Modelling & Analyzing Changes within LD Source Data

Alex Randles[1] and Declan O'Sullivan[1]

[1] *ADAPT Centre for Digital Content, Trinity College Dublin, Ireland*

Abstract

Linked data (LD) datasets are highly dynamic in nature with continuously changing data. These datasets are often created using mappings which transform the source data into Linked Data format. These datasets may not be regenerated and changes which occur within the source data may not be reflected in the Linked Data dataset. These source data changes could impact the quality of the linked data, or the artefacts used to create the Linked Data. Capturing information relating to these changes using a domain specific ontology will result in expressive machine-readable information which can facilitate the propagation of such changes into the Linked Data dataset. A system which uses the ontology to detect the impact of source data changes on the mapping artefacts and Linked Data dataset would promote high-quality and availability of fresh up-to-date linked data. In this paper we focus on the design of a proposed ontology which is used to capture information relating to changes which have occurred within the source data that have been used to generate Linked data datasets. Furthermore, the ontology is used within a proposed component within a mapping quality framework to allow users to easily analyze and address the impact of the changes on the mapping artefacts and the resulting linked data dataset. Moreover, the component regularly monitors the source data to detect and represent recent changes. In the paper an example of how the component and ontology is being applied in an example industry led use case is also described. It is hoped that the approach proposed offers one possible unified approach to ensure that linked data is kept in sync with underlying changes in source data.

Keywords

Dynamics; Linked Data; Mappings; Data Quality; Ontology Design.

1. Introduction

Linked data datasets are highly dynamic with resources continuously being added or removed [24]. "Freshness" is often mentioned within the context of dataset dynamics [3], which has been referred to as a quality dimension that relates to the age and occurrences of changes within data. It has been described as one of the most important attributes of data quality [3]. Data publishers are continuously attempting to improve the quality of data by updating vocabularies or adding and removing resources in datasets [25]. Consequently, there is a need to tackle the dynamics related to constantly evolving linked data [12].

The research challenge of how to identify and handle these changes has been present within the Linked Data community for over a decade [24]. However, there exists no consensus about how best to detect and propagate changes that occur within the linked data domain [24]. While approaches exist [12,13,28,29] to model and handle changes within the resulting data, none offer an approach which links and analyses source data changes with the corresponding mapping artefacts. We hope to tackle these state-of-the-art limitations by proposing a unified approach which would allow users to represent changes using a specific ontology with associated polices which would ensure the linked data is kept in sync with the underlying data sources. Furthermore, ensuring these changes are propagated into the resulting data will help to promote the freshness and quality of the linked data. Moreover, providing a

8th Workshop on Managing the Evolution and Preservation of the Data Web (MEPDaW)
co-located with the 21st International Semantic Web Conference (ISWC 2022)
EMAIL: alex.randles@adaptcentre.ie (A. 1); declan.osullivan@adaptcentre.ie (A. 2)
ORCID: 0000-0001-6231-3801 (A. 1); 0000-0003-1090-3548 (A. 2)

change notification mechanism will ensure that linked data publishers are aware of what changes in source data have occurred in a timely fashion, allowing them to take appropriate actions to prevent a decrease in data quality of the linked data that they publish [12].

In this paper we discuss the design and evaluation of our proposed Change Detection Ontology (CDO) which is used to capture changes which have occurred in the source data of published Linked Data datasets. The ontology is specifically targeted at datasets which have been created using declarative uplift mapping artefacts. Furthermore, we showcase the design of a Change Detection component which uses the ontology to define information related to changes which have occurred in source data in formats such as XML, CSV, and relational databases. Moreover, the impact of these changes on the declarative uplift mappings is analyzed by the component. The Change Detection component [20] has been integrated within our previously introduced Mapping Quality Vocabulary (MQV) [15,16] framework [17,18]. The framework was originally designed to assess and refine the quality of uplift mappings, however, the design has been extended to include the component such that quality processes are triggered depending on the type of changes that have been detected in the source data. The evolution of the MQV framework is now referred to simply as the **Mapping Quality (MQ) framework,** which we plan to evolve into a linked data quality improvement ecosystem. The objective of the research is to improve mapping and dataset quality while promoting mapping maintenance and re-use. The remainder of this paper is structured as follows: **Section 2** describes the design and evaluation of the Change Detection Ontology (CDO); **Section 3** discusses the design and implementation of the Change Detection Component; **Section 4** discusses related work in the area and **Section 5** concludes the paper and outlines some future work.

