both RML and SHACL have been standardized at W3C and widely adopted within the semantic web community through real-world initiatives such as PPDS [10], ERA [11], ISS [12], and Graph-Massivzer [13].

Ontologies evolve over time due to changes in domain knowledge or conceptualization, which must be consistently propagated to dependent assets such as RML mappings and SHACL shapes to maintain KG consistency [14, 15]. The evolution of ontologies has been studied in previous works [14], such as defining change operations [16], ontologies to describe these changes [17, 18], and creating ontology engineering methodologies that consider ontology evolution [19, 20]. The semantic related artifacts, such as the mapping rules used to construct the graph, the shapes used for validating the graph, or the SPARQL [21] queries for exploiting the graph, may need to change accordingly. This task is still largely performed manually, making it a labor-intensive and costly process. This is mainly due to the lack of standard frameworks and supporting tools to manage the evolution of interrelated artifacts involved in Knowledge Graph construction and validation.

There have been theoretical studies of how ontology evolution impacts the RML mappings, such as [22], where mappings are updated so that they answer the competency questions for DL-Lite$_R$ ontologies. In the context of KG validation, despite the tools and resources focused on the automatic creation of SHACL shapes [23, 24, 25, 26] no explicit work has been focused on the evolution of SHACL with respect to the ontology.

This paper presents *OntoRipple* [27], a tool that propagates ontology changes to its related RML [5] mappings and SHACL [8] shapes. The tool propagates 25 distinct change operations from the OWL Change Ontology that cover changes on OWL classes, properties, class relations, property relations, and property characteristics to RML and SHACL. When an assumption is made regarding how a change should be propagated in RML mappings the modified triples are added to review mappings for the user to check. This allows knowledge graph engineers to construct and validate the updated KG with minimal labor.

Problems and Objectives: To address the problem of updating semantic artifacts when an ontology evolves in a KG-driven project.

Proposed approach: OntoRipple, a tool that queries over the change description in RDF, processes the operations in a specific order, and performs the corresponding changes over the semantic artifacts in a fully declarative approach.

Contributions: (i) the OntoRipple software (ii) the declarative SPARQL templates implementing the propagation logic, (iii) validation of the tool in a real-world use case [10].

The remainder of the article is structured as follows. Section 2 will provide a detailed description of the OntoRipple tool, its software architecture, and functionalities. Section 3 will provide an illustrative example of the usage of the tool in its original use case, the PPDS project. The novelty, contributions,

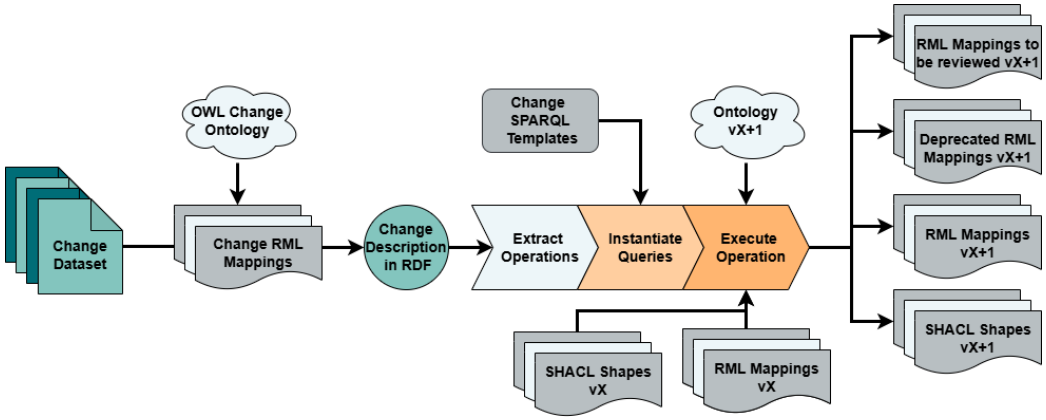

Figure 1: Diagram of the OntoRipple tool, its steps, inputs, and outputs.

and impact will be discussed in Section 4, and the conclusions, limitations, and future work will be reflected upon in Section 5.

2. Software description

2.1. Preliminaries: RML & SHACL

A short overview is provided to clarify which RML and SHACL elements are affected by ontology changes.

