

OGC Testbed-15

Semantic Web Link Builder and Triple Generator

Table of Contents

1. Subject	4
2. Executive Summary	5
2.1. Document contributor contact points	5
2.2. Foreword	5
3. References	7
4. Terms and definitions	8
4.1. Abbreviated terms	8
5. Overview	10
6. Introduction	11
6.1. Problem Definition	11
6.2. Status Quo	11
6.2.1. Lack of a Multi-Modal Data Integration Framework	11
6.2.2. Lack of a Unified, Extensible Logical Graph Model	12
6.2.3. Lack of Formal Semantics	12
6.2.4. Lack of Consistent Means for Recording and Exploiting Provenance and Pedigree Information (Source and Method)	12
7. Data Integration	13
7.1. Classifying Data: Structured, Unstructured, Semi-structured	13
7.2. Linked Data and Semantic Web	14
7.3. 5-Star Linked Open Data	15
7.4. Semantic Mediation	16
7.5. Deduplication	17
8. Data Fusion	21
8.1. Conflict Classification	21
8.2. Data Fusion Strategies and Answers	21
8.3. Data Fusion Answers	22
8.4. Conflict Resolution Functions	23
9. Shapes Constraint Language (SHACL)	26
9.1. Comparison of OWL and SHACL	27
9.2. SHACL Shapes	27
9.3. Constraint Components	30
9.3.1. Built-in Constraint Components	32
9.3.2. User-defined Constraint Components	33
9.3.3. Rules	34
9.3.4. Functions	35
9.3.5. Node Expressions	37
9.4. Application Profiles	38
9.4.1. Profiles Vocabulary	39

10. Data Sources	40
10.1. NRCAN Datasets	40
10.2. DBPedia	43
10.3. Wikidata	44
10.4. Geonames	45
10.5. OpenStreetMap	46
10.6. Freebase	47
11. Implementation	48
11.1. Architecture	48
11.1.1. Semantic Integration Pipeline	48
11.2. SHACL Engine	49
11.3. Correlation Ontology	50
11.3.1. LinkSetSpecification	50
11.3.2. CorrelationRuleSet	50
11.3.3. CorrelationRule	50
11.4. Similarity Ontology	51
11.4.1. Similarity Method	51
11.4.2. Comparison	52
11.4.3. Comparator	52
11.4.4. Aggregation	52
11.4.5. Aggregator	52
11.5. Metrics Ontology	53
11.5.1. Metric	53
11.5.2. Metric Type	54
11.5.3. Metric Unit	54
11.5.4. Example	55
11.6. Correlation Engine	59
11.7. Semantic Mediation Ontology	59
11.7.1. Alignment	60
11.7.2. Class Mapping	60
11.7.3. Property Mapping	60
11.8. REST API	61
11.9. WPS	65
11.10. Web Crawling	65
12. Future Work	67
12.1. Semantic Mediation Engine	67
12.2. Fusion Ontology and Fusion Engine	67
12.3. Integrated Fusion Pipeline	67
12.3.1. Fusion REST Service	67
12.3.2. Integration of Semantic Data Cubes with Conversational Agent	67
Appendix A: Appendix A	69

A.1. Metric Ontology	69
A.2. Similarity Ontology	70
A.3. Mediation Ontology	73
Appendix B: Revision History	77
Appendix C: Bibliography	78

Publication Date: 2019-12-17

Approval Date: 2019-11-22

Submission Date: 2019-10-31

Reference number of this document: OGC 19-021

Reference URL for this document: <http://www.opengis.net/doc/PER/t15-D001>

Category: OGC Public Engineering Report

Editor: Esther Kok, Stephane Fellah

Title: OGC Testbed-15: Semantic Web Link Builder and Triple Generator

OGC Public Engineering Report

COPYRIGHT

Copyright © 2019 Open Geospatial Consortium. To obtain additional rights of use, visit <http://www.opengeospatial.org/>

WARNING

This document is not an OGC Standard. This document is an OGC Public Engineering Report created as a deliverable in an OGC Interoperability Initiative and is not an official position of the OGC membership. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard. Further, any OGC Public Engineering Report should not be referenced as required or mandatory technology in procurements. However, the discussions in this document could very well lead to the definition of an OGC Standard.

LICENSE AGREEMENT

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD. THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER'S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR's sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to

indicate compliance with any LICENSOR standards or specifications.

This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

Chapter 1. Subject

This OGC Testbed 15 Engineering Report (ER) describes a generalized approach towards performing data fusion from multiple heterogeneous geospatial linked data sources. The specific use case is semantic enrichment of hydrographic features provided by Natural Resources Canada (NRCan). The ER attempts to define and formalize the integration pipeline necessary to perform a fusion process for producing semantically coherent fused entities.

Chapter 2. Executive Summary

The web and enterprise intranets have facilitated access to a vast amount of information. When data from multiple sources can be combined together, its usefulness increases dramatically. Users want to query information from different sources, combine it and present it into a uniform, complete, concise and coherent view using an information integration system. However, today there is no well-defined multi-modal data integration framework available. Such a framework can provide the user a *complete* yet *concise* and *coherent* overview of all existing data without the need to access each of the data sources separately. *Complete* because no object is forgotten in the result, *concise* because no object is represented twice, and *coherent* because the data presented to the user is without logical contradiction. Ensuring coherence is difficult because information about entities is stored in multiple sources and because of semantic heterogeneity.

A number of ontologies for supporting correlation and semantic mediation are defined using the new World Wide Web Consortium (W3C) Shape Constraint Language, as well as a correlation engine that has been implemented to be accessible through an Application Programming Interface (API) based on Representational State Transfer (REST). Future work will need to implement semantic mediation and fusion engine.

This engineering report makes the following recommendations for future work:

- The implementation of a semantic mediation engine supported by a mediation ontology.
- The formalization of different types of conflict and resolution strategies, enabled by ontologies.
- The demonstration of a complete fusion pipeline that includes semantic mapping, correlation, mediation and integration of entities from multiple sources defined in different ontologies.
- Extension of the REST-based Fusion Service API to support Create, Read, Update and Delete (CRUD) functions.
- Integration of Semantic Data Cubes with Conversational Agents.

2.1. Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Contacts

Name	Organization	Role
Esther Kok	Solenix	Editor/Contributor
Stephane Fellah	ImageMatters	Editor/Contributor
Nicola Policella	Solenix	Contributor

2.2. Foreword

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

Chapter 3. References

The following normative documents are referenced in this document.

NOTE: Only normative standards are referenced here, e.g. OGC, ISO or other SDO standards. All other references are listed in the bibliography.

- **OGC: OGC 06-121r9, OGC® Web Services Common Standard** [https://portal.opengeospatial.org/files/?artifact_id=38867&version=2]
- **OGC: OGC 09-026r2, OGC Filter Encoding 2.0 Encoding Standard - With Corrigendum, 2014** [<http://docs.opengeospatial.org/is/09-026r2/09-026r2.html>]
- **OGC: OGC 11-052r4, OGC GeoSPARQL- A Geographic Query Language for RDF Data, 2011** [https://portal.opengeospatial.org/files/?artifact_id=47664]
- **OGC: OGC 14-106. Unified Geo-data Reference Model for Law Enforcement and Public Safety, 2014** [<https://docs.opengeospatial.org/bp/14-106/14-106.html>]
- **W3C: RDF Schema 1.1, W3C Recommendation 25 February 2014** [<http://www.w3.org/TR/rdf-schema/>]
- **W3C: OWL 2 Web Ontology Language Document Overview (Second Edition), W3C Recommendation 11 December 2012** [<https://www.w3.org/TR/owl2-overview/>]
- **W3C: OWL 2 Web Ontology Language, Direct Semantics (Second Edition), W3C Recommendation 11 December 2012** [<http://www.w3.org/TR/2012/REC-owl2-direct-semantics-20121211/>]
- **W3C: OWL 2 Web Ontology Language, Structural Specification and Functional-Style Syntax (Second Edition), W3C Recommendation 11 December 2012** [<http://www.w3.org/TR/2012/REC-owl2-syntax-20121211/>]
- **W3C: SKOS Simple Knowledge Organization System Reference, W3C Recommendation 18 August 2009** [<http://www.w3.org/TR/2009/REC-skos-reference-20090818/>]
- **W3C: Data Catalog Vocabulary (DCAT), W3C Recommendation 16 January 2014** [<https://www.w3.org/TR/vocab-dcat/>]
- **W3C: SPARQL Protocol and RDF Query Language (SPARQL), last visited 12-09-2016** [<https://www.w3.org/TR/rdf-sparql-query>]
- **W3C: JSON-LD 1.1: A JSON-based Serialization for Linked Data - Candidate Recommendation, 2019** [<https://www.w3.org/TR/json-ld/>]
- **W3C: JSON-LD 1.0: A JSON-based Serialization for Linked Data - W3C Recommendation, 2014** [<https://www.w3.org/TR/2014/REC-json-ld-20140116/>]
- **ISO: ISO 19106:2004 Geographic information — Profiles** [<https://www.iso.org/standard/26011.html>]
- **ISO: ISO 19101-2:2018 Geographic information — Reference model — Part 2: Imagery** [<https://www.iso.org/standard/69325.html>]

Chapter 4. Terms and definitions

For the purposes of this report, the definitions specified in Clause 4 of the OWS Common Implementation Standard [OGC 06-121r9](https://portal.opengeospatial.org/files/?artifact_id=38867&version=2) [https://portal.opengeospatial.org/files/?artifact_id=38867&version=2] shall apply. In addition, the following terms and definitions apply.

• Unstructured Data

Traditional unstructured data is made of text documents and other file types such as videos, audios, images. Large amount of unstructured data is coming from sources external from enterprise such as social media. Natural language text is usually considered Unstructured Data.

• Semi-structured Data

Semi-structured Data is technically a subset of unstructured data and refers to tagged or taggable data that does not strictly follow a tabular or database record format. Examples include languages like XML, JSON and HTML.

• Structured Data

Structured data is "data that resides in fixed fields within a record or file". Examples include tables, spreadsheets, or databases (relational or NoSQL). Structured data is mostly understood today as data that conforms to a well-known schema (RDBMS schema, XML schema, JSON Schema). Schema defines the structure and syntactic constraints on the data.

• Triple

A Triple is the most high-level abstraction in the semantic web. It describes a statement using a triple of "Subject - Predicate - Object". URIs are used to identify the subject of the statement. The object of the statement can be another URI or a literal like a string or number. The triple model is a minimalist model to capture any form of data including table, tree, graph.

• Semantic Mediation

Semantic mediation is defined as the transformation from one conceptual model to another, in particular from one ontology to another. Instances of the target classes are created from the values of instances of the source classes.

4.1. Abbreviated terms

- API Application Programming Interface
- DCAT Data Catalog Vocabulary
- DCAT-AP DCAT Application Profile for Data Portals in Europe
- EARL Evaluation and Report Language EU European Union
- ER Engineering Report
- GeoDCAT-AP Geographical extension of DCAT-AP
- ISO International Organization for Standardization
- N3 Notation 3 format
- OGC Open Geospatial Consortium

- OWL Web Ontology Language
- RDF Resource Description Framework
- RDFS RDF Schema
- SHACL SHape Constraint Language
- SKOS Simple Knowledge Organization System
- SPARQL SPARQL Protocol and RDF Query URI Uniform Resource Identifier
- TTL Turtle Format
- URI Uniform Resource Identifier
- URL Uniform Resource Locator
- W3C World Wide Web Consortium
- XML eXtensible Markup Language

Chapter 5. Overview

[Section 6: Introduction](#) defines the problem of enriching geospatial data with semantic data using hydro features as a specific use case while also addressing the status quo of semantic enrichment.

[Section 7: Data Integration](#) discusses the topic of data integration, outlines the challenges and defines a formal approach for performing data fusion from multiple heterogeneous data sources.

[Section 8: SHACL](#) introduces the new W3C Shapes Constraint Language (SHACL) standard, a language for validating RDF graphs against a set of conditions, depicting shapes of graph data, and providing validation, transformation and inference rules. This ontology is used in the correlation and semantic mediation ontology and simplify previous mediation ontologies designed in previous OGC Testbeds 10, 11 and 12.

[Section 9: Data Sources](#) describes the main data sources that are used for the implementation of the integration pipeline which can perform data fusion from multiple heterogeneous geospatial linked data sources.

[Section 10: Implementation](#) presents the implementation of the integration pipeline and main work done in this Testbed thread. The section shows the architectural overview, the approach for correlation phase and semantic mediation and the REST API designed to support the correlation phase.

[Section 11: Future Work](#) addresses future work in this field.

[Appendix A](#) documents the ontologies developed during this Testbed.

Chapter 6. Introduction

This section defines the problem of enriching geospatial data with semantic data using hydrographic features as a specific use case. This section also addresses the status quo of semantic enrichment.

6.1. Problem Definition

Efforts in OGC Testbeds-11 [1] and 12 [2, 3] targeted semantic mediation support. The focus was exploring the high-level description of ontologies and the metadata needed to enable search on controlled vocabularies. As part of the Testbed-12 activity, a REST API was developed to access vocabulary metadata. This Semantic Registry Service was used as an aide in a first pass of the data in the further enrichment of an existing knowledge base.

The work described in this ER utilizes the vocabulary described in the Semantic Registry Information Model (SRIM) as an enabler for further knowledge base population. The SRIM is defined as a superset of the W3C DCAT standard and encoded as an OWL (Web Ontology Language) ontology. However, there should be restrictions in place (such as mandatory, recommended or optional fields) that cannot be captured with OWL, and require human interpretation. The Shape Constraint Language (SHACL) [4, 5] is a W3C standard that provides a framework to define the shape of the graph data. Proposed future work included investigation of SHACL shapes for its applicability to application profiles, form generation and data entry, data validation and quality control of linked data information.

The work described in this ER was focused on enriching geospatial structured data sources. The tools created in this Testbed harvest semantic data sources and attempt to correlate and fuse this information. The hydrological features for the Richelieu River/Watershed were the data source for this use case. One of the challenges in the work was to overcome mapping the ontology vocabulary in the Semantic Registry Information model to other ontologies used in geospatial databases. The alignment and "mapability" needs to be addressed to confirm the relation and/or add more information to the relation.

The high-level goal of the work described in this ER was to perform knowledge base population on the hydrological entities in the Richelieu River/Watershed. The ultimate goal was to see if the solution for this use case could be generalized for multiple problems.

6.2. Status Quo

This section describes the current challenges related to data integration and fusion.

6.2.1. Lack of a Multi-Modal Data Integration Framework

Inexpensive networks such as the web and enterprise intranets have facilitated the access to a vast amount of information. An information integration system can be utilized to combine and present data from multiple sources to increase the usefulness of the information. Unfortunately, today there is no well-defined multi-modal data integration framework available that can present a complete, concise, and coherent representation of the data to the user.

6.2.2. Lack of a Unified, Extensible Logical Graph Model

Current end-to-end data processing environments are plagued by inconsistent data models and formats that were developed in closed technology platform/program settings, are often proprietary, are not easy to extend, and are not suited for satisfying “Unified Knowledge Graph” needs. Such an eclectic approach leads to logical inconsistencies that must be addressed by human-intensive tools and processes, which impedes operational readiness and time-to-action. Logical inconsistencies can also render data useless. Moreover, the automation of graph workflows is severely inhibited by the lack of a well-known, standards-based logical graph model that works across many systems and organizations.

6.2.3. Lack of Formal Semantics

Current data processing environments do not formally model and explicitly represent semantics, nor do they address the semantic validity and consistency of their data and data models. This leads to ambiguity in most data, which increases the cognitive workload on users, severely inhibits machine-to-machine interoperability and understanding, and greatly reduces the potential for task/workflow automation (machine reasoning) and multi-source data integration.

6.2.4. Lack of Consistent Means for Recording and Exploiting Provenance and Pedigree Information (Source and Method)

Few systems formally capture and exploit the provenance and pedigree (P&P) of sources and methods. Moreover, rarely is P&P captured for all significant information elements, nor done consistently between systems. This leads to downstream processing impediments where users are unable to conflate or assess their confidence in data. In turn, this uncertainty inhibits the automation of data processing workflows, and increases the operating burden on users.

What is needed is a formal recording and tracking of P&P information about the sources and methods of all major information elements for all repository holdings. This is essential to database integrity management and answering questions users may have about the source and method for any information that is subsequently extracted, disseminated, and exploited. Formal recording and tracking of P&P information must be built into the database, and handled as a key function of database services.

Chapter 7. Data Integration

To improve the usefulness of data from different sources, an information integration system can be defined to present the data to the user. This section discusses the challenges of data integration and discusses an approach for performing data fusion from multiple heterogeneous data sources.

7.1. Classifying Data: Structured, Unstructured, Semi-structured

Before we delve into how to implement a robust integration system, it is important that we understand the different types of data that are available on the network. Data can be placed into three categories: unstructured, semi-structured, structured.