2. Change Detection Ontology

The Change Detection Ontology (CDO) [19] is designed to represent and interchange information related to changes which have occurred in the source data of Linked Data datasets. The objective of the ontology is to represent the information in a machine-readable format which would allow mapping engineers and software agents to analyze the changes and update the mapping where appropriate. Furthermore, the ontology could be used to indicate that a dataset needs to be regenerated as important changes in the source data may not being reflected in the current published version. It is hoped the ontology will assist in improving mapping and dataset quality while promoting mapping maintenance and reuse. The ontology is primarily targeted at data producers and consumers of Linked Data who are concerned with freshness and quality of linked data.

2.1. Ontology Design

The proposed Change Detection Ontology (CDO) extends the Linking Open Descriptions of Events (LODE) [22] vocabulary to represent the changes. LODE is designed to represent historical linked data events and allows linking with other event related ontologies. In addition, the ontology reuses the Rei Policy Ontology [8] to represent a notification policy which defines when users are notified of changes. The Rei Policy Ontology is a universal policy language which can be applied in various domains. The flexibility of the policy language makes it suitable for modelling the necessary information. The design of the CDO ontology has been inspired by previous work in linked data change detection [3,20,25,28,29]. **Figure 1** shows the class interaction diagram for CDO. Concepts within the diagram are represented using different colors.

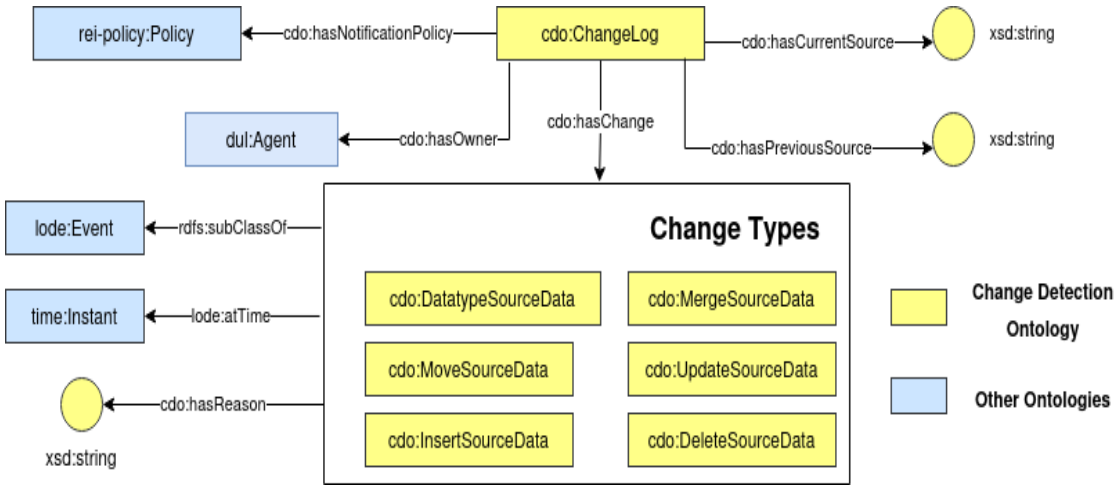

Figure 1: Class interaction diagram of Change Detection Ontology (CDO)

Changes which occur in the source data are represented by different change types. For instance, if data is inserted into the source data, a data insertion change (`cdo:InsertSourceData`) has occurred. Defining types allows changes to be identified and processed uniquely when required. The model is extendible where users can create new change types using the same format. Categorizing changes based on the action which caused them was inspired by related work [13,21,23,25] within the state of the art where changes within linked data have been categorized. All the change types are subclass of the LODE Event (`lode:Event`) which is used to represent non-specific events within the linked data domain. The reuse of LODE was inspired by previous work [13] in modelling and detecting changes in the linked data domain. Each change has information related to why the change has occurred (`cdo:hasReason`) and when the change occurred (`time:Instant`), which allows the linked data to be synchronized with the source data. The changes are grouped into a log (`cdo:ChangeLog`) which enables related changes to be grouped. The source data is represented by the current (`cdo:hasCurrentSource`) and the previous version (`cdo:hasPreviousSource`) of the data. For relational databases the source data is represented using the connection information. The details of the source data allow changes to be detected periodically to capture recent changes. The log has one or more owners (`dul:Agent`). The DUL ontology [4] is designed to model things and is used by LODE to model actors. Furthermore, each log has a notification policy (`rei-policy:Policy`) which represents when users are notified of changes, allowing them to take appropriate action. The definition of a notification policy has been inspired by sparqlPuSH [12] which provides notifications of data changes within linked data stores. Once the notification policy has been triggered, changes are no longer detected for that specific process.