RML defines how data from heterogeneous sources (CSV, JSON, relational DBs, etc.) is transformed into RDF triples using different ways of generating RDF terms such as constant values (`rml:constant`), references (`rml:reference`), and URI templates (`rml:template`). They are used into the maps of subjects (`rml:subjectMap`), predicates (`rml:predicateMap`), and objects (`rml:objectMap`) that make up the triples that populate the KG. SHACL defines shapes and constraints for validating RDF data, ensuring structural and semantic consistency. These constraints are applied to nodes (`sh:NodeShape`), and properties (`sh:PropertyShape`), and focus on classes (`sh:targetClass, sh:class`), property datatypes (`sh:datatype`), or enforce SPARQL query-based constraints (`sh:SPARQLConstraintComponent`).

2.2. Software architecture

The tool follows a modular pipeline architecture. An overview of the software is depicted in Figure 1. The tool takes as input the change description in RDF, compliant with the OWL Change Ontology (OCH) [18], the outdated RML mappings for the previous ontology version, the outdated SHACL shapes for the previous ontology version, and the current ontology version,

```
python3 -m ontoripple --changes path_to_change_kg.nt
--mapping path_to_old_mapping.rml.ttl
--shapes path_to_old_shapes.sh
--ontology path_to_new_ontology.ttl
--new path_output_mappings.rml.ttl
```

Listing 1: Execution Example

it is required for inference tasks when propagating changes. The change operations that are applied to the semantic artifacts are atomic in nature. An atomic change is the smallest change that can be recorded between two versions of an ontology, for OWL ontologies this is the addition/deletion of an axiom. By processing atomic changes the ambiguity of change operations is reduced to a minimum.

Change Propagation Engine: This is the component that starts the change propagation process. There is a total of 25 change operations from the OWL Change Ontology that OntoRipple provides support for. It processes changes in a predefined order since some change operations have to take place before others so that the intended effect can take place. For instance, if within a changelog there are AddClass and AddSubClass changes involving the same class, then the AddClass change would have to take effect first to add the TriplesMap/NodeShape before the subClass relationship is added. In table 2 it can be seen how it was ordered and which change operations have an effect on mappings and which ones have an effect on shapes. The number of change operations that have an effect on SHACL shapes is higher than those that have an effect on RML mappings because most class relations, property relations, and property characteristics have very little effect on how the data is materialized, but are very important for the validation rules for that data, and have a higher impact. The tool, in that order, calls on the functions that correspond to the change type for each of the semantic artifacts that have been provided as a parameter.

Evolution Handlers: The evolution handler contains 35 different functions that apply the change operations of OCH to the RML mappings and the SHACL shapes. The Add/Remove Characteristic change operations provide support for seven different OWL 2 property characteristics. In OWL, properties are fundamentally divided into object properties and data properties based on the type of values they relate to. The other types of OWL properties are not distinct property kinds but property characteristics, since they capture behavioral semantics rather than structural typing and have been modeled in OCH as property characteristics. The change operations that are

Table 2: Change Operations Processing Order and Supported Artefacts

Order	Change Operation	RML	SHACL
1	Add Class	✓	✓
2	Add Subclass	✓	✓
3	Add Object Property	✓	✓
4	Add Data Property	✓	✓
5	Remove Class	✓	✓
6	Remove Subclass	✓	✓
7	Remove Object Property	✓	✓
8	Remove Data Property	✓	✓
9	Deprecate Entity	✓	✓
10	Revoke Deprecation	✓	✓
11	Rename Entity	✓	✓
12	Add Equivalent Class		✓
13	Remove Equivalent Class		✓
14	Add Disjoint Class		✓
15	Remove Disjoint Class		✓
16	Add Characteristic		✓
17	Remove Characteristic		✓
18	Add Inverse Property		✓
19	Remove Inverse Property		✓
20	Add Disjoint Property		✓
21	Remove Disjoint Property		✓
22	Add Subproperty		✓
23	Remove Subproperty		✓
24	Add Equivalent Property		✓
25	Remove Equivalent Property		✓

supported by the tool are those that Add, Remove, Deprecate, Undeprecate, and Rename the main entities in OWL ontologies: classes, properties, annotations, class relations, property relations, property characteristics, domains, ranges, and individuals. Most of the change operations only require the change description in RDF and the outdated mappings/shapes as parameters, but others require the ontology to be provided for information regarding datatype and parentage relations between terms.