- **Unstructured Data** are data that are not so easily organized or formatted. Traditional unstructured data are made of text documents and other file types such as videos, audios, and images. Large volumes of unstructured data are coming from sources external from enterprises, such as social media, and constitute the vast majority of available data on the web (about 80 percent of all available data [6]). The volume increases every year. Collecting, processing, and analyzing unstructured data presents a significant challenge. With more information becoming available via the web, and most of which is unstructured, finding ways to use the data has become a vital strategy for many businesses. Recent advances in Deep Learning have addressed many challenges that seemed impossible to solve just 10 years ago. Deep Neural Networks are now capable of identifying objects in images and video scenes, robustly identify entities and relationships, and understand context and meaning in texts. (ex. BERT [7], OpenAI GPT-2 [8], RoBERTA [9]). While these types of data are out of scope for this Testbed, investigating the application of these latest advances in future Testbeds will help tap into the vast knowledge information buried in this information.
- **Semi-structured Data** is technically a subset of unstructured data and refers to tagged or taggable data that does not strictly follow a tabular or database record format. Examples include languages like XML, JSON and HTML.
- **Structured Data** is "data that resides in fixed fields within a record or file" (Webopedia). Examples include tables, spreadsheets, or databases (relational or NoSQL). Structured data is most understood today as data that conforms to a well-known schema (RDBMS schema, XML schema, JSON Schema). Schema defines the structure and syntactic constraints on the data.

While this categorization of data is well understood and widely adopted in the industry, it misses an important aspect: whether the data can be understood by machine-based algorithms or not. Understanding data means that computers can unambiguously interpret the meaning of information and be capable of inferring new information from data. This capability is the realm of Linked Data and the Semantic Web. Semantics are required for data and service interoperability. Semantics are also imperative for machine-to-machine understanding, reasoning (inference), and automation. Semantics greatly aid in search and navigation. They constitute the unifying means for interrelating heterogeneous data. They also aid in data abstraction, categorization, organization, and validation. Finally, semantics give data context and unambiguous meaning.

Many people use the term “data format” to suggest that they have a common, interoperable data

model. In the world of formal data modeling, this is considered to be flawed thinking that is fraught with system interoperability challenges. This thinking tends to surface in rushed stovepipe development efforts, and in settings where data modeling experts are absent. (System engineers and software developers who lack data modeling expertise notoriously skirt formal data models, and often misinterpret or vary from data standards.) For example, many developers make the simple mistake of treating a file format as their data model for interoperability. First of all, a (file) format is simply a convenient encoding for point-to-point data exchange purposes. A format is not a logical data model, per se, although it may have a formal logical data model behind it. Whereas a well-designed file format like Shapefile or KML may resolve schema and syntax issues, they do not explicitly deal with the semantics and context of the content it exchanges. Likewise, a collection of (file) formats is not a data model. File formats may actually hinder interoperability because each format requires custom mapping and translation software to another format, and the transformed results often do not align with sound, industry-approved coherent data models that were designed with interoperability in mind (as well exemplified by the work at OGC). Whereas file formats may be effective convenience mechanisms for point-to-point data exchanges, they are not good interoperability mechanisms for open service-based platforms and environments with many-to-many data exchange nodes. In summary, they tend to keep us from achieving the desired higher level of uniformity, logical consistency, and semantic harmony we seek for a System-of-Systems.

7.2. Linked Data and Semantic Web

In a famous article of Scientific American [10] Tim Berners-Lee described the aim of the Semantic Web bringing the web to its full potential. To enable wider adoption of the Semantic Web, the term of Linked Data was introduced in 2008 [11], which provides a simplified view of semantic web as a web of linkage between data nodes. The idea of linked data is similar to the web of hypertext, but the semantic web is not merely about publishing data on the web, but more about making links in such a way that a person or a machine can explore the data. The linked data leads to other related data. The semantic web is also constructed in such a way that it can be parsed and reasoned about. The web of hypertext is constructed with links anchored in HTML documents, but the semantic web is constructed in such a way that arbitrary links between entities are described by Triples in RDF. URIs are used to identify any kind of concept. [12]

The following describes some of the important concepts of the semantic web:

- **URI:** Uniform Resource Identifier is a sequence of characters that unambiguously identifies an abstract or physical resource. A URI can be used as a reference inside an RDF graph. Any resource describing any real-world or an abstract concept is identified by a URI, which is unambiguous and can be defined in a decentralized way (using domain ownership).
- **Triple:** A Triple is the most high-level abstraction in the semantic web. It describes a statement using a triple of "Subject - Predicate - Object". URI's are used to identify the subject of the statement. The object of the statement can be another URI or a literal like a string or number. The triple model is a minimalist model to capture any form of data including a table, tree, or graph.
- **RDF:** The Resource Description Framework (RDF) is a data model for representing any kind of information in the web (using the triple model). RDF is intended as a base notation for a range of extended notations such as OWL and RDF schema.
- **RDFS:** RDF Schema defines classes that represent concepts for the triples. RDFS captures

information about the types of relationships between facts and add meaning to the facts. This allows definition of sub-types of more general types.

- **OWL:** Web Ontology Language (OWL) is a Semantic Web language designed to represent rich and complex knowledge about things, groups of things, and relations between things. OWL adds semantics to RDF schema, also expressed in Triples. OWL allows for the deeper specification of the properties of classes, creating possibilities to not only join data from different sources as linked data, but also to define inferencing rules (transitive, symmetric, and inverse relationships for example).

The rise of the semantic web and linked open data has helped create a large number of available open data sources that could be explored. They could contain controlled vocabularies that structure and constrain the interpretation of the available data. Vocabularies can for example be defined as ontologies using RDF Schema and OWL, linked data with SPARQL, or as constraints that can be defined using SHACL. Ontologies define the meaning of the linked data, which can be modelled as a directed labelled graph. The nodes represent resources and the edges represent properties, assigned to the meaning in the ontology. Access to vocabularies is often provided at a SPARQL endpoint. SPARQL and its geospatial extension GeoSPARQL are now well-established standards for querying Linked Data representation that contains geospatial information.

SPARQL: SPARQL Protocol and RDF Query Language, the generic RDF query language.

Unfortunately, in practice a lot of resources and properties do not have resolvable resource URIs, so the exploration of other linked data is hindered. The principles of 5-star linked open data, which are described in the next subsection, aim to resolve some of the hindrance. [Figure 1](#) illustrates the 5-star principles.

7.3. 5-Star Linked Open Data

As a means to encourage individual users and governments to implement and improve on linked data [12] in 2010 Tim Berners-Lee developed a 5-star system for grading linked open data. Linked data is not necessarily linked open data, where the data is published under an open license. Linked data can also be 5-star (see [Figure 1](#)) within an internal system, but linked open data is data that follows the following graded scale:

1. **On the web:** available on the web (whatever format) (but with an open license in order to be Open Data)
2. **Machine-readable data:** available as machine-readable structured data
3. **Non-proprietary format:** as (2) plus non-proprietary format
4. **RDF standards:** all the above plus use of open standards from W3C (RDF and SPARQL) to be identifiable, so that others can link to the data
5. **Linked RDF:** all the above plus linking the data to other data to provide context

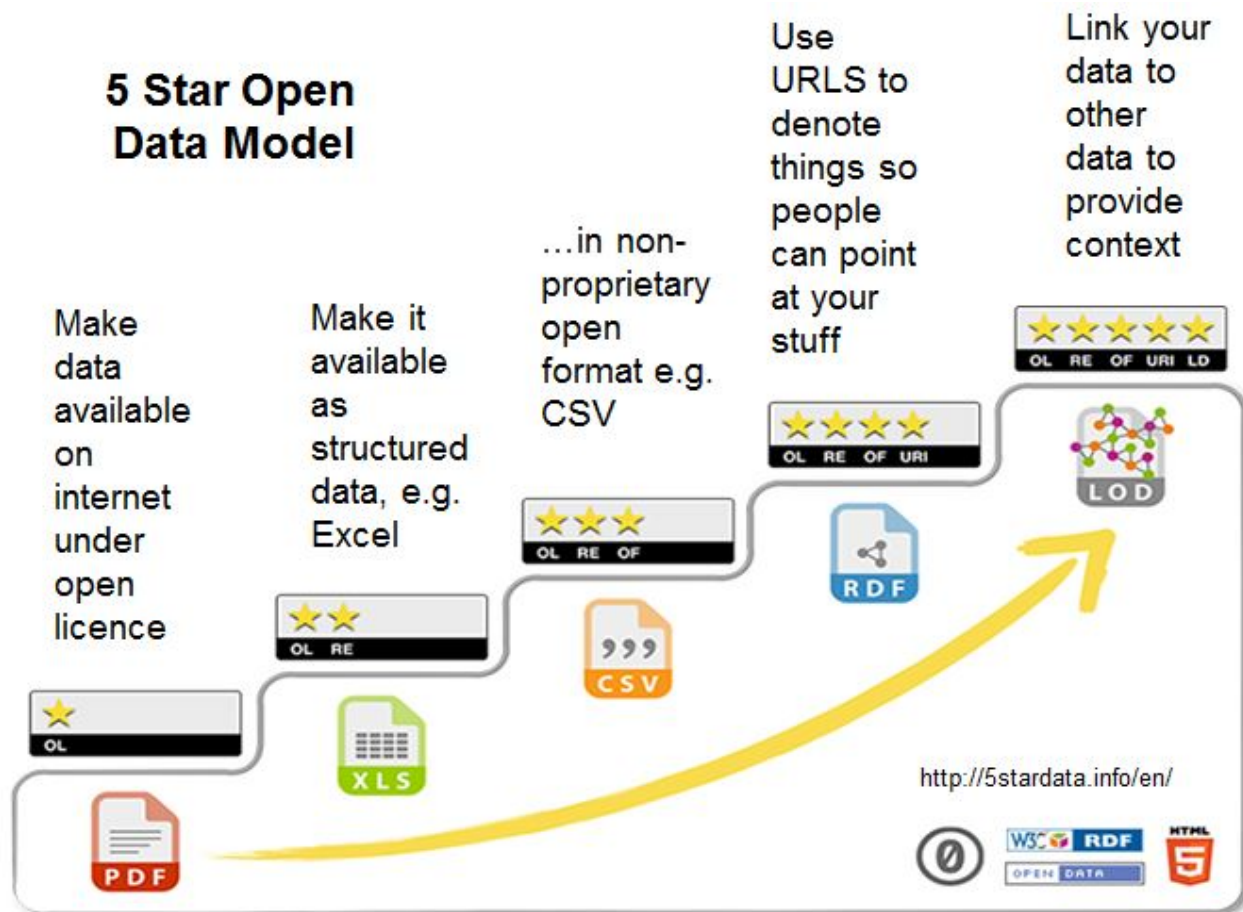


Figure 1. Five Star open data model

7.4. Semantic Mediation

Semantic mediation is defined as the transformation from one conceptual model to another, in particular from one ontology to another. Instances of the target classes are created from the values of instances of the source classes. Related work on this topic was documented in OGC Testbed-11 Symbology Mediation ER (OGC 15-058 [1]).

Semantic Mediation was addressed to some extent in OWS-8 [13], OWS-9 [14] and OWS-10 [15]. Those Testbeds were mostly focused on performing semantic mediation for taxonomies. For example, gazetteers such as the [Geographic Names Information System \(GNIS\)](https://www.usgs.gov/core-science-systems/ngp/board-on-geographic-names/download-gnis-data) [https://www.usgs.gov/core-science-systems/ngp/board-on-geographic-names/download-gnis-data], [GEOnet Names Server](http://geonames.nga.mil/gns/html/) [http://geonames.nga.mil/gns/html/], and [Geonames](http://geonames.org) [http://geonames.org] often use different taxonomies for classifying feature types. To support semantic mediation, mappings are required from one concept in a source taxonomy to another one in the target taxonomy (using SKOS mapping relationships such as `skos:exactMatch`, `skos:broadMatch`, `skos:closeMatch`). The semantic mediation was demonstrated using the OGC Web Feature Service-Gazetteer (WFS-G), however the mediation was performed as black box on syntactic representation of the features (using Geography Markup Language (GML)). In OWS-10, the hydrology sub-thread of CCI [15] attempted to address a more general approach for mediation by defining some mapping between two different hydrologic models. However, the solution was based on UML tools that perform the mapping and no formal model was defined to encode the semantic mapping. The mapping between different hydrological models was formalized through mediation by means of shared semantics. A two-step mapping approach was applied using the HY_Features model as mediating model. In the first step, the

hydrological models is mapped to common feature concepts of HY_Features. In the second step, the HY_Features model is “re-mapped” to the target models. To achieve semantic interoperability among the National Hydrography Dataset Plus (NHD)+ - and National Hydrographic Network (NHN) data models, two separate mappings are required: (1) mapping of NHD+ features to the equivalent concepts of HY_Features, and (2) mapping NHN features to HY_Features equivalents.

To address the semantic mediation of symbology, Testbed-11 addressed semantic mediation by representing information as linked data (see OGC 15-058). The goal of Testbed-11 was to formally address the semantic mediation for taxonomies by defining extensions to GeoSPARQL (geosparql:skosMatch) to perform the semantic mapping and providing an extensible, sharable encoding of the semantic mappings that can be processed by machine. A RESTful Semantic Mediation Service was demonstrated to perform semantic mapping between the Homeland Security Working Group (HSWG) Incident Model to the EMS Incident Model. For this purpose, existing linked data standards (RDF, OWL, SPARQL) were leveraged to represent semantic mappings. These semantic mappings can be managed by a semantic mediation service to perform transformation between two models. OGC 15-058 [1], Symbology Mediation Engineering Report from Testbed-11 describes the basic principles of semantic mediation using the example of two portrayal ontologies that need to be aligned in an ad hoc manner. In Testbed-12 (OGC 16-059 [2]), the semantic mediation work in Testbed-12 was closely related to the semantic portrayal work described above and built on the achievements from Testbed-11. Testbed-12 focused on the usage of a schema registry to store information about schemas and schema mappings to support ad hoc transformations between source and a target schemas. Schema mappings were considered as a simple form of semantic mediation, but were defined without explicit formalization of the underlying semantic knowledge required to map from one schema to another. For that reason, the idea was to design a Semantic Mediation Service REST API and integrate the API with the Semantic Registry and the CSW eBRIM profile for Schema Registry.

Since the conclusion of Testbed-13, W3C has standardized the SHACL specification [5]. The SHACL specification shows a lot of overlap with the work performed during Testbed-11 (see OGC 15-058). In particular, SHACL defines Shapes, Parameter, Function and Rules that have been defined in the semantic mapping ontology from Testbed-11. A comparison of SHACL with OWL was performed during Testbed-14 (OGC 18-094r1) and a model was defined to represent application profile for linked data. Within this context, an application profile defines a subset of concepts and properties from one or more ontologies to support mediation and in many use cases, semantic mediation is needed between two application profiles. W3C is currently conducting an effort to define application profiles for Linked data [16] and many application profiles such as DCAT-AP and GeoDCAT are encoded using the SHACL standard. One of the main efforts of Testbed-15 was to update the mapping ontology by using the SHACL standard. By doing so, the semantic mediation can be considerably simplified and be usable with emerging SHACL APIs. This would favor interoperability and sharing of semantic alignments.

7.5. Deduplication

When integrating multiple linked data sources, equivalent entities may be identified with different identifiers. The goal of the deduplication phase is to identify entities that are similar semantically in order for them to be merged into one single entity in the next phase: semantic fusion. To perform this task, the rules of linkage between two entities that are similar need to be defined.

The literature suggests a number of Linked Data frameworks addressing this task (also called Linked Discovery). Silk - Link Discovery Framework [17] uses the declarative Silk - Link Specification Language (Silk-LSL) defined in XML and a proprietary path syntax to define the linkage rules. Data publishers can specify which types of RDF links should be discovered between data sources as well as which conditions data items must fulfill in order to be interlinked. These link conditions can apply different similarity metrics to multiple properties of an entity or related entities which are addressed using a path-based selector language. The resulting similarity scores can be weighted and combined using various similarity aggregation functions.

Silk accesses data sources via the SPARQL protocol and can thus be used to discover links between local and remote data sources. SILK uses a multi-dimensional blocking technique (MultiBlock [18]) to optimize the linking runtime through a rough index pre-matching. To parallelize the linking process, SILK relies on MapReduce. The framework allows user-specified link types between resources as well as owl:sameAs links. SILK incorporates element-level matchers on selected properties using string, numeric, temporal and geo-spatial similarity measures. SILK also supports multiple matchers, as it allows the comparison of different properties between resources that are combined together using match rules. SILK also implements supervised and active learning methods for identifying Link Specifications (LS) for linking. One of the shortcomings of the framework is that it uses a proprietary document-based format for configuring linkset specification and custom path syntax. The model does not provide a mechanism for defining additional constraints on the nodes referred by the paths. The SILK framework comes with a workbench which allows to define the link set specification in a visual way (see Figure 2).