2.2. Ontology Evaluation

The design of the ontology has been evaluated using three different methods which include the usage of Protégé, Competency Questions and the Ontology Pitfall Scanner.

Protégé. Protégé [26] is an open-source tool which provides the capability to create and edit ontologies using an intuitive graphical interface. Plugins are also available which allow extra functionality such as alternative visualization. Furthermore, reasoners are available which can be used to detect inconsistencies within the ontology design. CDO was created and reasoned over using Protégé. No inconsistencies could be found in protégé while designing the ontology[2].

Ontology Competency Questions. Competency Questions [2] are used to represent the requirements of the ontology. These questions state information which the ontology should contain and are often defined in natural language. These questions ensure that design requirements have been satisfied. Furthermore, these questions can be answered by using SPARQL queries to query the ontology or

[2] CDO displayed in Protégé at https://github.com/alex-randles/MEPDaW-2022/blob/main/Protege_Screenshot.png

instances. Competency questions[3] have been designed for CDO and have been answered using SPARQL queries which query sample instances. Each competency question was answered correctly. **Table 1** shows an example of a competency questions used to validate the design requirement of CDO.

Table 1: Example of an Ontology Competency Questions designed for CDO

Question	SPARQL Query	Sample Answer

What type of changes occurred?	`SELECT ?changeType`
	`WHERE {`
	`?changeLog a cdo:ChangeLog;`
	`cdo:hasChange ?change .`
	`?change a ?changeType.`
	`}`
		`cdo:InsertSourceData`
		`cdo:DeleteSourceData`

Who owns the data?	`SELECT ?owner`
	`WHERE {`
	`?changeLog a cdo:ChangeLog;`
	`cdo:hasOwner ?owner.`
	`}`
		`ex-user:1`

Ontology Pitfall Scanner. OOPS! (Ontology Pitfall Scanner) [14] is a web-based tool which is used to detect issues within the design of an ontology. The tool provides a method to validate and verify the design of an ontology. Furthermore, the tool is independent of any ontology development environment. Moreover, the tool provides recommendations for how issues can be repaired. The issues are represented in different severity which include minor, important, and critical. Critical issues must be repaired to ensure an adequate quality level. CDO has been input into the tool and a report has been generated[4]. The report details that there are no quality issues present within the ontology design. The report validates that the ontology is sufficient for usage by potential users.

3. Change Detection Component

The following section discusses the motivation and design of the component in Section 3.1. Then in Section 3.2, an example of the component applied to an industry relevant use case is described.

3.1. Design

The Change Detection Component [20] is a component within the MQ framework [18,20] which is a framework designed to produce high-quality and fresh linked data. The component is designed to capture changes which have occurred in the source data of Linked Data datasets which have been created by mapping artefacts. Furthermore, the component analyses these changes to detect if these changes could impact mapping quality or dataset quality. It is hoped the component will allow factors which could impact quality to be captured and removed early. Furthermore, the process could improve mapping maintenance and reuse. **Figure 2** shows a screenshot of the Change Detection Component interface displaying information relating to sample CSV source data.

The component was designed using Python libraries for creating a web application (Flask [6]) and querying the RDF data created (RDFLib [9]). Furthermore, an R2RML [5] engine was used to uplift the information related to changes which have occurred and the notification policy.

The current implementation can detect changes within data stored in relational databases or represented in CSV or XML format. The process starts by the users entering the source data details which include the URL of the file versions for CSV and XML or the relational database server details. Thereafter, changes are detected within the data and stored within a relational database. Furthermore, the users enter the notification thresholds for each change type and an email address where the notification can be sent when these thresholds have been reached. These details are also stored within

[3] Ontology competency questions and answers at https://github.com/alex-randles/MEPDaW-2022/blob/main/CDO%20Competency%20Questions.pdf

[4] Report generated by OOPS! executed on CDO at https://github.com/alex-randles/MEPDaW-2022/blob/main/OntOlogy%20Pitfall%20Scanner!%20-%20Results.png

the relational database. Storing the details within a relational database allows the information to be uplifted into Linked data format using R2RML mappings [5]. The graph generated contains the changes detected, notification policy and contact information. The graph generated is queried to populate the user interface of the component.