Declarative SPARQL queries: The tool uses SPARQL queries for graph manipulation for updating the mappings and shapes. The software follows a fully declarative pipeline, since the SPARQL queries specify what the pattern of triples that are desired is, it is up to the SPARQL engine to decide how to obtain those triplets, making this software very adaptable. The SPARQL

queries used for enacting the changes on the semantic artifacts improve maintainability, since those are not only in the code, but also in a set of templates[1], as seen in Figure 1, and can be easily adapted for a wide variety of the components of RML and SHACL, and cases more specific than those covered.
As seen in the Figure 1 the output of the tool is the updated set of SHACL shapes, the updated set of RML mappings, the file with the deprecated mappings (SHACL has a component for deprecating, so it does not need its own file), and a file with the RML mappings to be reviewed by the user whenever an assumption regarding inherited predicate object maps has been made by the tool.

2.3. Software functionalities

The OntoRipple tool provides functionalities for the automatic propagation of changes from ontology to RML mappings and SHACL shapes to ensure consistency as ontologies evolve.
The core functionalities of the tool are the following:

1. **Propagate changes over existing or upcoming W3C recommendations**: The tool provides support for applying ontology changes to SHACL and RML. Both of these standards are widely used in academia, industry, and European projects. There are queries defined for each of the change operations and each of the artefacts. For instance, in RML mappings TriplesMaps are created/deleted when a class is added/removed from the ontology. In SHACL if an inverse property is added/removed an `sh:SPARQLConstraintComponent` is added/removed that ensures that the constraint is being enforced.
2. **Review System**: In the Remove Class change operation, when removing a class if that class has a parent class in the ontology the `PredicateObjectMaps` can either be removed or added to the parent class's `TriplesMap` instead of being removed. In our tool when this happens those `PredicateObjectMaps` are inserted in a new graph and exported to the Review Mappings file to be reviewed by the user.

3. Illustrative examples

Illustrative examples from different use cases for this research will be provided. First, there is the Emolex[2] ontology, an ontology that semantically models emotional lexica, which will be used to exemplify the step-by-step

[1] `https://github.com/oeg-upm/OntoRipple/tree/main/queries`
[2] `https://w3id.org/def/emolex`

propagation of a single change to RML and SHACL shapes. Then, we will provide a wider example of the usage of the Ontoripple tool to propagate an entire changelog for the eProcurement ontology, which models procurement data for the member states of the European Union. Data on the changes, and the structures affected from RML and SHACL will be provided, to test the suitability of the tool for production environments.

3.1. Atomic Change Execution Example

The Emolex ontology[3] is a domain-specific ontology designed to semantically model emotional lexica. It provides a vocabulary to represent emotional words and their associated metrics, such as valence, arousal, and dominance, based on psychological models. The EmoW-KG[4] is a large-scale knowledge graph aligned with the EmoLex ontology. It currently integrates 71,284 emotional words sourced from 18 existing datasets, annotated with their corresponding affective scores, which are represented in almost 2 million RDF triples. EmoW-KG is the result of a reproducible and modular construction pipeline using standard technologies such as RML and SHACL to facilitate the incorporation of new sources and updates for long-term maintainability and scalability. This makes it suitable for evaluating the OntoRipple tool, since it uses different semantic artifacts and it models a different domain. In this section a step by step propagation of a change will be displayed.
The data involved in this section are available here[5]; the change data has been manually crafted from the Github commits from the emolex repository. The ontology, shapes, and mappings have been obtained from their corresponding repositories. The code in Listing 2 represents the added code as seen in the GitHub commits from the ontology. The code in Listing 3 models that same change using OCH. The change is the addition of the OWL Class emolex:BahaviouralMeasures to the ontology.

```
@prefix emolex: <https://w3id.org/def/emolex#> .
emolex:BehavioralMeasures rdf:type owl:Class .
```

Listing 2: Added Class in Emolex

OntoRipple iterates over the changes provided by the changelog and processes them individually. First, the appropriate python function is selected by the SPARQL query from Listing 4 for each specific change. Then, for that

[3] https://github.com/citiususc/emolex/
[4] https://github.com/citiususc/semantic-emofinder
[5] https://github.com/oeg-upm/OntoRipple/tree/main/examples/emolex

```
@prefix emolex_changes: <https://w3id.org/def/emolex-changes> .
@prefix och: <https://w3id.org/def/och#> .
emolex_changes:ac2 rdf:type och:AddClass;
    och:addedClass emolex:BehavioralMeasures .
```