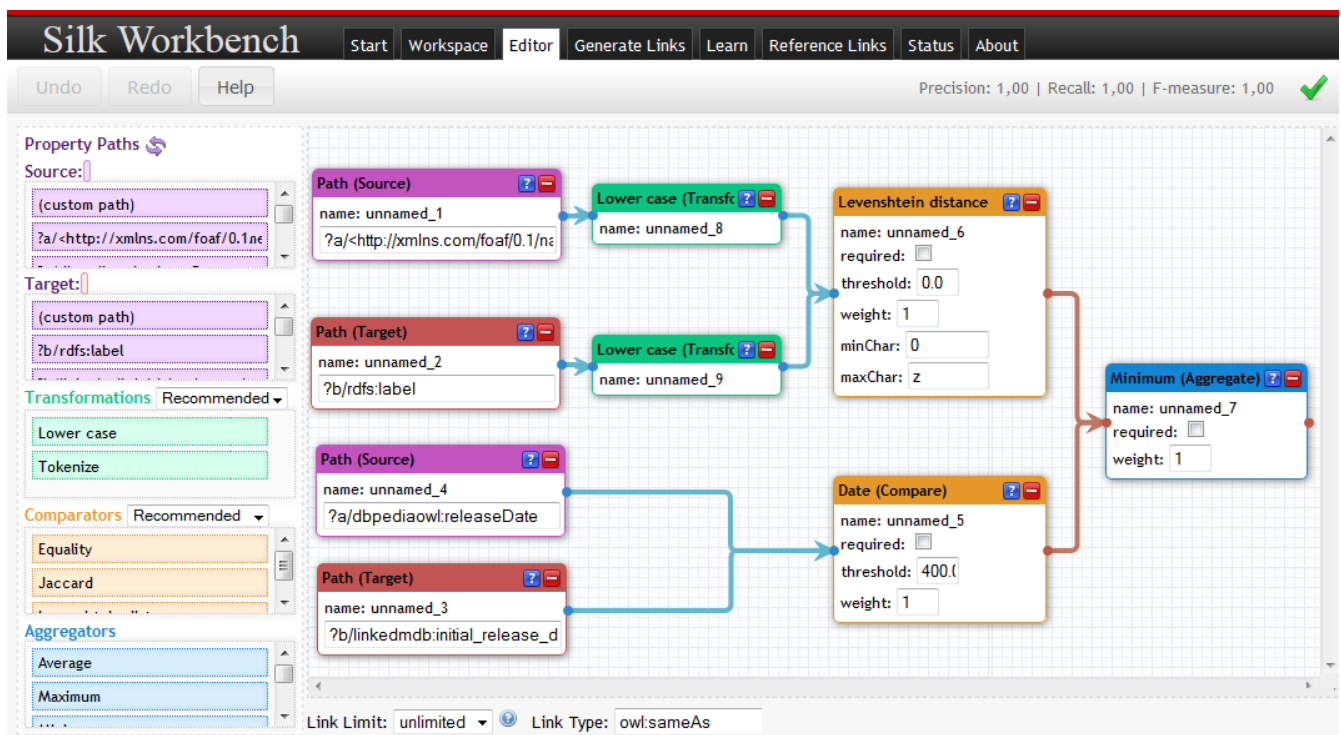


Figure 2. SILK Workbench

Another, and more recent framework, is the Linked discovery framework for MEtric Spaces (LIMES). This framework implements time-efficient approaches for large-scale link discovery based on the characteristics of metric spaces. [19] It is easily configurable via a configuration file as well as through a graphical user interface. According to its web site: [20]

"LIMES implements novel time-efficient approaches for link discovery in metric spaces. Our

approaches facilitate different approximation techniques to compute estimates of the similarity between instances. These estimates are then used to filter out a large amount of those instance pairs that do not suffice the mapping conditions. By these means, LIMES can reduce the number of comparisons needed during the mapping process by several orders of magnitude. The approaches implemented in LIMES include the original LIMES algorithm for edit distances, HR3 [21], HYpersphere aPPrOximation algorithm (HYPPPO) [22], and ORCHID [23]. Additionally, LIMES supports the first planning technique for link discovery HELIOS [24], that minimizes the overall execution of a link specification, without any loss of completeness. Moreover, LIMES implements supervised and unsupervised machine-learning algorithms for finding accurate link specifications. The algorithms implemented here include the supervised, active and unsupervised versions of EAGLE [25] and WOMBAT [26].

The LIMES framework consists of eight main modules of which each can be extended to accommodate new or improved functionality. The central modules of LIMES is the controller module, which coordinates the matching process. The matching process is carried out as follows: First, the controller calls the configuration module, which reads the configuration file and extracts all the information necessary to carry out the comparison of instances, including the URL of the SPARQL-endpoints of the knowledge bases S (source) and T(target), the restrictions on the instances to map (e.g., their type), the expression of the metric to be used and the threshold to be used.

Given that the configuration file is valid w.r.t. the LIMES Specification Language (LSL), the query module is called. This module uses the configuration for the target and source knowledge bases to retrieve instances and properties from the SPARQL-endpoints of the source and target knowledge bases that adhere to the restrictions specified in the configuration file. The query module writes its output into a file by invoking the cache module. Once all instances have been stored in the cache, the controller chooses between performing Link Discovery or Machine Learning. For Link Discovery, LIMES will re-write, plan and execute the Link Specification (LS) included in the configuration file, by calling the rewriter, planner and engine modules resp. The main goal of Linked Discovery is to identify the set of links (mapping) that satisfy the conditions opposed by the input LS. For Machine Learning, LIMES calls the machine learning algorithm included in the configuration file, to identify an appropriate LS to link S and T. Then it proceeds in executing the LS. For both tasks, the mapping will be stored in the output file chosen by the user in the configuration file. The results are finally stored into an RDF or an XML file. ""

Like the SILK framework, the configuration of LIMES is defined with a proprietary XML format and syntax for expressing rules as illustrated in [Figure 3](#)

Examples (toggle)
Drugbank
Diseases
SimCities



Download:
Manual | Distribution

Source:	Target:
Endpoint: <input type="text" value="http://mesh.bio2rdf.org/sparql"/>	Endpoint: <input type="text" value="http://data.linkedct.org/sparql"/>
Var: <input type="text" value="?y"/>	Var: <input type="text" value="?x"/>
Pagesize: <input type="text" value="1000"/>	Pagesize: <input type="text" value="1000"/>
Restriction: <input type="text" value="?y rdf:type meshr:Concept"/>	Restriction: <input type="text" value="?x rdf:type linkedct:condition"/>
Property: <input type="text" value="dc:title"/>	Property: <input type="text" value="linkedct:condition_name"/>
Metric: <input type="text" value="levenshtein(y.dc.title, x.linkedct.condition_name)"/>	
Exemplars: <input type="text"/>	
Acceptance:	Review:
Threshold: <input type="text" value="0.98"/>	Threshold: <input type="text" value="0.95"/>
Relation: <input type="text" value="owl:sameAs"/>	Relation: <input type="text" value="owl:sameAs"/>

Detected prefixes:
rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns# (-)
rdfs: http://www.w3.org/2000/01/rdf-schema# (-)
owl: http://www.w3.org/2002/07/owl# (-)
linkedct: http://data.linkedct.org/resource/linkedct/ (-)
dc: http://purl.org/dc/terms/ (-)
meshr: http://bio2rdf.org/ns/meshr# (-)

Figure 3. LIMES Workbench

The Testbed-15 approach to model correlation is based on Open Linked Data Standards (RDF, SHACL, OWL, SPARQL). The rationale is to be able to favor reusability of the correlation rules between two linked data sets, by leveraging the linking properties that is built-in in the linked data framework by referring to URI identifier. Three ontologies were formalized to model metrics [Metrics Ontology](#), similarity [Similarity Ontology](#) and correlation [Correlation Ontology](#) to support correlation task using a correlation engine implemented for this testbed.

Chapter 8. Data Fusion

In the context of data integration, Data Fusion is defined as the “process of fusing multiple records representing the same real-world object into a single, consistent, and clean representation [27].

8.1. Conflict Classification

Due to the decentralized nature of the web, different communities can represent the same-real world objects in different way. This results in conflicts. We can distinguish three type of conflict [27].

- **Schematic conflicts:** Such as, different attribute names or differently structured data sources.
- **Identity conflicts:** As the way of identifying a real-world object may be different in the data sources.
- **Data conflicts:** For the same real-world object (e.g., a building), semantically equivalent attributes, from one or more sources, do not agree on its attribute value (e.g., source 1 reporting “23” as the building’s age, source 2 reporting “25”).

The first two kinds of conflict are resolved during the semantic mediation and correlation phase of data integration. The third kind of conflict, data conflicts, remain and are not resolved until data fusion and are caused by the remaining multiple representations of same real-world objects.

We distinguish two kinds of data conflict: (a) uncertainty about the attribute value, caused by missing information; and (b) contradictions, caused by different attribute values [27]:

- **Uncertainties:** An uncertainty is a conflict between a non-null value and one or more null values that are all used to describe the same property of an object. In the Testbed-15 scenario, this is caused by missing information, such as null values in the sources, or an attribute completely missing in one source. The reason for considering uncertainties as a special case of conflict is that they are generally easier to cope with than contradictions. We deliberately choose to assume most null values in a data integration scenario being unknown values. But even with considering null values as being inapplicable or withheld, as are the three most common semantics of null values [28], the assessment of the different techniques and systems remains valid.
- **Contradictions:** A contradiction is a conflict between two or more different non-null values that are all used to describe the same property of the same object. In the Testbed-15 data integration scenario, this is the case if two or more data sources provide two or more different values for the same attribute on the same object, sameness as given by the schema matching and duplicate detection steps before.

8.2. Data Fusion Strategies and Answers

We can distinguish three categories of conflict strategies:

- **Conflict-ignoring strategies** do not make a decision as to what to do with conflicting data and sometimes are not even aware of data conflicts. An example for an ignoring strategy is Pass It

On, which presents all values and that way defers conflict resolution to the user.

- **Conflict-avoiding strategies** acknowledge the existence of possible conflicts in general, but do not detect and resolve single existing conflicts. Instead, they handle conflicting data by applying a unique decision equally to all data, such as preferring data from a special source with the Trust Your Friends strategy.
- **Conflict resolution strategies** do regard all the data and metadata before deciding on how to resolve a conflict. They can further be subdivided into deciding and mediating strategies, depending on whether they choose a value from all the already present values (deciding) or choose a value that does not necessarily exist among the conflicting values (mediating).

Bleiholder et al (2005) [29] formalize some possible conflict handling strategies summarized in [Table 1](#)

Table 1. Possible Conflict Handling Strategies from Bleiholder et al (2005)[19]

Strategy	Classification	Short Description
PASS IT ON	ignoring	escalates conflicts to user or application
CONSIDER ALL POSSIBILITIES	ignoring	creates all possible value combinations
TAKE THE INFORMATION	avoiding, instance based	prefers values over null values
NO GOSSIPING	avoiding, instance based	returns only consistent tuples
TRUST YOUR FRIENDS	avoiding, metadata based	takes the value of a preferred source
CRY WITH THE WOLVES	resolution, instance based, deciding	takes the most often occurring value
ROLL THE DICE	resolution, instance based, deciding	takes a random value
MEET IN THE MIDDLE	resolution, instance based, mediating	takes an average value
KEEP UP TO DATE	resolution, metadata based, deciding	takes the most recent value

8.3. Data Fusion Answers

The fusion result to an integrated information system should have the following characteristics:

- **Complete:** A complete answer contains all the objects (extensionally complete) and also all attributes (intentionally complete) that have been present in the sources. A complete answer is not necessarily concise, as it may contain objects or attributes more than just once.
- **Concise:** An answer is concise if all real-world objects (extensionally concise) and all semantically equivalent attributes (intentionally concise) present are described only once.
- **Consistent:** A consistent answer contains all tuples from the sources that are consistent with respect to a specified set of integrity constraints (inclusion or functional dependencies) [30]. In this sense, such an answer is not necessarily complete, as all inconsistent object representations

are left out of the result. However, given that one of the integrity constraints is a key constraint on some real-world identifier, a consistent answer is extensionally concise for all included object representations.

- **Complete and Consistent:** A complete and consistent answer combines the advantages of completeness and conciseness and consists of all real-world object descriptions

8.4. Conflict Resolution Functions

The following table summarizes a number conflict resolution functions found in the literature [29].

Table 2. Conflict Resolution Functions with Type (*S*(ingle), *SP*(arameter), *M*(ultiple)*P*, *SMP*), Domain (*A*(ll), *N*(umeric), *C*(ategorical), *T*(axonomical), *D*(ate), *S*(tring)) and *D*(eciding) or *M*(ediating).

Function	Description	Type	Domain	D or M
<i>All</i>	Returns all values.	S	A	D
<i>Any</i>	Returns an arbitrary (non-NULL) value.	S	A	D
<i>First</i>	Returns the first (non-NULL) value, respectively. Requires ordering of the values on input.	S	A	D
<i>Last</i>	Returns the last (non-NULL) value, respectively. Requires ordering of the values on input.	S	A	D
<i>Random</i>	Returns a random (non-NULL) value. The chosen value differs among calls on the same input.	S	A	D
<i>Certain</i>	If input values contain only one distinct (non-NULL) value, returns it. Otherwise returns NULL or empty output (depending on the underlying data model).	S	A	D
<i>Best</i>	Returns the value with the highest data quality value. The quality measure is application-specific.	SP	A	D
<i>TopN</i>	Returns n best values (see Best). n is a parameter.	S	A	D
<i>Threshold</i>	Returns values with data quality higher than a given threshold. The threshold is given as a parameter.	SP	ND	D
<i>BestSource</i>	Returns a value from the most preferred source. The preference of source may be explicit (given preferred order of sources) or based on an underlying data quality model.	SP	A	D
<i>MaxSourceMetadata</i>	Returns a value from the source with a maximal source metadata value. The metadata value may be, e.g., timestamp of the source, access cost or a data quality indicator. The used type of source metadata is either given as a parameter or fixed.	SP	A	D
<i>MinSourceMetadata</i>	Returns a value from the source with the minimal source metadata value (see MaxSourceMetadata).	SP	A	D
<i>Latest</i>	Returns the most recent (non-NULL) value. Recency may be available from another property, value/entity metadata or source metadata (the last case is a special case of MaxSourceMetadata).	SP	A	D

Function	Description	Type	Domain	D or M
<i>ChooseSource</i>	Returns a value originating from the source given as a parameter.	SP	A	D
<i>Vote</i>	Returns the most-frequently occurring (non-NULL) value. Different strategies may be employed in case of tie, e.g., choosing the first or a random value.	S	A	D
<i>WeightedVote</i>	Same as <i>Vote</i> but each occurrence of a value is weighted by the quality of its source.	SP	A	D
<i>Longest</i>	Returns the longest (non-NULL) value.	S	SCT	D
<i>Shortest</i>	Returns the shortest (non-NULL) value.	S	SCT	D
<i>Min</i>	Returns the minimal (non-NULL) value according to an ordering of input values.	S	SND	M
<i>Max</i>	Returns the maximal (non-NULL) value according to an ordering of input values.	S	SND	M
<i>Filter</i>	Returns values within a given range. The minimum and/or maximum are given as parameters.	SP	SND	D
<i>MostGeneral</i>	Returns the most general value according to a taxonomy or ontology.	SP	T	D
<i>MostSpecific</i>	Returns the most specific value, according to a taxonomy or ontology (if the values are on a common path in the taxonomy).	SP	T	D
<i>Concat</i>	Returns a concatenation of all values. The separator of values may be given as a parameter. Annotations such as source identifiers may be added to the result.	S	A	M
<i>Constant</i>	Returns a constant value. The constant may be given as a parameter or be fixed (e.g. NULL).	SP	A	M
<i>CommonBeginning</i>	Returns the common substring at the beginning of conflicting values.	S	S	M
<i>CommonEnding</i>	Returns the common substring at the end of conflicting values.	S	S	M
<i>TokenUnion</i>	Tokenizes the conflicting values and returns the union of the tokens.	SP	S	M
<i>TokenIntersection</i>	Tokenizes the conflicting values and returns the intersection of the tokens.	SP	S	M
<i>Avg</i>	Returns the average of all (non-NULL) input values.	S	N	M
<i>Median</i>	Returns the median of all (non-NULL) input values.	S	N	M
<i>Sum</i>	Returns the sum of all (non-NULL) input values.	S	N	M
<i>Count</i>	Returns the number of distinct (non-NULL) values.	S	A	M
<i>Variance, StdDev</i>	Returns the variance or standard deviation of values, respectively.	S	N	M

Function	Description	Type	Domain	D or M
<i>ChooseCorresponding</i>	Returns the value that belongs to an entity (resource) whose value has already been chosen for a property A, where A is given as a parameter.	MP	A	D
<i>ChooseDepending</i>	Returns the value that belongs to an entity (resource) which has a value v of an property A, where v and A are given as parameters.	MP	A	D
<i>MostComplete</i>	Returns the (non-NULL) value from the source having fewest NULLs for the respective property across all entities.	SP	A	D
<i>MostDistinguishing</i>	Returns the most distinguishing value among all present values for the respective property.	SP	A	D
<i>Lookup</i>	Returns a value by doing a lookup into the source given as a parameter, using the input values.	SP	A	M
<i>MostActive</i>	Returns the most often accessed or used value.	SP	A	D
<i>GlobalVote</i>	Returns the most-frequently occurring (nonNULL) value for the respective property among all entities in the data source.	S	A	D
<i>Coalesce</i>	Takes the first non-null value appearing.	S	A	D
<i>Group</i>	Returns a set of all conflicting values. Leaves resolution to the user.	S	A	x
<i>Highest Quality</i>	Evaluates to the value of highest information quality, requiring an underlying quality model.	SP	A	D
<i>Most Recent</i>	Takes the most recent value. Most recentness is evaluated with the help of another property or other data about recentness of tuples/values.	SMP	A	D
<i>Most Active</i>	Returns the most often accessed or used value. Usage statistics of the knowledge base can be used in evaluating this function.	S	A	D
<i>Choose Corresponding</i>	Chooses the value that belongs to the value chosen for another column.	S	A	D
<i>Most complete</i>	Returns the value from the source that contains the fewest null values in the attribute in question.	S	A	D
<i>Most distinguishing</i>	Returns the value that is the most distinguishing among all present values in that property.	S	A	D
<i>Highest information value</i>	According to an information measure this function returns the value with the highest information value.	S	A	D

This list of functions could be used as a starting point to design an ontology for fusion which defines conflict resolution function (may be based on SHACL functions). Unfortunately, due to time constraints, this would have to be investigated in future Testbeds.