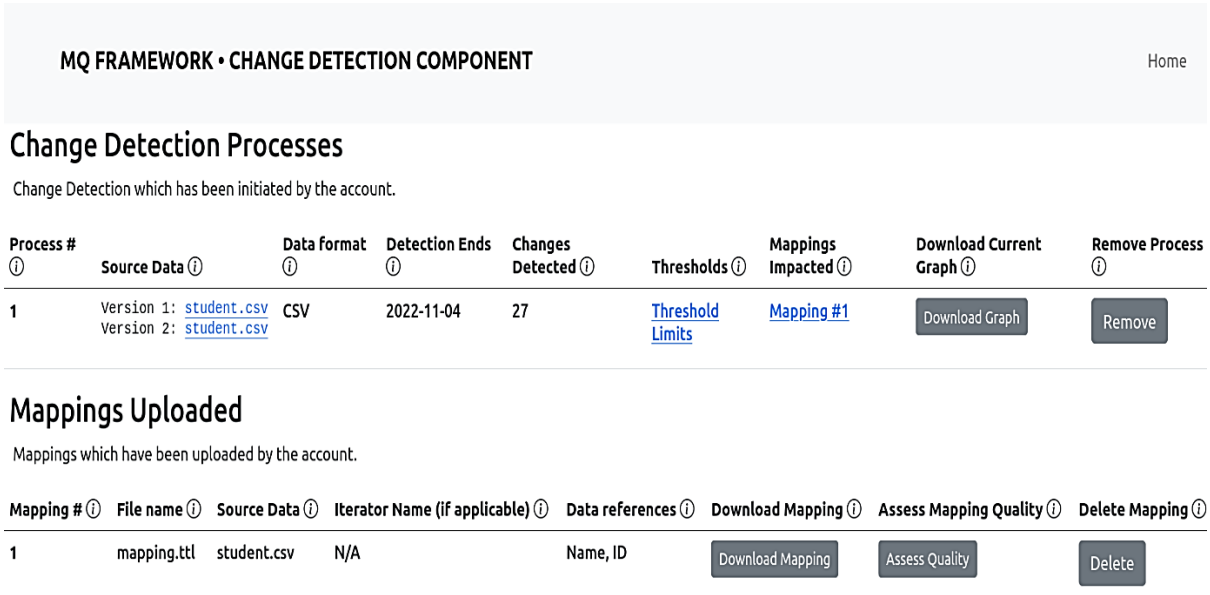

Figure 2: Screenshot of Change Detection Component

A sample graph[5] has been generated using the RML test case files[6]. **Table 2** shows an extract of the graph, which includes 1 insert change and sample notification policy details.

Table 2: Extract of sample graph generated by the component

```
change-log:0 a cdo:ChangeLog;
             cdo:hasOwner user:1;
             cdo:hasChange insert-change:0;
             cdo:hasCurrentVersion <https://raw.githubusercontent.com/kg-
construct/rml-test-cases/master/test-cases/RMLTC0009a-CSV/student.csv> ;
             cdo:hasPreviousVersion https://raw.githubusercontent.com/kg-
construct/rml-test-cases/master/test-cases/RMLTC0002a-CSV/student.csv> .

insert-change:0 a cdo:InsertSourceData;
             cdo:hasReason "ID: 10";
             lode:atTime detection-time:0 .

detection-time:0 a time:Instant;
   time:inXSDDateTimeStamp "2022-07-20 22:15:45.843376"^^xsd:dateTimeStamp .

notification-policy:0 a rei-policy:Policy;
             rei-policy:desc "Notification policy for user 1" ;
             rei-policy:grants policy-obligation:0 .

policy-obligation:0 a rei-deontic:Obligation;
             rei-deontic:action cdo:sendNotification;
             rei-deontic:obligedTo cdo:softwareAgent;
             rei-deontic:startingConstraint notification-constraint:0;
             rei-policy:actor user:1, cdo:softwareAgent .
```

The change log (`change-log:0`) contains 1 insert change (`insert-change:0`) which has been detected within the CSV source data (`cdo:hasCurrentVersion`, `cdo:hasPreviousVersion`). The inserted data (`cdo:hasReason`) has been detected at a specific point in time (`lode:atTime`). The detection time is represented as a date time stamp

[5] Sample graph at https://github.com/alex-randles/MEPDaW-2022/blob/main/sample_graph.ttl
[6] RML test case files at https://rml.io/test-cases/

(`time:inXSDDateTimeStamp`).The notification policy (`notification-policy:0`) is described (`rei-policy:desc`) and grants an obligation (`rei-policy:grants`). The obligation (`policy-obligation:0`) requires a notification be sent (`rei-deontic:action`) to the user (`user:1`).