Listing 3: Added Class OCH compliant

```
PREFIX och: <http://w3id.org/def/och#>
SELECT DISTINCT ?change WHERE {
    ?change rdf:type och:AddClass .
}
```

Listing 4: Change Select Query

change, the associated RDF terms are selected in Listings 5, specifically the class that has been added.
In the case of RML mappings, it performs the following set of steps: first, it checks whether there is already an `rml:subject` that points to that class in Listing 6. If there is one, then no new mappings are added; if there are none, then a template `rml:TriplesMap` will be added, as seen in Listing 7.

```
PREFIX och: <http://w3id.org/def/och#>
SELECT DISTINCT ?class WHERE {
    <https://w3id.org/def/emolex-changesac2> och:addedClass ?class .
}
```

Listing 5: Added Class Query

```
PREFIX rml: <http://semweb.mmlab.be/ns/rml#>
PREFIX emolex: <https://w3id.org/def/emolex#>
ASK {
    ?triples_map rdf:type rml:TriplesMap .
    ?triples_map rml:subjectMap ?subject .
    ?subject rml:class emolex:BehavioralMeasures
}
```

Listing 6: Check SubjectMap Query

In the case of SHACL shapes, it inserts a `sh:NodeShape` that targets the class to which it is being added. The corresponding property shapes are added when the property changes have taken place. The query is in Listing 8.

```
PREFIX rml: <http://semweb.mmlab.be/ns/rml#>
INSERT DATA {
    ex:BehavioralMeasures_TM a rml:TriplesMap ;
        rml:logicalSource [
           rml:source [ "XXX" ] ;
           rml:referenceFormulation "XXX" ;
        ] ;
        rml:subjectMap [
            rml:template "XXX" ;
            rml:class emolex:BehavioralMeasures
        ] .
}
```

Listing 7: Insert TriplesMap Query

```
PREFIX sh: <http://www.w3.org/ns/shacl#>'
INSERT DATA {
    ex:BehavioralMeasuresShape a sh:NodeShape ;
        sh:targetClass emolex:BehavioralMeasures .
}
```

Listing 8: Insert NodeShape Query

3.2. Full Change Log Execution Example

The EU Public Procurement Data Space (PPDS) [10] aims to provide a semantic layer of public and private procurement data across Europe, allowing among others, the calculation of a set of standard performance indicators for each EU member state following a systematic approach. Technically, PPDS aims to construct a decentralised KG, by declaratively mapping any procurement data from each member state to the e-Procurement Ontology (ePO)[6]. However, the ontology's development remains ongoing, resulting in continuous updates with the integration of new features and knowledge. The ontology code is publicly available in a GitHub repository[7] and changes between consecutive versions are documented in the form of HTML files[8].Each time a new version of ePO is released, PPDS knowledge graph engineers process the release notes and manually accommodate the associated mappings, and SHACL shapes to produce and validate the desired RDF data.

[6] https://docs.ted.europa.eu/EPO/latest/

[7] https://github.com/OP-TED/ePO

[8] https://docs.ted.europa.eu/EPO/latest/release-notes

Table 3: Change Description form 3.0.0-3.0.1 ePO

Operation type	Count
AddDomain	49
AddObjectProperty	34
AddRangeObjectProperty	34
RemoveDomain	34
RemoveObjectProperty	26
RemoveRangeObjectProperty	26
AddDataProperty	15
RemoveDataProperty	8
AddClass	3
AddSubClass	3
Total	232

For this usage example in the PPDS context, the changelog between the 3.0.0 and 3.0.1 versions[9] will be used. Since the release notes are in HTML format, a custom script, and RML mappings have been used for generating the change description in RDF[10]. The change description in RDF can also be generated using the Widoco extension [28], or manually crafted.
Table 3, shows the data regarding the change operations that can be found in the ePO changelog between these ontology versions. Then we execute the OntoRipple tool providing as input the ontology, outdated mappings, outdated shapes, the current ontology, and a path for where the updated mappings, and shapes will be generated. The data used can be found in the project repository[11].
Table 4a displays the changes that have been produced by the change operation from the 3.0.0-3.0.1 changelog 3 in the RML mappings. The AddClass operations have resulted in new TriplesMap, with new LogicalSources and SubjectMaps. One of these change operations has not taken place because that class was already in the RML mappings. The addition/removal of a property results in the addition/removal of a PredicateObjectMap if it is not already within mappings. Also, many of the ontology modifications have been represented within the HTML changes as Remove and Add changes;