Chapter 9. Shapes Constraint Language (SHACL)

As SHACL is used in the ontologies defined for this Testbed to support a knowledge fusion pipeline, this section provides an overview of the W3C standard Shape Constraint Language (SHACL).

Work in previous Testbeds [31] utilized RDF and OWL to define sets of classes and properties that could be reused by a large number of external vocabularies. RDF Schema is a vocabulary for expressing classes and their properties, as well as associations of properties with classes. OWL is an extension of RDF Schema to express restrictions. However, providing restrictions and constraints in the ontology itself, and enforcing them cannot be captured with these technologies. Providing restrictions required the user or developer to read the documentation and implement the constraints in code.

The Shape Constraint Language (SHACL) is a W3C standard vocabulary for describing and validating RDF graph structures. These graph structures are captured as "shapes", which correspond to nodes in RDF graphs. These shapes identify predicates and their associated cardinalities, and datatypes. SHACL shapes can be used to communicate data structures associated with a process or interface, to generate or validate data, or to drive user interfaces. The vocabulary allows for well-defined and complex integrity constraints defined in RDF and SPARQL/JavaScript constraints. SHACL is not a replacement for RDFS/OWL, but a complementary technology that is not only very expressive but also highly extensible.

Both OWL and SHACL rely on RDF Schema to define vocabulary terms (classes/properties) and their hierarchies (subclasses, sub-properties). The property constraints (cardinality, valid values, etc.) can be captured using SHACL. SHACL can accommodate multiple profiles by providing different shapes for the same ontology.

The SHACL vocabulary is not only defined in RDF itself, but the same macro mechanisms can be used by anyone to define new high-level language elements and publish them on the web. This means that SHACL will not only lead to the reuse of data schemas but also to domain-specific constraint languages. Further, SHACL can be used in conjunction with a variety of languages beside SPARQL, including JavaScript. Complex validation constraints can be expressed in JavaScript so that they can be evaluated client-side. In addition, SHACL can be used to generate validation report for quality control with potentially suggestions to fix validation errors. Overall, SHACL is a future-proof schema language designed for the Web of Data.

In summary, features of SHACL include:

- RDF vocabulary used to express shapes, targets and constraints.
- Constraints can be expressed in extension languages like SPARQL.
- SHACL shapes can be mixed with other semantic web data that is compatible with RDF and linked data principles.
- SHACL definitions can be serialized in multiple RDF formats.

9.1. Comparison of OWL and SHACL

There is a fundamental difference in the interpretation of OWL restrictions versus SHACL constraints. OWL is designed for inferencing, meaning that the interpretation of restrictions leads to OWL making assumptions and inferences about the data. OWL is based on an Open-World Assumption [4], where absent statements and properties can be filled later, and absence of data does not invalidate the statement. The application should infer the missing data. Violations in for example cardinality mean that the OWL processor will assume the violating values must represent the same entity with different URIs. This is because OWL makes a distinction between URI and real-world entity where multiple URIs can represent the same entity. This is also referred to as the Unique-Name Assumption. [4]

SHACL is designed based on a Closed-World Assumption [4], where lack of data invalidates the statement, meaning that the interpretation of restrictions is based on the assumption that the knowledge base is complete or can be assumed to be as complete as possible with the current information. Aside from the built-in constraints, SHACL constraints can be expressed using SPARQL or in JavaScript, making it highly flexible and extensible. In contrast to OWL, where limited data validation is done via inferencing, SHACL separates checking data validity from reasoning and inferring new facts. OWL's built-in nature of the Open World Assumption and the Unique Name Assumption contradicts established approaches from schema languages and makes the meaning of certain statements (e.g., cardinality) different from what most modelers expect. [4]

Due to the differences in design philosophy and implementation, one of the main advantages of SHACL over OWL is that SHACL is extensible while OWL is limited to the features as defined by the OWL committee. In the end, the SHACL vocabulary is complementary to OWL, but has a higher usability. The usability increase can be experienced through the SHACL shapes in the definition of constraints and constraint targets, as well as the built-in constraint types.

9.2. SHACL Shapes

When the SHACL standard was released by the W3C in 2017, it introduced the concept of shapes. SHACL shapes are described in terms of RDF graphs, which is then referred to as a shapes graph, and follow a hierarchy of shapes based on the RDF Schema language. SHACL aims to validate RDF graphs against the shapes, where the data being validated is called a data graph.

There are two types of shape: *node shape* that declare constraints directly on a node and *property shape* that declare constraints on the values associated with a node through a path. Node shapes declare constraints directly on a node e.g., node kind (Internationalized Resource Identifier (IRI), literal or blank), IRI regex, etc. Property shapes declare constraints on the values associated with a node through a path, e.g., constraints about a certain ongoing or incoming property of a focus node; cardinality, datatype, numeric min/max, etc. (see [Figure 4](#))

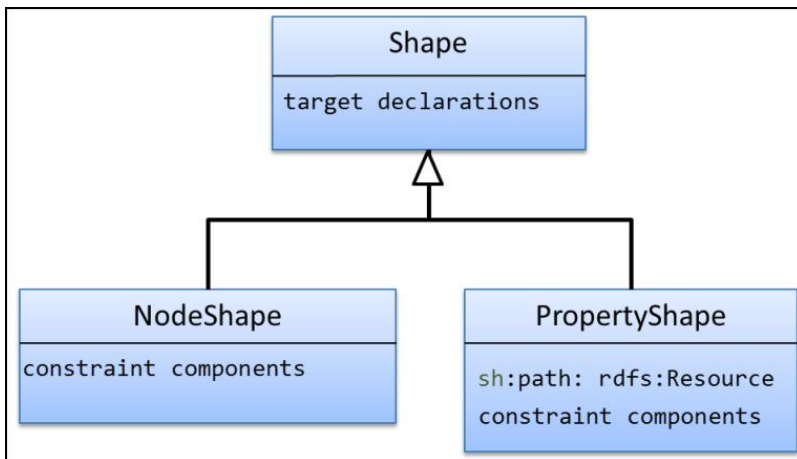


Figure 4. SHACL Shape Hierarchy

SHACL shapes may define several target declarations. Target declarations specify the set of nodes that will be validated against a shape, e.g., directly pointing to a node, or all nodes that are subjects to a certain predicate, etc. The target declarations of a shape in a shapes graph are triples with the shape as the subject and one of **sh:targetNode**, **sh:targetClass**, **sh:targetObjectsOf** or **sh:targetSubjectsOf** as a predicate.

Shapes also declare constraints which are constraints on the focus nodes and value nodes of their properties. **Constraints** to be applied to the target, e.g. cardinalities, ranges of values, data types, property pairs, etc. They can also declare rules that can be used to add inferences or perform mapping from one model to another (see [Figure 5](#)).

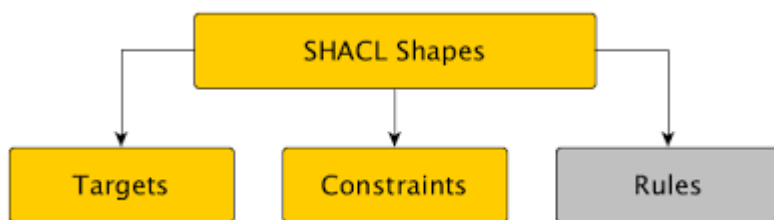


Figure 5. SHACL Architecture Overview

A more detailed description of the SHACL model is shown in [Figure 6](#). The SHACL specification has a number of extensions that enable expanding the expressiveness of the validation. The SHACL-SPARQL extension provides mechanism to add Constraint Component plugins and Rules based on the standard SPARQL. The SHACL advanced features provides additional mechanism to add rules for inferencing and transformation. SHACL-JS provides JavaScript extensions for SHACL, which could implement functions for constraint component validators, target selections and rules.

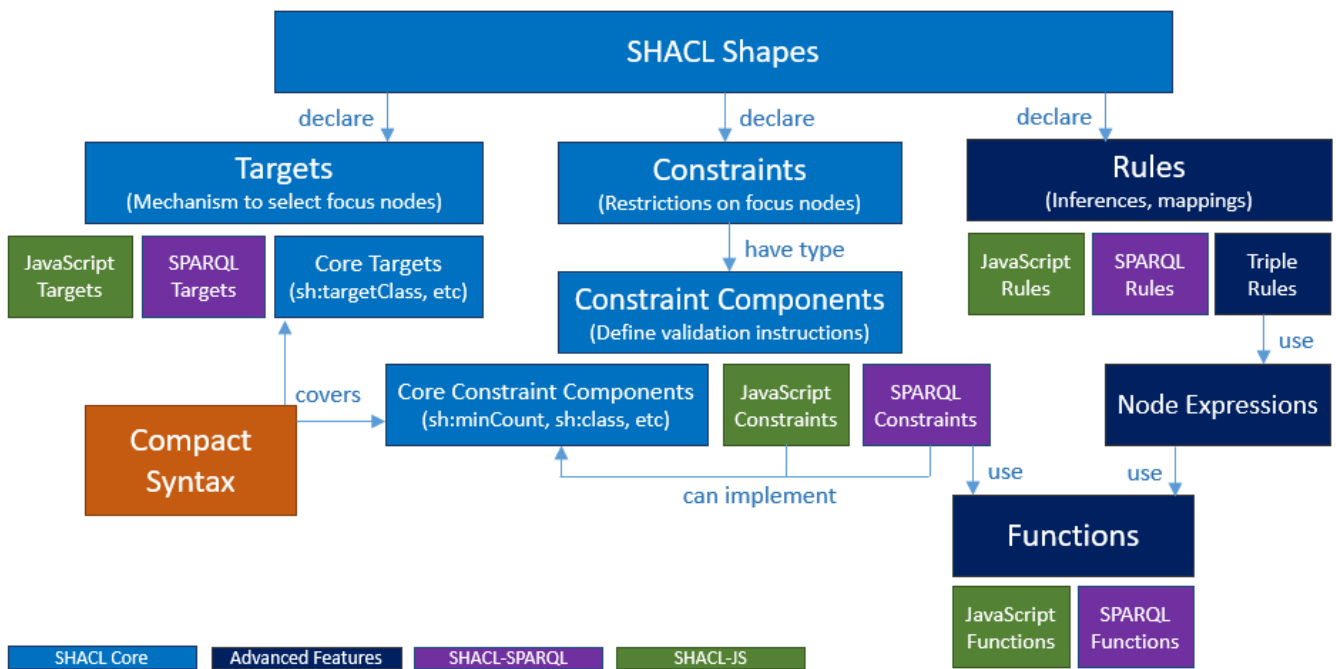


Figure 6. SHACL Detailed Architecture

RDF terms produced by targets are not required to exist as nodes in the data graph. Targets of a shape are ignored whenever a **focus node** is provided directly as input to the validation process for that shape. A focus node is an RDF term that is validated against a shape using the triples from a data graph.

An RDF graph uses namespaces to anchor to the appropriate (ontology) vocabularies, which can be stored in industry standard formats like Turtle, JSON-LD and RDF/XML. SHACL itself is defined as part of a namespace as well. There are a number of standard namespace prefixes that can be encountered as part of the shapes definition:

Table 3. Namespace Prefix Bindings

Prefix	Namespace
rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs:	http://www.w3.org/2000/01/rdf-schema#
sh:	http://www.w3.org/ns/shacl#
xsd:	http://www.w3.org/2001/XMLSchema#

While SHACL is defined using OWL, there was no UML diagrams available for the specification. To support the implementation of the SHACL engine, a UML Object Model compliant with the SHACL ontology has been defined as part of this work. The model may not capture all the nuances of the specification. However, the model was designed to have share a common understanding. The model should not be considered normative, but only informative. The model provides extensions points for Functions, Rules, Constraint Components and TargetTypes. The built-in and most commonly used constraint components can be directly set to the shapes to simplify their encodings. An overview of the model is shown in Figure 7.

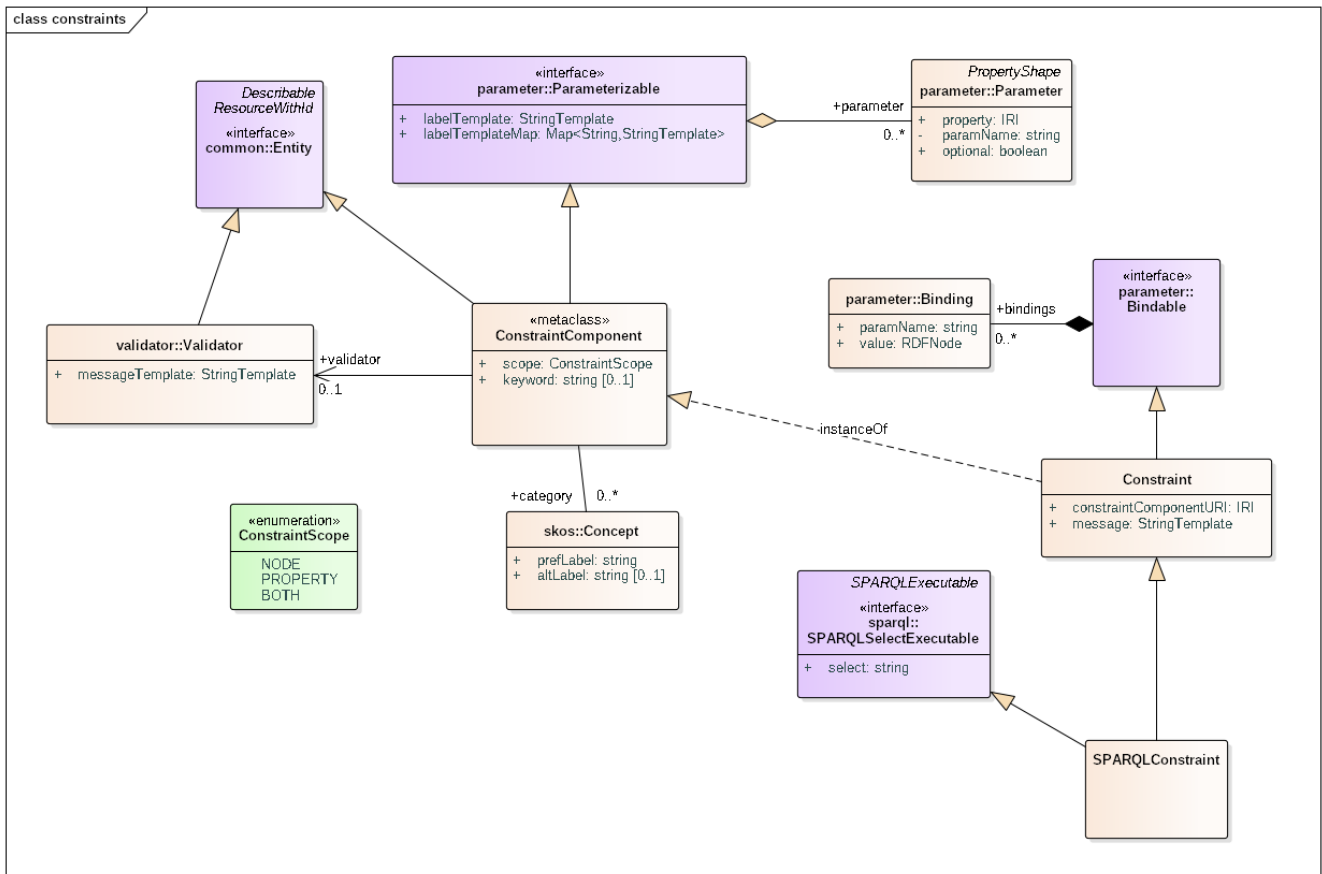


Figure 8. Constraint Model

Constraint components are associated with *validators*, (see Figure 9), which define the behavior of the constraint. Writing custom Constraint Component is considered as an advanced feature of the system. However, this provides a powerful extension mechanism to represent more advanced constraints such as spatial-temporal constraints.