3.2. Use Case

Currently, the MQ framework is being applied in a research collaboration project with Ericsson Software Technology. The use case involves the transformation and analysis of cloud native monitoring data into linked format. The information is collected from the Prometheus [27] monitoring service. The data is stored within a relational database and transformed into linked data using R2RML [5] mappings. The component has been connected to the database and mappings used within the project have been uploaded. **Figure 3** shows components involved within the use case interaction.

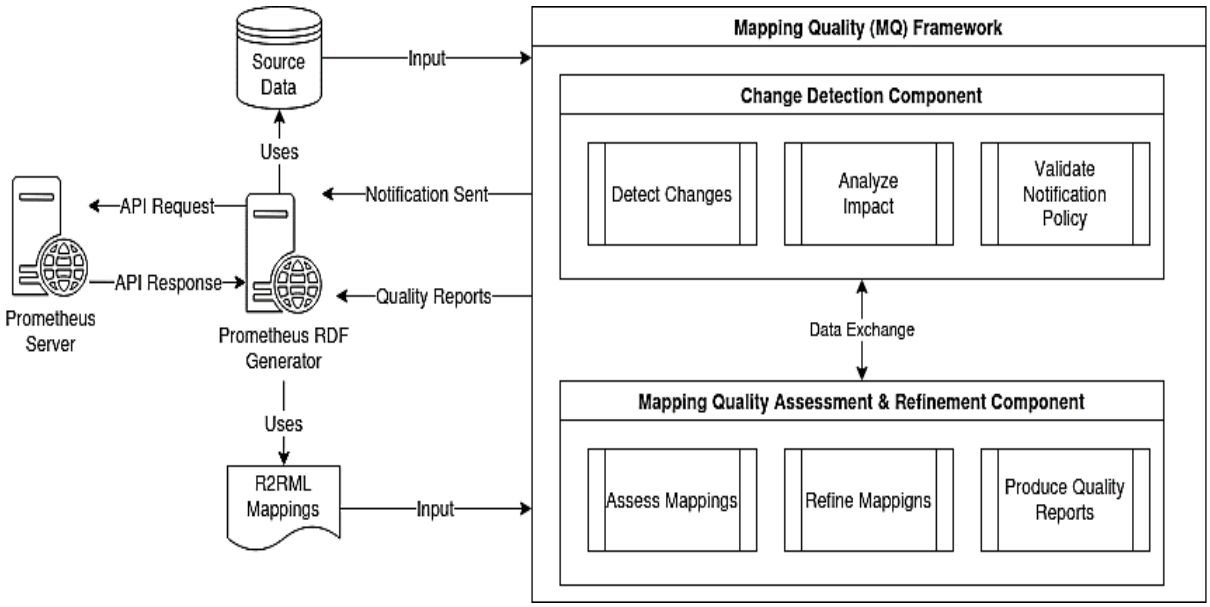

Figure 3: Components used within the use case interaction

The components[7] involved detect changes within the source data which is used to generate the graphs containing monitoring information using R2RML [5]. These components ensure that the information contained within the graphs produced by the system are high-quality and fresh, which in the context of cloud native monitoring is highly important. Furthermore, data is exchanged between each component to link semantically related information, such as mapping data sources and source data changes.

4. Related Work

The state of the art in ontologies which represent changes within the Linked Data domain has been reviewed. Furthermore, systems which use these ontologies are mentioned. These ontologies are highlighted within a survey which is focuses on dataset dynamics [24]. A system could not be found within the state of the art which relates to the impact of source data changes on the mapping and resulting Linked data dataset.

DSNotify EventSet Vocabulary [13] is designed to represent events that resulted in resource modifications in Linked data datasets. The vocabulary applies a resource-centric perspective and preserves the timely order of changes. The changes are modelled as Resource Change Events and

[7] Component description at https://github.com/alex-randles/MEPDaW-2022/blob/main/Component%20Descriptions.pdf

grouped into a container of events called an EventSet. These EventSets are represented as voiD [1] datasets.