[9] https://docs.ted.europa.eu/EPO/3.0.1/release-notes.html

[10] https://github.com/oeg-upm/OntoRipple/tree/main/examples/ppds/input_data/epo-changes_data

[11] https://github.com/oeg-upm/OntoRipple/tree/main/examples/ppds/3.0.0-3.0.1_example

Table 4: Differences in RML Mappings and SHACL Shapes between the ePO 3.0.0 and 3.0.1 versions. In RML, TM* has one SubjectMap and one LogicalSource also associated.

Component	Count
TM* — Added	2
TM* — Replaced/Modified	3
TM* — Removed	0
POMs — Added	15
POMs — Replaced/Modified	34
POMs — Removed	0

(a) RML Mappings (TM = TriplesMap, POMs = PredicateObjectMaps).

Component	Count
NS — Added	3
NS — Replaced/Modified	0
NS — Removed	0
PS — Added	13
PS — Replaced/Modified	168
PS — Removed	0

(b) SHACL Shapes (NS = NodeShape, PS = PropertyShape)

that is why the number of POM removed is 0 and the modified ones is 34 in Table 4a despite the high number of removals in Table 3.
Table 4b shows the changes that have been produced by the change operation from the 3.0.0-3.0.1 changelog (Table 3) in the SHACL shapes. The AddClass operations have resulted in the new NodeShapes. The addition/removal of a property results in the addition/removal of a PropertyShape if it is not already within mappings. The ontology modifications have been represented within the HTML changes as Remove and Add changes; that is why the number of modified/replaced Property shapes is 168 whereas the removed properties are 0.

4. Impact

The OntoRipple software tool successfully manages to apply the automatically generated OWL ontology changes to the RML mappings and SHACL shapes, reducing the manual effort of the knowledge graph engineers. Currently, this is the only open source approach that provides this functionality, and has the potential to save time, money, and other resources in any KG driven project. This tool has successfully answered the following research questions:

- **RQ1**: How much of the ontology change propagation to semantic artifacts process can be automated for RML mappings and SHACL shapes?

Other contributions we have made in this line of research are the following: The OWL Change Ontology (OCH) [18] was created to ease representing change operations in RDF, and thus allows for better processing of changes for this tool. The Widoco [29] tool for automatic generation of ontology documentation was extended [28] to automatically extract the OCH-compliant

changes between two ontology versions. LOT4KG [30, 31], an extension of the LOT methodology [20] was developed to address the gap in methodologies that integrate the ontology lifecycle and KG lifecycle; this tool covers one of the most important aspects of the methodology, the change detection and impact analysis of the ontology changes.
OntoRipple opens new avenues for expansion that result in new research questions.

- There have been other ontology change representation proposals for complex changes [16] made of the a set of simple changes. Complex operations are made up of simpler ones, but they can provide more information regarding the motivation of certain changes, and how those can be propagated remains unclear. **RQ2**: What is the impact that the representation of complex changes has on the propagation of changes?

- SHACL and RML are standards for the construction and the validation of ontologies, used in many projects, there are many more semantic artifacts, used for different tasks, such as the KG exploitation, that face the same problems during the KG lifecycle and the tool can be extended to include those **RQ3**: Can the modeled change propagation mechanisms be generalized and extended to other semantic artifacts, enabling consistent evolution and synchronization across the entire semantic ecosystem?

- The tool reduces the manual labor required to update the RML mappings to regenerate the graph, however the entire graph has to be regenerated and re-uploaded. A way to apply these changes in patches could have a positive effect on performance and reduce computational costs. **RQ4**: How does this approach affect performance? Can these changes be applied as patches to avoid regenerating the entire graph, or validate all the rules?

- Just as the ontology changes over time the data sources that make up the knowledge graph undergo many changes. Thus, the RML mappings have to be modified to correctly regenerate the graph, and the SHACL shapes have to be modified so that it validates the updated data sources. **RQ5**: How can the changes in data sources be conceptualized and propagated to RML and SHACL?

5. Conclusions

In this paper, we have presented a software tool that allows for the integration of ontology evolution within the KG construction and validation lifecycle.