For example, the [DASH](http://datashapes.org/constraints.html) [http://datashapes.org/constraints.html] namespace includes a collection of SHACL constraint components that extend the Core of SHACL with new constraint types including value types, string-based, property pairs, relationship constraint components.

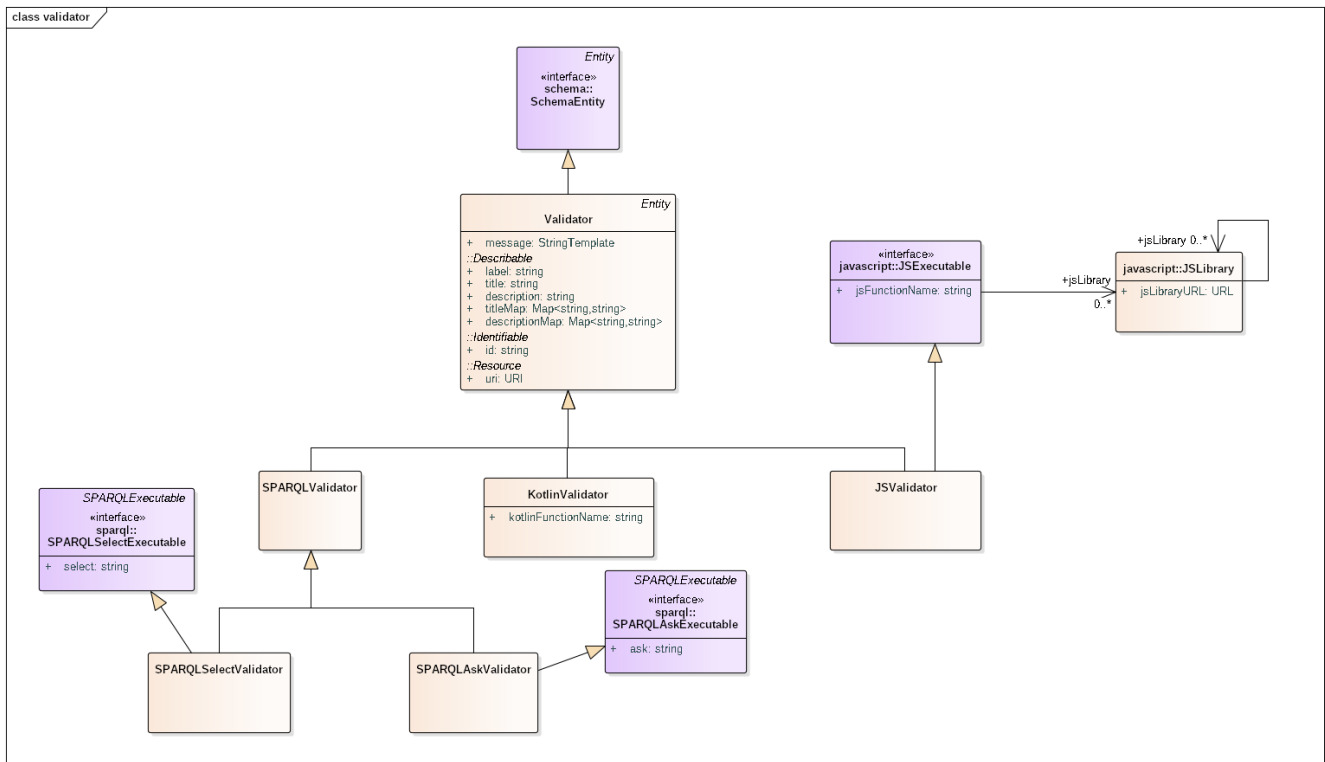


Figure 9. Validator Model

9.3.1. Built-in Constraint Components

The SHACL Core contains a list of built-in constraint components that are classified in Table 3. In the table, the parameter names are included because they are shorter than the component IRIs.

Table 4. Built-in Constraint Components

Operation	Parameters
Cardinality constraints	<i>sh:minCount</i> , <i>sh:maxCount</i>
Value types	<i>sh:class</i> , <i>sh:datatype</i> , <i>sh:nodeKind</i> , <i>sh:in</i> , <i>sh:hasValue</i>
Value range constraints	<i>sh:minInclusive</i> , <i>sh:maxInclusive</i> , <i>sh:minExclusive</i> , <i>sh:maxExclusive</i>
String based constraints	<i>sh:minLength</i> , <i>sh:maxLength</i> , <i>sh:length</i> , <i>sh:pattern</i>
Language based	<i>sh:uniqueLang</i> , <i>sh:languageIn</i>
Logical constraints	<i>sh:and</i> , <i>sh:or</i> , <i>sh:xone</i> , <i>sh:not</i>
Shape-based constraints	<i>sh:node</i> , <i>sh:property</i> <i>sh:qualifiedValueShape</i> , <i>sh:qualifiedValueShapesDisjoint</i> <i>sh:qualifiedMinCount</i> <i>sh:qualifiedMaxCount</i>
Closed shapes	<i>sh:closed</i> , <i>sh:ignoredProperties</i>
Property pair constraints	<i>sh:equals</i> , <i>sh:disjoint</i> <i>sh:lessThan</i> , <i>sh:lessThanOrEquals</i>
Non-validating constraints	<i>sh:name</i> , <i>sh:description</i> , <i>sh:order</i> , <i>sh:group</i>

The following ConstraintComponent is defined in SHACL as built-in.

```

sh:MinCountConstraintComponent
  a sh:ConstraintComponent ;
  rdfs:label "Min-count constraint component"@en ;
  rdfs:comment "A constraint component that can be used to restrict the minimum
number of value nodes."@en ;
  sh:parameter sh:MinCountConstraintComponent-minCount ;
  rdfs:isDefinedBy sh: .

sh:MinCountConstraintComponent-minCount
  a sh:Parameter ;
  sh:path sh:minCount ;
  sh:datatype xsd:integer ;
  sh:maxCount 1 ;
  rdfs:isDefinedBy sh: .

sh:minCount
  a rdf:Property ;
  rdfs:label "min count"@en ;
  rdfs:comment "Specifies the minimum number of values in the set of value
nodes."@en ;
  rdfs:range xsd:integer ;
  rdfs:isDefinedBy sh: .

```

9.3.2. User-defined Constraint Components

User-defined constraint components are defined by declaring a list of parameters and associating them with **validators**. Those validators are usually declared in SPARQL, although there is a WG note for allowing JavaScript-based validations.

SHACL-SPARQL provides a mechanism to declare reusable constraint components based on SPARQL query. Once defined, they can be used just like the other built-in SHACL Core components, without the need to write SPARQL. [Figure 10](#) shows an example of SHACL-SPARQL Constraint Component.

```

:FixedListConstraintComponent
a sh:ConstraintComponent ;
rdfs:label "Fixed list constraint component" ;
sh:parameter [
  sh:path :size ;
  sh:name "Size of list" ;
  sh:description "The size of the list" ;
] ;
sh:labelTemplate "Size of values: |"${size}"|" ;
sh:propertyValidator [
  a sh:SPARQLSelectValidator ;
  sh:message "${PATH} must have length {?size}, not {?count}" ;
  sh:prefixes [ sh:declare [
    sh:prefix "rdf" ;
    sh:namespace "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  ]
] ;
sh:select """
  SELECT $this ?value $count WHERE {
    $this $PATH ?value .
    { { SELECT $this ?value (COUNT(?member) AS ?count) $size WHERE {
      ?value rdf:rest*/rdf:first ?member
    } GROUP BY $this ?value $size
    }
    FILTER (!isBlank(?value) || ?count != $size)
  }
}"""
]

```

Figure 10. SPARQL Constraint Component

To create a new Constraint Component, a new vocabulary and vocabulary release needs to be created, so we can assign a unique namespace for the component.

The serialization of the user-defined constraints is be stored as Linked Data and indexed as JSON Objects with parameters and validator inline, so they can be quickly searched and retrieved, which improves performance of SHACL validators.

9.3.3. Rules

Business rules can be associated with shapes (that satisfy specific constraint conditions) to derive inferred RDF triples from existing asserted triples. SHACL rules build on SHACL to form a light-weight RDF vocabulary for the exchange of rules.

The UML Rule Model is shown in [Figure 11](#)

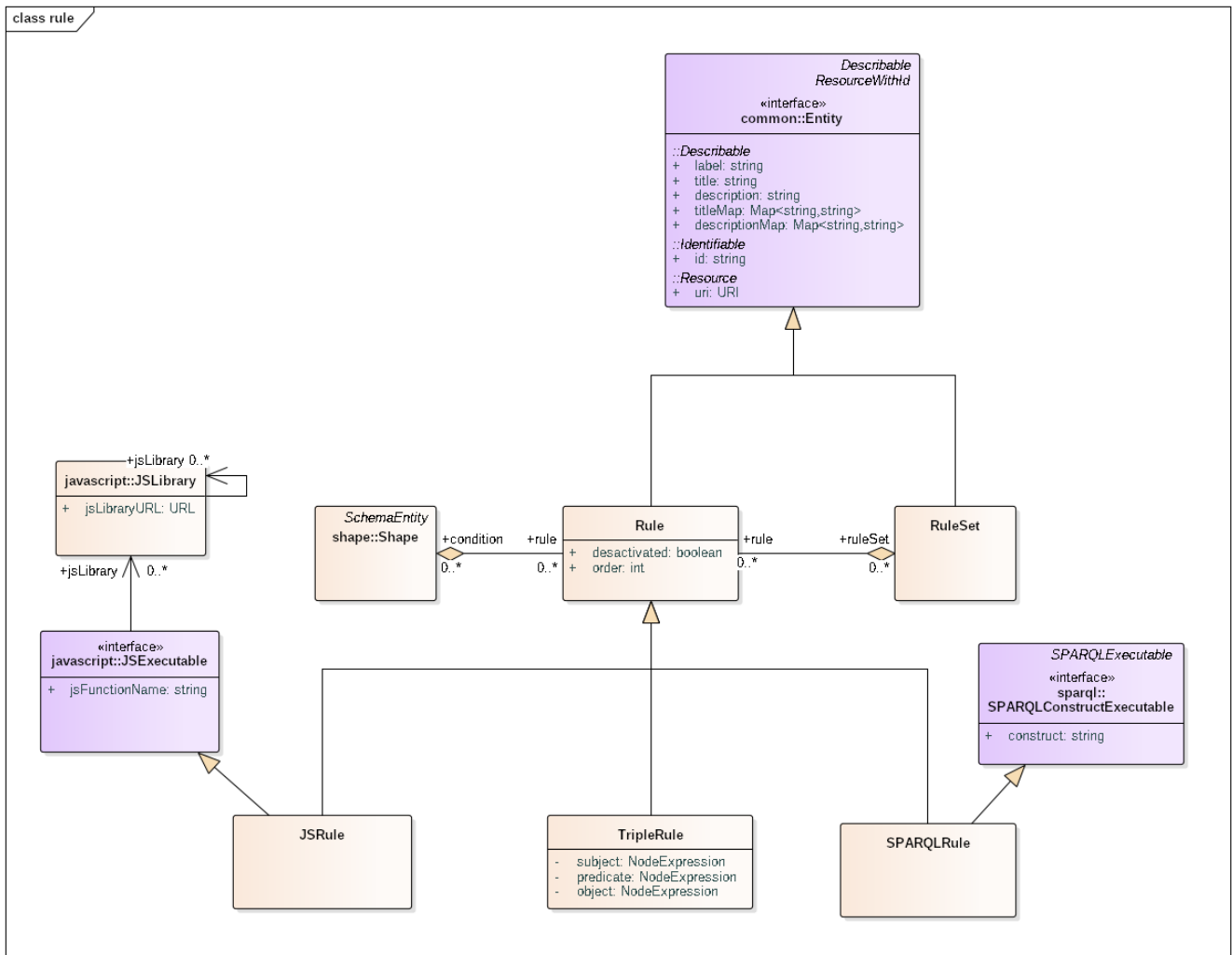


Figure 11. Rule Model

There are two main ways to describe rules. The first one is using the simple TripleRule class, which is an objectification of a statement. The second way is to use SPARQL construct, which is capable of creating multiple triples. A W3C note also proposed creating triples using a JavaScript Rule implemented by a JS function. Additional types could be added down the road using Typescript [32] or Kotlin [33].

The model also introduces the concept of RuleSet (which does not exist in SHACL). A RuleSet is an aggregation of Rules that capture all the business rules for a given domain. A RuleSet can also include reference to other rule sets. A RuleSet can be associated to an RDF dataset to add extra inferences on top of existing OWL reasoners. A RuleSet can also be used to implement the transformation from one model to another.

9.3.4. Functions

Functions play an important role in the constructions of rules, queries, targets and constraint components. The SPARQL query language provides a number of built-in functions that are defined by well-defined symbols (e.g. +, STRLEN, CONCAT..) but also uses IRI for user-defined functions (such as GeoSPARQL spatial functions).

SHACL functions declare operations that produce an RDF term based on zero or more parameters and a data graph. Each SHACL function has an IRI. The actual execution logic (or algorithm) of a SHACL function can be declared in a variety of execution languages. SHACL defines one specific

kind of SHACL functions, the SPARQL-based function. JavaScript-based Functions are defined in the separate SHACL-JS document. If it declares execution instructions for these platforms, the same function IRI can potentially be executed on a multitude of platforms.

SHACL functions can be called within FILTER or BIND clauses and similar features of SPARQL queries. SHACL functions can also be used declaratively in frameworks such as the SHACL node expressions which are used in SHACL rules. In those scenarios they may be used to perform data transformations such as string concatenation.

The UML Function Model is shown in [Figure 12](#)

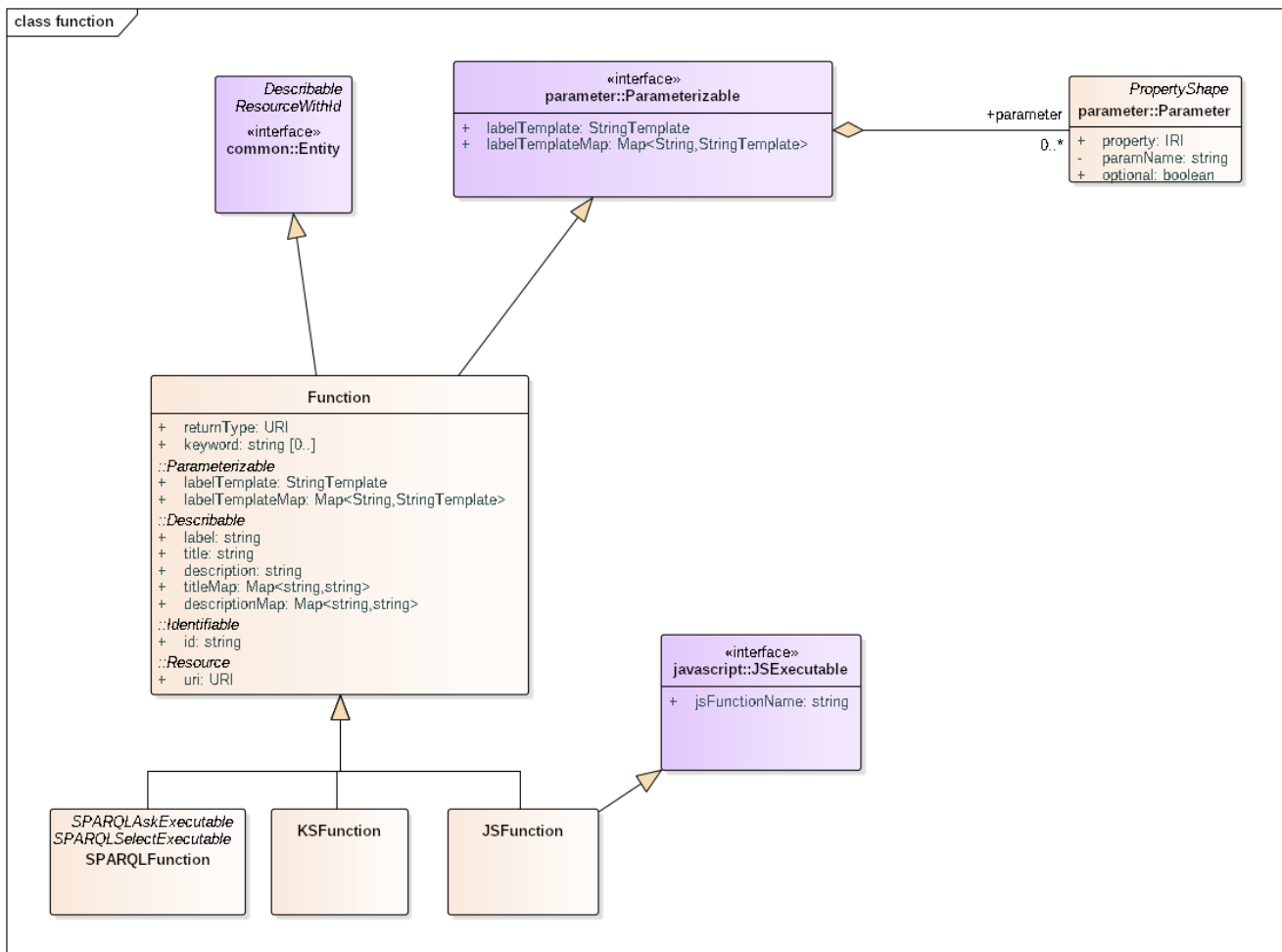


Figure 12. Function Model

Function Parameters

The parameters of a SHACL function are declared using the property **sh:parameter**. This corresponds closely to the parameter declarations of SPARQL-based constraint components, and the same syntax rules apply.