Dataset Dynamics Vocabulary (DaDy) [28] is designed to represent information about the frequency and type of changes which are occurring within a Linked data dataset. The vocabulary is designed to be used in conjunction with voiD [1]. Furthermore, the vocabulary can be used to represent the updated source URI which allows for a change notification mechanism definition.

Talis ChangeSet [29] is designed to represent ChangeSets, which define the delta between two versions of a resource description. A resource description is described as a set of triples which include a description of a resource. The delta is represented by two set of triples which describe the addition or removal of resource descriptions. A ChangeSet can be used to modify the resource description based on which triples have been added or removed.

OWL 2 Change Ontology [11] is designed to represent changes which have occurred within ontologies. The design comprises of a fine-grained taxonomy that considers varying levels of operations. The approach consists of a generic change ontology that can be specialized for different ontology languages. The generic change ontology has been implemented as an extension of the Ontology Metadata Model (OMV) [7]. A metadata model is used to link the generic change ontology with elements of the specialized model. Furthermore, the ontology allows changes to be represented in chronological order.

The Change and Annotations Ontology (CHAO) is designed to represent ontology changes which occur in the evolution of an ontology in Protégé [26]. The initial ontology design represents basic changes. However, an extension can be defined to represent complex changes. Furthermore, multiple changes can be grouped into a higher-order of change. Moreover, metadata information about changes can be defined such as the author or timestamp.

5. Conclusion

Detecting and analyzing changes within the source data underlying Linked Data datasets which have been generated by uplift mapping artefacts, will ensure these changes can be assessed and propagated into the Linked Data dataset. The lack of a standard approach to detect and propagate changes within linked data ecosystem is a limitation within the state of art [24], and which we provide an approach to address in this paper. Source data changes could have a significant impact on how mapping artefacts are defined and maintained while ensuring the quality of data for consumers. A process which automatically detects source data changes and generates machine-readable information for analysis by software agents or engineers, we would argue will improve the situation.

Future work includes completing the implementation of the system and evaluating the component with respective end users. The evaluation will involve users interacting with the system over a certain period and reporting usability issues. Furthermore, standardized metrics such as the Post Study System Usability Questionnaire (PSSUQ) [10] which will be used to evaluate user interaction in a quantitative metric. Moreover, the results of the evaluation will be analyzed, and improvements will be made to the component based on the analysis. Finally, the component could be expanded to handle other data formats such as JSON.

6. Acknowledgment

This research was conducted with the financial support of the SFI AI Centre for Research Training under Grant Agreement No. 18/CRT/6223 at the ADAPT SFI Research Centre (Grant # 13/RC/2106_P2) at Trinity College Dublin. A research collaboration with Ericsson Software Technology is helping to evaluate the proposed solution. With special thanks to John Keeney & Liam Fallon for their collaboration and support.

7. References

[1] Keith Alexander, Ltd Talis, Cyganiak, Michael Hausenblas, and Jun Zhao. 2010. Describing

Linked Datasets-On the Design and Usage of voiD, the'Vocabulary of Interlinked Datasets'. *Linked Data Web Work. (LDOW 09), conjunction with 18th Int. World Wide Web Conf. (WWW 09)* 538, (February 2010).

[2] Camila Bezerra, Fred Freitas, and Filipe Santana. 2013. Evaluating ontologies with Competency Questions. In *Proceedings - 2013 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Workshops, WI-IATW 2013*. DOI:https://doi.org/10.1109/WI-IAT.2013.199

[3] Mokrane Bouzeghoub. 2004. A framework for analysis of data freshness. In *Proceedings of the 2004 international workshop on Information quality in information systems*, 59–67.

[4] Victor Charpenay, Sebastian Käbisch, and Harald Kosch. 2016. Introducing Thing Descriptions and Interactions: An Ontology for the Web of Things. In *SR+ SWIT@ ISWC*, 55–66.

[5] Souripriya Das, Seema Sundara, and Richard Cyganiak. 2012. R2RML: RDB to RDF Mapping Language. *W3C Recomm.* (2012). DOI:https://doi.org/10.1017/CBO9781107415324.004

[6] Miguel Grinberg. 2018. *Flask web development: developing web applications with python*. " O'Reilly Media, Inc."

[7] Jens Hartmann, Y Sure, P Haase, R Palma, and M Suarez-Figueroa. 2005. OMV–ontology metadata vocabulary. In *ISWC*.