The tool, taking the ontology change description, updates the corresponding RML mappings and SHACL shapes to accommodate the changes that have taken place between ontology versions. The ontology change description in RDF can be automatically generated between ontology versions [28]. OntoRipple provides support for 25 distinct change operations, covering most of the OWL expressivity. We have also tested the tool on a use case for the eProcurement Ontology, where the changes between ontology versions documented as HTML have been successfully propagated over the project's RML mappings and SHACL shapes. The main limitation of OntoRipple is that full automation has not been achieved. Since the RML mappings are between the ontology and the input data sources, there are parts of the mappings that are dependent on the data sources. The tool focuses on the ontology changes and does not account for the input data changes that also affect the mappings [32].

For future work, we will perform a performance study to analyse the impact the tool can have on the regeneration and validation of the graph, with a focus on the execution time and the memory consumption when dealing with large ontologies or complex sets of changes. The OWL Change Ontology is currently being updated to include complex changes, Ontoripple will be updated to propagate said changes. A Github Action that generates the RDF Change Data whenever a new commit is made will be created. Constructing the updated mappings in the form of patches that can be applied individually instead of regenerating the whole graph is an avenue for improving the KG maintenance as well. We will study how the inverse propagation process, where we propagate changes from the input data sources to the ontology and semantic artifacts, can be performed. The inclusion of other semantic artefacts represented by different language is also being considered.

Acknowledgements

David Chaves-Fraga is funded by the Agencia Estatal de Investigación (Spain) (PID2023-149549NB-I00), the Xunta de Galicia — Consellería de Cultura, Educación, Formación Profesional e Universidades (Centro de investigación de Galicia accreditation 2024–2027 ED431G-2023/04 and Reference Competitive Group accreditation 2022–2026, ED431C 2022/19) and the European Union (European Regional Development Fund–ERDF).

References

[1] R. Studer, V. R. Benjamins, D. Fensel, Knowledge engineering: Principles and methods, Data & knowledge engineering 25 (1-2) (1998) 161–197. doi:10.1016/S0169-023X(97)00056-6.

[2] RDF-Schema 1.1, https://www.w3.org/TR/rdf11-schema/, accessed: 2025-04-29 (2024).

[3] S. Bechhofer, F. Van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, L. A. Stein, et al., Owl web ontology language reference, W3C recommendation 10 (2) (2004) 1–53.

[4] C. Sundara, S. Das, R. Cyganiak, R2RML: RDB to RDF mapping language, W3C recommendation, W3C, http://www.w3.org/TR/2012/REC-r2rml-20120927/ (Sep. 2012).

[5] A. Iglesias-Molina, D. Van Assche, J. Arenas-Guerrero, B. De Meester, C. Debruyne, S. Jozashoori, P. Maria, F. Michel, D. Chaves-Fraga, A. Dimou, The RML Ontology: A Community-Driven Modular Redesign After a Decade of Experience in Mapping Heterogeneous Data to RDF, in: International Semantic Web Conference, Springer, 2023, pp. 152–175.

[6] H. García-González, I. Boneva, S. Staworko, J. E. Labra-Gayo, J. M. C. Lovelle, Shexml: improving the usability of heterogeneous data mapping languages for first-time users, PeerJ Computer Science 6 (2020) e318.

[7] E. Daga, L. Asprino, P. Mulholland, A. Gangemi, et al., Facade-X: an opinionated approach to SPARQL anything, Studies on the Semantic Web 53 (2021) 58–73.

[8] Shapes Constraint Language (SHACL), https://www.w3.org/TR/shacl/, accessed: 2025-08-07 (2017).

[9] E. Prud'hommeaux, J. E. Labra Gayo, H. Solbrig, Shape expressions: an rdf validation and transformation language, in: Proceedings of the 10th International Conference on Semantic Systems, 2014, pp. 32–40.

[10] The Public Procurement Data Space (PPDS), https://single-market-economy.ec.europa.eu/single-market/public-procurement/digital-procurement/public-procurement-data-space-ppds_en, accessed: 2025-04-29 (2023).

[11] European Research Area (ERA), https://research-and-innovation.ec.europa.eu/strategy/strategy-research-and-innovation/our-digital-future/european-research-area_en, accessed: 2025-09-04 (2024).

[12] E. Ruckhaus, J. Toledo, E. A. Martínez, C. Oscar, Lessons learned from the combined development of ontologies using owl and shacl, in: The Thirteenth International Conference on Knowledge Capture, 2025.