Parameters are ordered, corresponding to the notation of function calls in SPARQL such as `ex:exampleFunction(?param1, ?param2)`. The ordering of function parameters is determined as follows:

If any of the parameters have a value for **sh:order** then all of them are ordered in ascending order by the parameters' numeric values of `sh:order`, using 0 as default value if unspecified. If none of the parameters have a value for `sh:order` then all of them are ordered in ascending order of the local

names of their declared `sh:path` values. Each parameter may have its property `sh:optional` set to true to indicate that the parameter is not mandatory. If a function gets invoked without all its mandatory parameters then it returns no result node (an error in SPARQL, producing unbound in a BIND statement).

sh:returnType

A function may declare a single return type via `sh:returnType`.

A function has at most one value for `sh:returnType`. The values of `sh:returnType` are IRIs.

The return type may serve for documentation purposes only. However, in some execution languages such as JavaScript, the declared `sh:returnType` may inform a processor how to cast a native value into an RDF term.

SPARQL-based Functions

SHACL instances of `sh:SPARQLFunction` that are IRIs in a shapes graph are called SPARQL-based functions.

SPARQL-based functions have exactly one value for either `sh:ask` or `sh:select`. The values of these properties are strings that can be parsed into SPARQL queries of type ASK (for `sh:ask`) or SELECT (for `sh:select`) using the SHACL-SPARQL prefix declaration mechanism. SELECT queries return exactly one result variable and do not use the SELECT * syntax.

When the function is executed, the SPARQL processor pre-binds variables based on the provided arguments of the function call. In the SHACL functions example above, the value for the parameter declared as `ex:op1` is pre-bound to the SPARQL variable `$op1`, etc. For ASK queries, the function's return value is the result of the ASK query execution, i.e. true or false. For SELECT queries, the function's return value is the binding of the (single) result variable of the first solution in the result set. Since all other bindings will be ignored, such SELECT queries should only return at most one solution. If the result variable is unbound, then the function generates a SPARQL error.

9.3.5. Node Expressions

This section defines a feature called **node expressions**. Node expressions are declared as RDF nodes in a shapes graph and instruct a SHACL engine as to how to compute a set of nodes for a given focus node. Each node expression has one of the following types, each of which is defined together with its evaluation semantics in the following sub-sections.

Table 5. Node Expression Types

Node Expression Type	Syntax (Informative)	Summary
Focus Node Expression	<code>sh:this</code>	The set consisting of the current focus node.
Constant Term Expression	Any IRI or literal except <code>sh:this</code>	The set consisting of the given term.
Function Expression	Blank node with a list-valued triple	The results of evaluating a given SHACL Function.

Node Expression Type	Syntax (Informative)	Summary
Path Expression	Blank node with sh:path	The values of a given property path.
Filter Shape Expression	Blank node with sh:filterShape and sh:nodes	The sub-set of the input nodes that conform to a given shape.
Intersection Expression	Blank node with sh:intersection	The intersection of two or more input sets.
Union Expression	Blank node with sh:union	The union set of two or more input sets.

The basic idea of these expressions is that they can be used to derive a set of RDF nodes from a given focus node, such as the set of all values of a given property of the focus node. Some of these expressions can be nested: i.e, they use the output of another expression as their input, leading to evaluation chains and trees.

```
[ ex:concat (
  [ sh:path ex:firstName ]
  [ sh:path ex:lastName ]
)
] .
```

9.4. Application Profiles

A profile in the context of linked data is defined as "a named set of constraints on one or more identified base specifications, including the identification of any implementing subclasses of datatypes, semantic interpretations, vocabularies, options and parameters of those base specifications necessary to accomplish a particular function." Application profiles are included in this definition.

NOTE: ISO and OGC define an Application Profile as a set of one or more base standards and - where applicable - the identification of chosen clauses, classes, subsets, options and parameters of those base standards that are necessary for accomplishing a particular function [ISO 19101, ISO 19106].

A profile is based on an existing specification, like a standard, that can either be another profile or another specification. A profile can be expressed using a single specification, or there can be more than one component. A profile, in the context of linked data, may consist of:

- A vocabulary in the form of a schema (such as OWL or XML schema)
- A text file containing documentation for persons doing data creation
- Actionable validation code, such as a SHACL document

Profiles serve to enable different views of the same data, making it possible to define a specification of the data to fit specific user or application needs. The original vocabulary does not change, but the specific representation does. They can take a number of forms and can have a variety of relationships to existing vocabularies, standards, and other profiles. The applicability of the profile

can be tested for goodness of fit to assess the appropriateness for a certain application.

In the context of application profiles, a profile may consist of one or more (sub)sets of terms that gain information from one or more other vocabularies deemed appropriate for a specific application operating in a specific context.

9.4.1. Profiles Vocabulary

The [Profiles Vocabulary](https://www.w3.org/TR/dx-prof/) [https://www.w3.org/TR/dx-prof/] is an RDF vocabulary formalized by W3C to describe profiles of (one or more) standards for information resources. It describes the general pattern of narrowing the scope of a specification with additional, but consistent, constraints, and is particularly relevant to data exchange situations where conformance to such profiles is expected and carries additional context. The Profiles Vocabulary enables profile descriptions to specify the role of resources related to data exchange such as schemas, ontologies, rules about use of controlled vocabularies, validation tools, and guidelines. The ontology may however be used to describe the role of artefacts in any situation where constraints are made on the usage of more general specifications.

Chapter 10. Data Sources

To support this Testbed, NRCAN has provided a set of TTL files related to hydrological features that could be used as a seed for semantic enrichment from different data sources. We have investigated a number of data sources that could potentially be used for the semantic enrichment, however due to time constraints, the semantic enrichment was demonstrated only on two data sources: DBPedia and Wikidata. The reason for using these two sources was because they provided ready-to-use SPARQL endpoints that could be used by the integration framework without the need to perform a semantic mapping from data to Linked Data. This section also describes other potential data sources that could be used in the future for further enrichment.

10.1. NRCAN Datasets

The NRCAN datasets contains features classes shown in [Figure 13](#)

- ▼ ● owl:Thing
 - ▼ ● gw:GW_HydrogeoUnit
 - ▶ ● gw:GW_AquiferUnit
 - gw:GW_Well
 - hy:HY_Catchment
 - ▼ ● hy:HY_HydrometricFeature
 - hy:HY_HydrometricStation
 - hy:HY_Waterbody

Figure 13. NRCAN Ontology Classes

The NRCAN datasets contains features properties shown in [Figure 14](#)









-
- ▼  gw
 -  gw:gwAquiferSystem
 -  gw:gwAquiferSystemPart
 - ▼  hy
 -  hy:close-to
 -  hy:contains
 -  hy:downstreamOf
 -  hy:drains
 -  hy:drains-into
 -  hy:hydraulically-connected-to
 -  hy:inside
 -  hy:located-on
 -  hy:location-for
 -  hy:measuredBy
 -  hy:measures
 -  hy:near
 -  hy:overlaps
 -  hy:sameAs
 -  hy:upstreamOf

Figure 14. NRCan Ontology Properties

The following is a sample of original hydrologic features defined by NRCan.

```

@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix sesame: <http://www.openrdf.org/schema/sesame#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix fn: <http://www.w3.org/2005/xpath-functions#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix dct: <http://purl.org/dc/terms/>.
@prefix hy: <http://geosciences.ca/def/hydraulic#>.
@prefix dcat: <http://www.w3.org/ns/dcat#>.
@prefix ex: <http://gin.gw-info.net/example#>.
@prefix gw: <http://geosciences.ca/def/groundwater#>.

<https://geoconnex.ca/id/waterbody/9483b203be4111d892e2080020a0f4c9> a
hy:HY_WaterBody;
rdfs:label "Plan d'eau: Ruisseau a l'Ours"@fr, "Water body: Ruisseau a l'Ours"@en.

<https://geoconnex.ca/id/waterbody/1e041853be2d11d892e2080020a0f4c9> a
hy:HY_WaterBody;
rdfs:label "Plan d'eau: Riviere L'Acadie"@fr, "Water body: Riviere L'Acadie"@en.

<https://geoconnex.ca/id/waterbody/3effba23be2a11d892e2080020a0f4c9> a
hy:HY_WaterBody;
rdfs:label "Plan d'eau: Riviere des Hurons"@fr, "Water body: Riviere des Hurons"@en.

<https://geoconnex.ca/id/waterbody/60c56a06be4911d892e2080020a0f4c9> a
hy:HY_WaterBody;
rdfs:label "Plan d'eau: Riviere Richelieu"@fr, "Water body: Riviere Richelieu"@en.

<https://geoconnex.ca/id/catchment/020J> a hy:HY_Catchment;
rdfs:label "Bassin versant: Richelieu"@fr, "Watershed: Richelieu"@en.

<https://geoconnex.ca/id/catchment/020J*CD> a hy:HY_Catchment;
hy:contains <https://geoconnex.ca/id/featureCollection/wellsIn020J_CD>;
rdfs:label "Bassin versant: Ruisseau Landry - Riviere Richelieu"@fr, "Watershed:
Ruisseau Landry - Riviere Richelieu"@en.

```

To perform the correlation tasks, the labels were normalized by removing the typing prefix such as "Water body:" or "Plan d'eau:". By doing so the matching scoring was given better results against DBpedia and Wikidata. The following shows a sample of cleaned feature labels:

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix hy: <http://geosciences.ca/def/hydraulic#>.

<https://geoconnex.ca/id/waterbody/60c56a06be4911d892e2080020a0f4c9> a
hy:HY_WaterBody;
  rdfs:label "Riviere Richelieu"@fr,
            "Richelieu River"@en.

<https://geoconnex.ca/id/catchment/020J*CF> a hy:HY_Catchment;
  rdfs:label "Riviere des Iroquois "@fr,
            "Iroquois River"@en.

<https://geoconnex.ca/id/waterbody/1e041853be2d11d892e2080020a0f4c9> a
hy:HY_WaterBody;
  rdfs:label "Riviere L'Acadie"@fr,
            "Acadia River"@en.