[8] Lalana Kagal and others. 2002. Rei: A policy language for the me-centric project. (2002).

[9] D Krech. 2006. Rdflib: A python library for working with rdf. *Online https://github. com/RDFLib/rdflib* (2006).

[10] James R. Lewis. 2002. Psychometric Evaluation of the PSSUQ Using Data from Five Years of Usability Studies. *Int. J. Hum. Comput. Interact.* 14, 3–4 (September 2002), 463–488. DOI:https://doi.org/10.1080/10447318.2002.9669130

[11] Raul Palma, Peter Haase, Oscar Corcho, and Asuncion Gomez-Perez. 2009. Change Representation For OWL 2 Ontologies.

[12] Alexandre Passant and Pablo N Mendes. *sparqlPuSH: Proactive notification of data updates in RDF stores using PubSubHubbub*. Retrieved from http://www.ldodds.com/blog/2010/04/rdf-dataset-notifications/

[13] Niko Popitsch and Bernhard Haslhofer. 2011. DSNotify - A solution for event detection and link maintenance in dynamic datasets. *J. Web Semant.* 9, 3 (September 2011), 266–283. DOI:https://doi.org/10.1016/j.websem.2011.05.002

[14] María Poveda-Villalón, Mari Carmen Suárez-Figueroa, and Asunción Gómez-Pérez. 2012. Validating ontologies with OOPS! In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 267–281. DOI:https://doi.org/10.1007/978-3-642-33876-2_24

[15] Alex Randles, Ademar Crotti Junior, and Declan O'Sullivan. 2020. Towards a vocabulary for mapping quality assessment. *Proc. 15th Int. Work. Ontol. Matching 19th Int. Semant. Web Conf. (ISWC)*, (2020).

[16] Alex Randles, Ademar Crotti Junior, and Declan O'Sullivan. 2021. A Vocabulary for Describing Mapping Quality Assessment, Refinement and Validation. In *2021 IEEE 15th International Conference on Semantic Computing (ICSC)*, 425–430. DOI:https://doi.org/10.1109/ICSC50631.2021.00076

[17] Alex Randles and Declan O'Sullivan. 2021. Assessing quality of R2RML mappings for OSi's Linked Open Data portal. *4th Int. Work. Geospatial Linked Data ESWC 2021* (2021).

[18] Alex Randles and Declan O'Sullivan. 2022. Evaluating Quality Improvement techniques within the Linked Data Generation Process. In *18th International Conference on Semantics Systems (SEMANTiCS)*.

[19] Alex Randles and Declan O'Sullivan. 2022. Change Detection Ontology (CDO) Specification. Retrieved from https://alex-randles.github.io/Change-Detection-Ontology/

[20] Alex Randles, Declan O'Sullivan, John Keeney, and Liam Fallon. 2022. Applying a Mapping Quality Framework in Cloud Native Monitoring. In *18th International Conference on Semantics Systems (SEMANTiCS)*.

[21] Julio Cesar Dos Reis, Cédric Pruski, Marcos Da Silveira, and Chantal Reynaud-Dela\^\itre. 2015. DyKOSMap: A framework for mapping adaptation between biomedical knowledge organization systems. *J. Biomed. Inform.* 55, (2015), 153–173.

[22] Ryan Shaw, Raphaël Troncy, and Lynda Hardman. 2009. Lode: Linking open descriptions of events. In *Asian semantic web conference*, 153–167.
[23] Anuj Singh, Rob Brennan, and Declan O'Sullivan. *DELTA-LD: A Change Detection Approach for Linked Datasets*.
[24] Jürgen Umbrich, B. Villazón-Terrazas, and M. Hausenblas. 2010. Dataset Dynamics Compendium: A Comparative Study. *undefined* (2010).
[25] J ¨ Urgen Umbrich, Michael Hausenblas, Aidan Hogan, Axel Polleres, and Stefan Decker. *Towards Dataset Dynamics: Change Frequency of Linked Open Data Sources*. Retrieved from http://code.google.com/p/pubsubhubbub/
[26] Protégé. Retrieved January 1, 2022 from https://protege.stanford.edu/
[27] Prometheus. Retrieved from https://prometheus.io/
[28] 2010. Dataset Dynamics (dady) Vocabulary. Retrieved June 14, 2022 from http://purl.org/NET/dady
[29] 2010. Talis Changeset Vocabulary. Retrieved from http://vocab.org/changeset/schema