[13] Graph-Massivizer, https://graph-massivizer.eu/, accessed: 2025-10-17.

[14] A. Polleres, R. Pernisch, A. Bonifati, D. Dell'Aglio, D. Dobriy, S. Dumbrava, L. Etcheverry, N. Ferranti, K. Hose, E. Jiménez-Ruiz, et al., How does knowledge evolve in open knowledge graphs?, Transactions on Graph Data and Knowledge 1 (1) (2023) 11–1.

[15] N. F. Noy, M. Klein, Ontology evolution: Not the same as schema evolution, Knowledge and information systems 6 (2004) 428–440.

[16] M. Hartung, A. Groß, E. Rahm, COnto–Diff: generation of complex evolution mappings for life science ontologies, Journal of biomedical informatics 46 (1) (2013) 15–32.

[17] R. Palma, P. Haase, Ó. Corcho, A. Gómez-Pérez, Change Representation For OWL 2 Ontologies, in: R. Hoekstra, P. F. Patel-Schneider (Eds.), Proceedings of the 5th International Workshop on OWL: Experiences and Directions (OWLED 2009), Chantilly, VA, United States, October 23-24, 2009, Vol. 529 of CEUR Workshop Proceedings, CEUR-WS.org, 2009.
URL https://ceur-ws.org/Vol-529/owled2009_submission_14.pdf

[18] The OWL Change Ontology, https://w3id.org/def/och, accessed: 2025-08-08 (2025).

[19] F. Zablith, G. Antoniou, M. d'Aquin, G. Flouris, H. Kondylakis, E. Motta, D. Plexousakis, M. Sabou, Ontology evolution: a process-centric survey, The knowledge engineering review 30 (1) (2015) 45–75.

[20] M. Poveda-Villalón, A. Fernández-Izquierdo, M. Fernández-López, R. García-Castro, Lot: An industrial oriented ontology engineering framework, Engineering Applications of Artificial Intelligence 111 (2022) 104755.

[21] SPARQL 1.1 Query Language, https://www.w3.org/TR/sparql11-query/, accessed: 2025-08-08 (2013).

[22] D. Lembo, R. Rosati, V. Santarelli, D. F. Savo, E. Thorstensen, Mapping repair in ontology-based data access evolving systems, in: IJCAI International Joint Conference on Artificial Intelligence, International Joint Conference on Artificial Intelligence, 2017, pp. 1160–1166. doi:10.24963/ijcai.2017/161.

[23] A. Cimmino, A. Fernández-Izquierdo, R. García-Castro, Astrea: Automatic generation of shacl shapes from ontologies, in: European Semantic Web Conference, Springer, 2020, pp. 497–513.

[24] X. Duan, D. Chaves-Fraga, O. Derom, A. Dimou, Scoop all the constraints' flavours for your knowledge graph, in: European Semantic Web Conference, Springer, 2024, pp. 217–234.

[25] K. Rabbani, M. Lissandrini, K. Hose, Extraction of validating shapes from very large knowledge graphs, Proceedings of the VLDB Endowment 16 (5) (2023) 1023–1032.

[26] E. Pürmayr, Shacl shapes extraction for evolving knowledge graphs, Ph.D. thesis, Technische Universität Wien (2025).

[27] OntoRipple Github Repository, https://github.com/oeg-upm/OntoRipple, accessed: 2025-09-04 (2025).

[28] Widoco Extension, https://github.com/DiegoCondeHerreros/Widoco-OCH, accessed: 2025-04-23 (2025).

[29] D. Garijo, Widoco: a wizard for documenting ontologies, in: The Semantic Web–ISWC 2017: 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part II 16, Springer, 2017, pp. 94–102. doi:10.5281/zenodo.591294.

[30] R. Pernisch, M. Poveda-Villalón, D. Conde-Herreros, D. Chaves-Fraga, L. Stork, When Ontologies met Knowledge Graphs: A Methodology Tale, in: The Semantic Web: ESWC 2024 Satellite Events, LNCS, Hersonissos, Greece, 2024.

[31] LOT4KG, LOT for Knowlege Graphs Construction, https://lot.linkeddata.es/LOT4KG/, accessed: 2025-08-08 (2024).

[32] D. Van Assche, J. A. Rojas, B. De Meester, P. Colpaert, Incremental knowledge graph construction from heterogeneous data sources, Semantic Web (Under Review) (2024).