<https://geoconnex.ca/id/waterbody/3effba23be2a11d892e2080020a0f4c9> a
hy:HY_WaterBody;
  rdfs:label "Riviere des Hurons"@fr,
            "Huron River"@en.
```

As the above data sources do not provide any specialized information on wells and stations, there were few useful properties beyond the label and the type of the feature that could be leveraged for performing the fusion against the open source data. Only water bodies and catchments provided some results.

10.2. DBPedia

The most successful and popular wiki by far is probably Wikipedia, the largest online encyclopedia created and maintained by a global distributed author community. Wikipedia is translated into 306 languages and the English version contains more than 5.9 million articles. However, Wikipedia is intended for human reading and cannot be processed by machines. To address this challenge, semantic web technology was used to make better use of the knowledge embedded in Wikipedia pages. [DBPedia](https://wiki.dbpedia.org/) [https://wiki.dbpedia.org/] is a community effort to extract structured information from Wikipedia and to make this information available on the web. DBPedia allows the user to ask sophisticated queries against Wikipedia (using SPARQL), and to link other data sets on the web to Wikipedia data.

This section discusses the different data sources that are used as data input in the implementation, and the different challenges that they present.

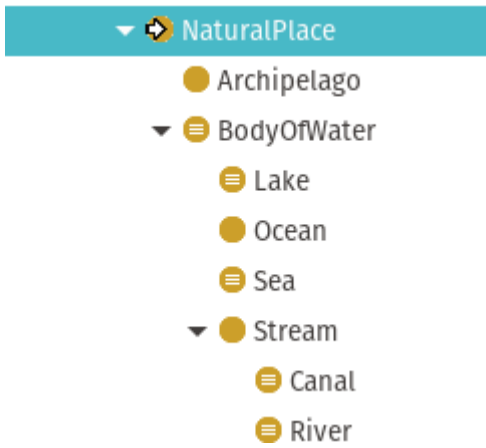


Figure 15. DBpedia Hydrologic Classes

We also identified the following relevant properties from DBpedia:

- outflow
- mouth place of
- mouth mountain of
- geometry
- source1location
- westOf
- northWestOf
- southWestOf
- eastOf
- northOf
- southOf

10.3. Wikidata

Wikidata is a project of Wikimedia Deutschland which started on October 30, 2012. The aim of the project is to provide data which can be used by any Wikipedia project, including Wikipedia. Wikidata does not only store facts, but also the corresponding sources, so that the validity of facts can be checked. Labels, aliases, and descriptions of entities in Wikidata are provided in more than 350 languages. Wikidata is a community effort, i.e., users collaboratively add and edit information. Also, the schema is maintained and extended based on community agreements.

The relevant properties from Wikidata for this project are shown in [Figure 16](#):

Title	ID	Data type	Description	Examples	Inverse
mouth of the watercourse	P403	Item	river mouth: the body of water to which the watercourse drains	Vltava <mouth of the watercourse> Elbe	-
origin of the watercourse	P885	Item	river source: main source of a river or stream	Inn <origin of the watercourse> Lughinsee	-
watershed area	P2053	Quantity	drainage basin: size of a stream's watershed (drainage basin)	Po <watershed area> 71,000 square kilometre	-
basin country	P205	Item	country that have drainage to/from or border the body of water	Vltava <basin country> Czech Republic	-
tributary	P974	Item	tributary: stream or river that flows into this main stem (or parent) river	Mississippi River <tributary> Missouri River	distributary
inflows	P200	Item	inflow: major inflow sources — rivers, aquifers, glacial runoff, etc. Some terms may not be place names, e.g. none	Lake Baikal <inflows> Upper Angara River	-
lake outflow	P201	Item	discharge: rivers and other outflows waterway names. If evaporation or seepage are notable outflows, they may be included. Some terms may not be place names, e.g. evaporation	Lake Ontario <lake outflow> Saint Lawrence River	origin of the watercourse
lakes on river	P469	Item	lake: lakes, reservoirs, waterfalls the river flows through	Missouri River <lakes on river> Lake Sharpe	-
bodies of water basin category	P1200	Item	the Wikimedia category associated with the basin of a bodies of water	Danube <bodies of water basin category> Category:Danube basin	-
instance of	P31	Item	instance of: that class of which this subject is a particular example and member (subject typically an individual member with a proper name label); different from P279; using this property as a qualifier is deprecated—use P2868 or P3831 instead	Volga <instance of> river	-
instance of	P31	Item	instance of: that class of which this subject is a particular example and member (subject typically an individual member with a proper name label); different from P279; using this property as a qualifier is deprecated—use P2868 or P3831 instead	Panama Canal <instance of> canal	-
instance of	P31	Item	instance of: that class of which this subject is a particular example and member (subject typically an individual member with a proper name label); different from P279; using this property as a qualifier is deprecated—use P2868 or P3831 instead	Molenbeek <instance of> stream	-
instance of	P31	Item	instance of: that class of which this subject is a particular example and member (subject typically an individual member with a proper name label); different from P279; using this property as a qualifier is deprecated—use P2868 or P3831 instead	Rhine <instance of> main stream	-
distance from river mouth	P2148	Quantity	river mile and river mouth: qualifier for P974 statements on streams	-- Property talk:P2148	-
discharge	P2225	Quantity	discharge: volume rate of water flow which is transported through a given cross-sectional area	Ab-donbal qanat <discharge> 0.16 cubic metre per second	-

Figure 16. Wikidata relevant Hydrofeature Classes

10.4. Geonames

The [GeoNames](https://www.geonames.org/) [https://www.geonames.org/] geographical database is available for download free of charge under a creative commons attribution license. The database contains over 25 million geographical names and consists of over 11 million unique features including 4.8 million populated places and 13 million alternate names. All features are categorized into one of nine feature classes and further subcategorized into one of [645 feature codes](http://forum.geonames.org/gforum/posts/list/130.page) [http://forum.geonames.org/gforum/posts/list/130.page]. There are a number of features code related to Hydrologic features codes:

The feature code H represents hydrofeatures such as stream, lake, bay, channel, pond, etc. [Table 5](#) has some relevant feature types that are relevant for this testbed:

Table 6. Geonames Hydrofeature relevant codes

Code	Description
CHN	channel the deepest part of a stream, bay, lagoon, or strait, through which the main current flows
CNL	canal an artificial watercourse
PND	pond a small standing waterbody
RSV	reservoir(s) an artificial pond or lake
SEA	sea a large body of salt water more or less confined by continuous land or chains of islands forming a subdivision of an ocean
STM	stream a body of running water moving to a lower level in a channel on
STMM	stream mouth(s) a place where a stream discharges into a lagoon, lake, or the sea
STMX	section of stream

STRT	strait a relatively narrow waterway, usually narrower and less extensive than a sound, connecting two larger bodies of water
WLL	well a cylindrical hole, pit, or tunnel drilled or dug down to a depth from which water, oil, or gas can be pumped or brought to the surface
WLLQ	abandoned well
WLLS	wells cylindrical holes, pits, or tunnels drilled or dug down to a depth from which water, oil, or gas can be pumped or brought to the surface
WTLD	wetland an area subject to inundation, usually characterized by bog, marsh
WTRC	watercourse a natural, well-defined channel produced by flowing water, or an artificial channel designed to carry flowing water

10.5. OpenStreetMap

OpenStreetMap is built by a community of mappers who contribute and maintain data about roads, facilities, waterways, trails, cafés, railway stations, and much more, all over the world.

[Figure 17](#) shows relevant features for the NRCan datasets.

Natural watercourses

Key	Value	Element	Description	Map rendering	Image	Count
waterway	river		The linear flow of a river, in flow direction.			2 116 1 219 237 17 058
waterway	riverbank		The water-covered area of a river			44 251 601 49 015
waterway	stream		A naturally-forming waterway that is too narrow to be classed as a river.			6 176 11 921 024 20 404
waterway	tidal_channel		A natural intertidal waterway in mangroves, salt marshes and tidal flats with water flow in the direction of the tide			0 2 145 3

This table is auto-generated. See [Taginfo/Taglists](#) for a documentation on it.

Man-made waterways

Key	Value	Element	Description	Map rendering	Image	Count
waterway	canal		An artificial "open flow" waterway used to carry useful water for transportation, waterpower, or irrigation.			2 158 370 805 1 541
waterway	drain		An artificial free flow waterway used for carrying superfluous water, usually lined with concrete.			504 874 498 579
waterway	ditch		A small man-made drainage waterway, usually unlined.			79 2 405 195 918
waterway	pressurised		An artificial tunnel or pipeline where water flows in a closed space without air			0 2 383 0
waterway	fairway		A navigable route in a lake or sea, marked by buoys or beacons.			1 1 107 18

Figure 17. Relevant Hydrofeature Classes from OpenStreetMap

10.6. Freebase

https://web.archive.org/web/20120516075431/http://blog.freebase.com/2008/10/30/introducing_the_rdf_service/] is a Knowledge Graph (KG) announced by Metaweb Technologies, Inc. in 2007 and was acquired by Google Inc. on July 16, 2010. In contrast to DBpedia, Freebase had provided an interface that allowed end-users to contribute to the KG by editing structured data. Besides user-contributed data, Freebase integrated data from Wikipedia, Notable Names Database (NNDB), Fashion Model Directory (FMD), and MusicBrainz. Freebase uses a proprietary graph model for storing also complex statements. On December 16, 2014, the Freebase team announced that Freebase would shutdown its services on June 30, 2015. Wikimedia Deutschland and Google plan to integrate Freebase data into Wikidata in the near future – a tool for that will be developed prior to August 2015 – and to then close the Freebase website at the earliest three months later.

Chapter 11. Implementation

11.1. Architecture

The integration pipeline system needs to address a number of challenges to get a unified integrated view of the information from different data sources [34]:

- Syntactic and semantic heterogeneity of data.
- Schema, identity, and data conflicts.
- Incorrect or otherwise flawed data.
- Identification of a target schema and schema translation.
- Presentation of results.

11.1.1. Semantic Integration Pipeline

One of the main goals of this Testbed was to establish a Linked Data integration framework capable to integrate various data sources expressed as Linked data and fuse the information into a unified, coherent and complete view. While the time frame to accomplish the implementation of the whole framework was too short for this Testbed, participants attempted to identify and define the main components of the integration framework pipeline. More work will be needed in future Testbeds to complete the whole pipeline.

Data integration systems use various combinations of steps to cope with the data integration challenges enumerated above. The approach used is illustrated in Figure 18 and consists of the following steps:

1. **Ingestion:** data are mapped to data schema
2. **Semantic Integration:** data are mapped against a semantic model (ontology)
3. **Mediation:** Semantic Model are mediated to a common model
4. **Deduplication:** Entities that are identical semantically are correlated using similarity metrics
5. **Conflict Resolution:** Correlated entities are fused together following some conflict resolution policies.

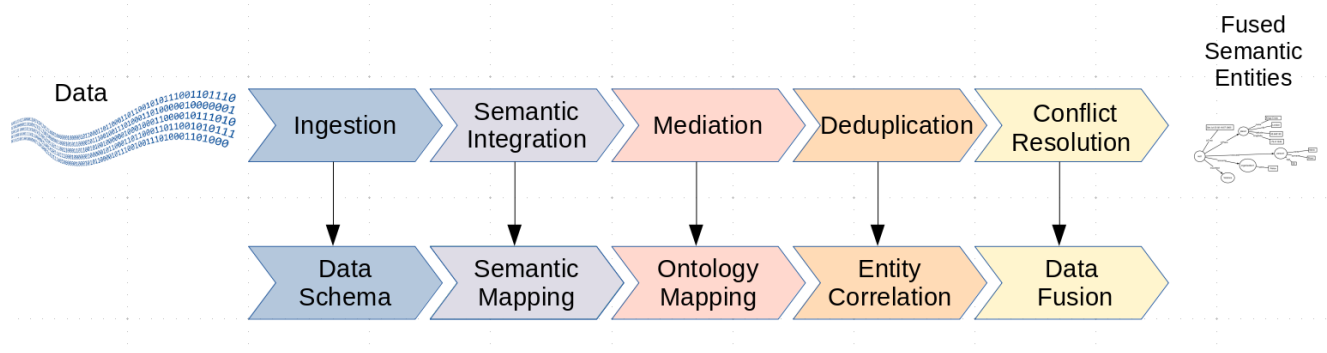


Figure 18. Proposed Integration Pipeline

Note that Step 3 and Step 4 can be switched in order.

Ingestion

The integration workflow starts with the ingestion phase which collects new sources of graph data, and ends with these contents being properly incorporated into the integrated, unified, complete, coherent model. Source Data can be received from a variety of sources, in various states of maturity and exploitation readiness. The source content models and formats vary widely, making it difficult to reduce all source graphs to an acceptable harmonized state for ready ingestion into the integration pipeline. We can distinguish them into the following categories of data sources:

- **RDF repositories:** Makes information directly available as Linked Data accessible through SPARQL endpoints: information can remain on the remote SPARQL repositories and be queryable using SPARQL repositories
- **Linked Data dumps:** These data can be easily ingested into local RDF repositories and be queryable using SPARQL.
- **Data sources that have APIs mappable to SPARQL queries:** Examples of such data sources are SQL datastores, NoSQL stores providing API such as ElasticSearch, MongoDB, Neo4J, XML stores. These types of stores can use a virtual knowledge layer that perform SPARQL query rewriting to native query (SQL, Gremlin, Elasticsearch API, etc). The advantage of this approach is that the data remains in their native store and leverage the existing lifecycle and optimization existing in these stores (Spatial indexing for example). Multiple virtual mappings can be defined for the same store using different ontologies.
- **Structured Data formats that is not accessible through APIs or with limited API expressiveness (not mappable to SPARQL):** Examples of such sources are CSV, JSON, XML files, Twitter feed, RSS feeds, etc. The typical approach to ingest these types is by writing RDF scrapping code that converts the format to RDF and stores the results in an RDF repository.
- **Unstructured Data:** This requires a more advanced processing pipeline to make the data readily available for integration with other sources of information. An example of a processing pipeline is to process text information by using Natural Language Processing (NLP) to extract annotations such as lexical entities, relationships and semantic grounding of the entities and relations. The output of the pipeline would be a Linked Data graph capturing the tacit knowledge conveyed by the text document. This output requires to be stored in an RDF repository to be further integrated with other data sources.

11.2. SHACL Engine

During this Testbed, the participants evaluated the use of an open source SHACL engine. The reference implementation of SHACL implemented by TopQuadrant was investigated. While the engine provides a complete implementation of SHACL-Core and SHACL Advanced Feature, it does not provide fine grained control of the different SHACL artefacts used by the different ontologies used for this Testbed. The integration with external resources such as SPARQL endpoint was not easily implementable, as the reference implementation requires at this point using graphs stored in memory. For this reason, Image Matters implemented its own SHACL engine that provides fine grained control of the SHACL elements such as functions, rules, property shapes, constraint components. The SHACL API was integrated with the correlation engine and is planned to be used for the mediation and fusion engine in the future. The performance of the new engine SHACL was about 10 times faster than the reference implementation for the test cases defined by W3C.

correlation is always associated with a similarity method (**sim:SimilarityMethod**) which is used to calculate the similarity score between two entities. The correlation rule defines the minimum threshold (**corr:threshold**) to use to generate a correlation link between two entities and an optional parameter to indicate the number of links (**corr:numLinks**) to generate.

11.4. Similarity Ontology

The Similarity Ontology defines concepts and properties for describing similarity methods to correlate entities described as Linked Data. The ontology uses the standard W3C SHACL **Property Shape** and **SHACL Path** model to describe the path and constraints on source and target node that needs to be correlated. The ontology leverages the Metrics Ontology (described below) to refer to metrics used for comparison. This ontology addresses some of the shortcomings of previous linked discovery framework such as SILK and LINES by providing a model that is based on open standards (RDF, OWL and SHACL) favoring its reusability using standard Linked Data mechanism. This model can be extended by adding new functions (defined in SHACL) and constraint conditions that need to be satisfied on the node values that needs to be compared. The ontology is also modularized in such a way that it can be reused in different domains and applications.

The ontology is defined by the namespace: <http://purl.org/ontology/fusion/similarity#> and uses the preferred prefix **sim**. The UML model is shown in Figure 20

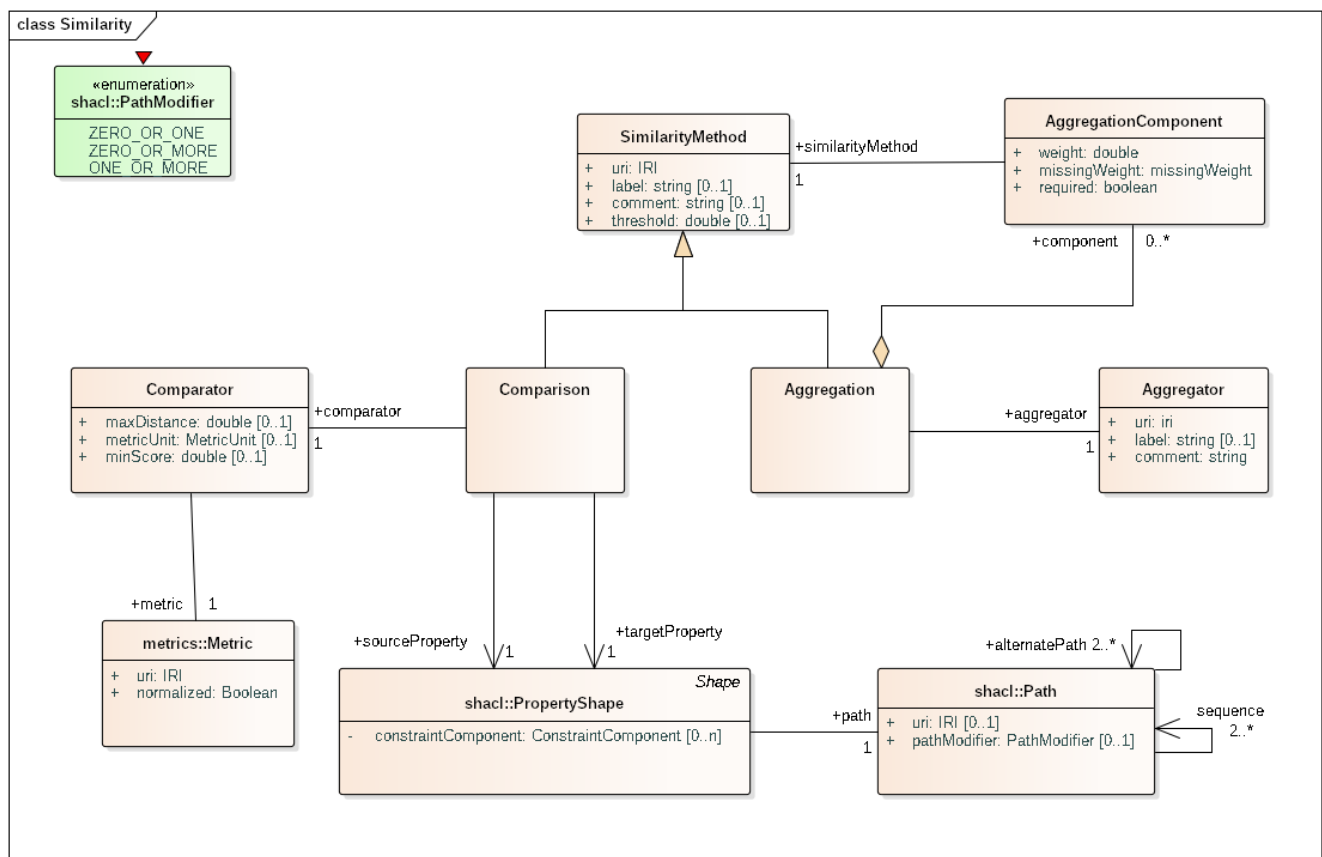


Figure 20. Similarity UML Model

11.4.1. Similarity Method

Two nodes of an RDF graph (literal or resource) have a similarity measure determined by a similarity method using a specific metric. **SimilarityMethod** is the top abstract class of the

ontology. It has two subclasses that can be instantiated **Comparison** and **Aggregation**.

11.4.2. Comparison

A **sim:Comparison** is defined as a subclass of **sim:SimilarityMethod**. It refers to a source property value and target property value specified with a **shacl:PropertyShape**. A **shacl:PropertyShape** must have a path from the focus node to a value node using the **shacl:Path** model. The nodes referred by the path can have additional constraints defined by **shacl:ConstraintComponent** as specified in the SHACL model. An example of constraints could be a node value that is defined a given language (English or French for example) using the constraint component **shacl:languageIn**. The values referred by the **sim:sourceProperty** and **sim:targetProperty** are compared using a **sim:Comparator**.

WARNING

The UML shows a generic property constraintComponent on shacl:PropertyShape as a placeholder for the numerous built-in constraint component properties (such as **shacl:minCount**, **shacl:languageIn**, **sh:in**, etc.)

11.4.3. Comparator

A comparator refers to a **metric:Metric** that is relevant for type of data that needs to be compared (spatial, temporal, lexical, semantic). The comparator has a property **sim:maxDistance** with optionally an associated **metric:metricUnit**. This is used to set the value of the comparator to 0 if the value of the comparison exceeds this distance. This property can be used with a Euclidian distance metric that is set to 0 if the calculated distance exceeds, for example, 10 km. Finally the comparator can optionally take a **sim:minScore** to keep only comparison that are greater than the minimum score value.

11.4.4. Aggregation

A **sim:Aggregation** is defined as a subclass of **sim:SimilarityMethod** which is composed of one or more **sim:AggregationComponent** elements that refer to a **sim:SimilarityMethod** instance (**Comparator** or nested **Aggregation**). Each component has a weight and/or **sim:missingWeight** that is used by an **sim:Aggregator** to calculate the final similarity score. The missing weight is used when there a component of the aggregation does not exist. The missing weight is used as a default value for component.

11.4.5. Aggregator

sim:Aggregator defines the way the weighted aggregation components are aggregated together to get a final score. There are a number of **sim:Aggregator** instances predefined in the ontology:

sim:Average	Calculate the weight average of the aggregation components
sim:GeometricMean	Calculate the geometric mean of the aggregation components
sim:Maximum	Returns the maximum value of the aggregate components

sim:Minimum	Returns the minimum value of the aggregate components
sim:QuadraticMean	Calculate the quadratic mean of the aggregation components

Other types of aggregator can be added such as geometric aggregator.

11.5. Metrics Ontology

There are a large number of metrics (whether distance or similarity metrics) that can be used to compare entity attributes such as lexical, spatial, temporal, semantic, numeric metrics. However, the participants have not identified any existing ontology that formalize these metrics. Therefore, a new ontology module to represent metrics is introduced in this Testbed. The metrics ontology module was defined separately in order to favor reusability for other applications. The ontology provides the core concepts to define metrics and their types that can be classified using different classification scheme. The aim of the ontology is to provide an extensible framework to introduce new metrics when necessary.

The ontology is defined by the namespace: <http://purl.org/ontology/metrics#> and uses the preferred prefix **metric**. The UML model of the ontology is shown in Figure 21.

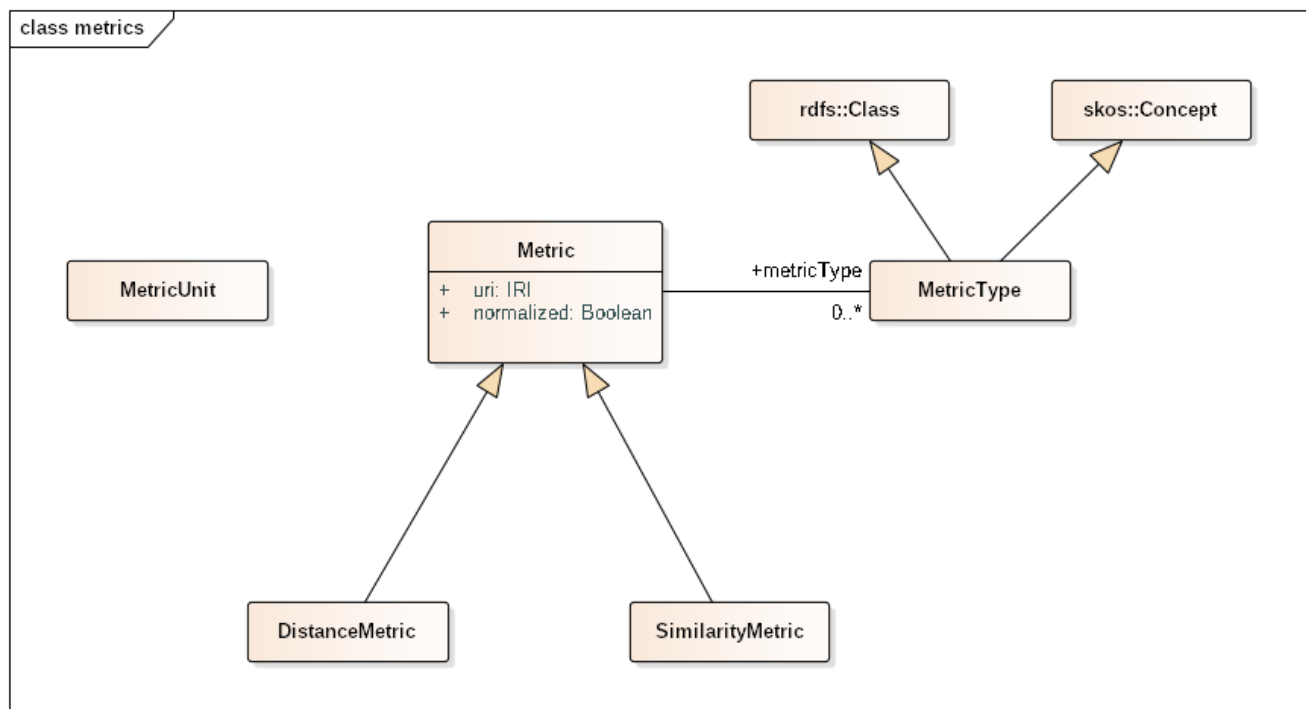


Figure 21. Metrics UML Model

11.5.1. Metric

metric:Metric is a top class and is always defined by a URI. The property **metric:isNormalized** is a property that indicates whether a metric is normalized (between 0 and 1) or not.

metric:Metric has two sub-classes:

- **metric:DistanceMetric**: Measures a distance, meaning the larger the measured value, the less

similar are the values that are compared.

- **metric:SimilarityMetric**: Measure a similarity distance. Typically, the value are normalized between 0 and 1, which 0 indicating that the two values are dissimilar, and 1 indicating that there are equals.

Metrics that have been implemented and are available are described in table [Table 7](#).

Table 7. Metric Categories

Metric	Category
Equality	Discrete
Numeric Distance	Numeric
Euclidian distance	Spatial
Haversine distance	Spatial
Hausdorff similarity	Spatial
Frechet distance	Spatial
Temporal interval similarity	Temporal
Date Time distance	Temporal
Duration Distance	Temporal
Levenshtein	String
Tokenized Levenshtein	String
Jaro distance	String
Jaro Winkler	String
Needleman Wunsch	String
Dice Coefficient	String
Level 2 Jaro	String
Level 2 Levenshtein	String
Jaccard distance	Semantic

11.5.2. Metric Type

The **metric:MetricType** class is modelled as a subclass of **skos:Concept**, so they can be modelled flexibly in a classification scheme (modeled as **skos:ConceptScheme**) (taxonomy). The Metric is associated with zero or more **metric:MetricType** instances using the property **metric:metricType** following a soft typing approach (instead of strong typing using **rdf:type** property). This allows more flexibility to classify Metric instance using different classification approaches such as spatial, temporal, numeric, discrete, semantic, string.

11.5.3. Metric Unit

metric:MetricUnit is used to define a unit of measure. There are a number of ontologies for units of measurement that are available. QUDT [36] is one the most popular. We decided not to commit to any of the ontologies available, so we simply introduced the class **metric:MetricUnit** that can be

used to "tag" any unit of measure instance defined in other ontologies. The ontology introduces two properties that refer to a MetricUnit: **metric:metricUnit** and its sub-property **metric:preferredMetricUnit**.

11.5.4. Example

The following is an example of LinkSetSpecification between NRCAN hydrology model and DBpedia accessible through SPARQL Endpoints. The LinkSetSpecification specification correlate NRCAN **HY_WaterBody** and **HY_Catchment** with DBpedia **BodyOfWater** using a Levenshtein string similarity metric [37] on the **rdfs:label**. Any correlations with similarity score above 0.91 are returned.

```
@prefix :      <http://www.imagemattersllc.com/testbed15/hydro/correlationModels#> .
@prefix corr:  <http://purl.org/ontology/correlation#> .
@prefix void:  <http://rdfs.org/ns/void#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix ks:    <http://www.usersmarts.com/ont/2005/06/ks#> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix prm:   <http://www.smartrealm.com/ont/ks/param#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix kspf:  <http://www.knowledgesmarts.com/ontologies/pf#> .
@prefix ksfun: <http://www.usersmarts.com/ont/2005/06/ks/functor#> .
@prefix shacl: <http://www.w3.org/ns/shacl#> .
@prefix metric: <http://purl.org/ontology/metrics#> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix sim:   <http://purl.org/ontology/fusion/similarity#> .
@prefix dash:  <http://datashapes.org/dash#> .
@prefix ksd:   <http://www.usersmarts.com/ont/2007/02/ks/descriptor#> .
@prefix dc:    <http://purl.org/dc/elements/1.1/> .

:      a          owl:Ontology ;
      shacl:declare [ shacl:namespace "http://dbpedia.org/ontology/" ;
                      shacl:prefix   "dbo"
                      ] ;
      shacl:declare [ shacl:namespace "http://dbpedia.org/resource/" ;
                      shacl:prefix   "dbr"
                      ] ;

      shacl:declare [ shacl:namespace "http://www.w3.org/2000/01/rdf-schema#" ;
                      shacl:prefix   "rdfs"
                      ] .

:WaterBodyShape a      shacl:NodeShape ;
                 shacl:targetClass <http://geosciences.ca/def/hydraulic#HY_WaterBody>,
                 <http://geosciences.ca/def/hydraulic#HY_Catchment>.

:RiverInCanadaShape a shacl:NodeShape ;
                    shacl:target :RiverInCanadaSPARQLTarget .

:RiverInCanadaSPARQLTarget
```

```

    a          shacl:SPARQLTarget ;
    shacl:prefixes : ;
    shacl:select "SELECT ?this \n WHERE { ?this a  dbo:BodyOfWater;\n
dbo:sourceCountry dbr:Canada . }" .

:nrcan-dbpedia-correlation-set
    a          corr:CorrelationRuleSet ;
    rdfs:comment "Rule Set to map NRCAN Hydro ontology to DBPedia" ;
    rdfs:label  "CorrelationRuleSet NRCAN Hydro-Dbpedia" ;
    corr:correlationRule :RiverCorrelationRule ;
    corr:sourceSchema <http://geosciences.ca/def/hydraulic#> ;
    corr:targetSchema <http://dbpedia.org/ontology/> .

:RiverCorrelationRule
    a          corr:CorrelationRule ;
    corr:sourceShape :WaterBodyShape ;
    corr:targetShape :RiverInCanadaShape ;
    sim:similarityMethod :NameComparison .

:NameComparison a      sim:Comparison ;
    sim:comparator [ a          sim:Comparator ;
                    metric:metric metric:Levenshtein
                    ] ;
    sim:sourceProperty [
        shacl:path rdfs:label
    ];
    sim:targetProperty [
        shacl:path rdfs:label
    ];
    sim:threshold "0.91"^^xsd:double .

metric:Levenhstein a metric:Metric .

:NRCAN-Wikidata-LinkSetSpec
    a          corr:LinkSetSpecification ;
    corr:correlationRuleSet :nrcan-dbpedia-correlation-set ;
    corr:sourceDataset <http://nrcan.org> ;
    corr:targetDataset <http://dbpedia.org> ;
    corr:linkType owl:sameAs.

<http://nrcan.org> a      void:Dataset ;
    void:sparqlEndpoint <http://localhost:33335/contexts/nrcan/sparql> .

<http://dbpedia.org>
    a          void:Dataset ;
    void:sparqlEndpoint <http://localhost:33335/contexts/dbpedia/sparql> .

```

The following example shows a LinkSetSpecification correlating NRCAN hydrology model and Wikidata accessible through SPARQL endpoints. The LinkSetSpecification specification correlates NRCAN **HY_WaterBody** with Wikidata **wd:Q4022** class ([River](https://www.wikidata.org/wiki/Q4022) [https://www.wikidata.org/wiki/Q4022])

using a Levenshtein string similarity metric [37] on the **rdfs:label**. Any correlations with similarity score above 0.9 are returned.

```
@prefix :      <http://www.imagemattersllc.com/testbed15/hydro/correlationModels#> .
@prefix corr:  <http://purl.org/ontology/correlation#> .
@prefix void:  <http://rdfs.org/ns/void#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix ks:    <http://www.usersmarts.com/ont/2005/06/ks#> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix prm:   <http://www.smartrealm.com/ont/ks/param#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix kspf:  <http://www.knowledgesmarts.com/ontologies/pf#> .
@prefix ksfun: <http://www.usersmarts.com/ont/2005/06/ks/functor#> .
@prefix shacl: <http://www.w3.org/ns/shacl#> .
@prefix metric: <http://purl.org/ontology/metrics#> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix sim:   <http://purl.org/ontology/fusion/similarity#> .
@prefix dash:  <http://datashapes.org/dash#> .
@prefix ksd:   <http://www.usersmarts.com/ont/2007/02/ks/descriptor#> .
@prefix dc:    <http://purl.org/dc/elements/1.1/> .

:      a          owl:Ontology ;
      shacl:declare [ shacl:namespace "http://www.wikidata.org/prop/" ;
                      shacl:prefix   "wdp"
                      ] ;
      shacl:declare [ shacl:namespace "http://www.wikidata.org/prop/direct/" ;
                      shacl:prefix   "wdt"
                      ] ;
      shacl:declare [ shacl:namespace "http://www.wikidata.org/entity/" ;
                      shacl:prefix   "wd"
                      ] ;
      shacl:declare [ shacl:namespace "http://www.w3.org/2000/01/rdf-schema#" ;
                      shacl:prefix   "rdfs"
                      ] ;
      shacl:declare [ shacl:namespace "http://www.wikidata.org/entity/statement/"
                      shacl:prefix   "wds"
                      ] ;
      shacl:declare [ shacl:namespace "http://www.wikidata.org/prop/" ;
                      shacl:prefix   "p"
                      ] ;
      shacl:declare [ shacl:namespace "http://schema.org/" ;
                      shacl:prefix   "schema"
                      ] ;
      shacl:declare [ shacl:namespace "http://www.wikidata.org/value/" ;
                      shacl:prefix   "wdv"
                      ] .

:WaterBodyShape a      shacl:NodeShape ;
                 shacl:targetClass <http://geosciences.ca/def/hydraulic#HY_WaterBody> ,
```

```
<http://geosciences.ca/def/hydraulic#HY_Catchment>.
```

```
:RiverInCanadaShape a shacl:NodeShape ;  
  shacl:target :RiverInCanadaSPARQLTarget .
```

```
:RiverInCanadaSPARQLTarget  
  a shacl:SPARQLTarget ;  
  shacl:prefixes : ;  
  shacl:select "SELECT ?this \n WHERE { ?this wdt:P31 wd:Q4022;\n wdt:P17  
wd:Q16 . }" .
```

```
:nrcan-wikidata-correlation-set  
  a corr:CorrelationRuleSet ;  
  rdfs:comment "Rule Set to map NRCAN Hydro ontology to Wikidata" ;  
  rdfs:label "CorrelationRuleSet NRCAN Hydro-Wikidata" ;  
  corr:correlationRule :RiverCorrelationRule ;  
  corr:sourceSchema <http://geosciences.ca/def/hydraulic#> ;  
  corr:targetSchema <http://www.wikidata.org/ontology#> .
```

```
:RiverCorrelationRule  
  a corr:CorrelationRule ;  
  corr:sourceShape :WaterBodyShape ;  
  corr:targetShape :RiverInCanadaShape ;  
  sim:similarityMethod :NameComparison .
```

```
:NameComparison a sim:Comparison ;  
  sim:comparator [ a sim:Comparator ;  
  metric:metric metric:Levenshtein;  
  ] ;  
  sim:sourceProperty [  
  shacl:path rdfs:label  
  ] ;  
  sim:targetProperty [  
  shacl:path rdfs:label  
  ] ;  
  sim:threshold "0.9"^^xsd:double .
```

```
metric:Levenshtein a metric:Metric .
```

```
:NRCAN-Wikidata-LinkSetSpec  
  a corr:LinkSetSpecification ;  
  corr:correlationRuleSet :nrcan-wikidata-correlation-set ;  
  corr:sourceDataset <http://nrcan.org> ;  
  corr:targetDataset <http://wikidata.org>;  
  corr:linkType owl:sameAs .
```

```
<http://nrcan.org> a void:Dataset ;  
  void:sparqlEndpoint <http://localhost:33335/contexts/nrcan/sparql> .
```

```
<http://wikidata.org>
```

```
  a                void:Dataset ;  
  void:sparqlEndpoint <https://query.wikidata.org/sparql> .
```

11.6. Correlation Engine

A correlation engine built on top of the SHACL engine was implemented to handle the Correlation LinkSetSpecification. The engine accepts a LinkedSetSpecification, a source and target model, which could be served through a SPARQL endpoint or loaded in memory and parameters to include generated statements and closure of target nodes. The engine was integrated with a RESTful API, so it could be made accessible by web client.

11.7. Semantic Mediation Ontology

The semantic mediation ontology is used to transform an entity expressed in a given ontology A to an entity expressed in a target ontology B. A first version of this ontology was defined during Testbed-12 to support the semantic mediation service. A number of supporting constructs needed to be introduced such as Rule and Function expressible in RDF. For this Testbed, an update of the ontology was done to leverage the SHACL standard. By leveraging SHACL shapes to select target entities, functions and rules to model transformation, the model is considerably simplified compared to the version developed during Testbed-12. The new ontology introduces only 3 core classes: alignment (**mediation:Alignment**), class mapping (**mediation:ClassMapping**) and property mapping (**mediation:PropertyMapping**) (see Figure 22).

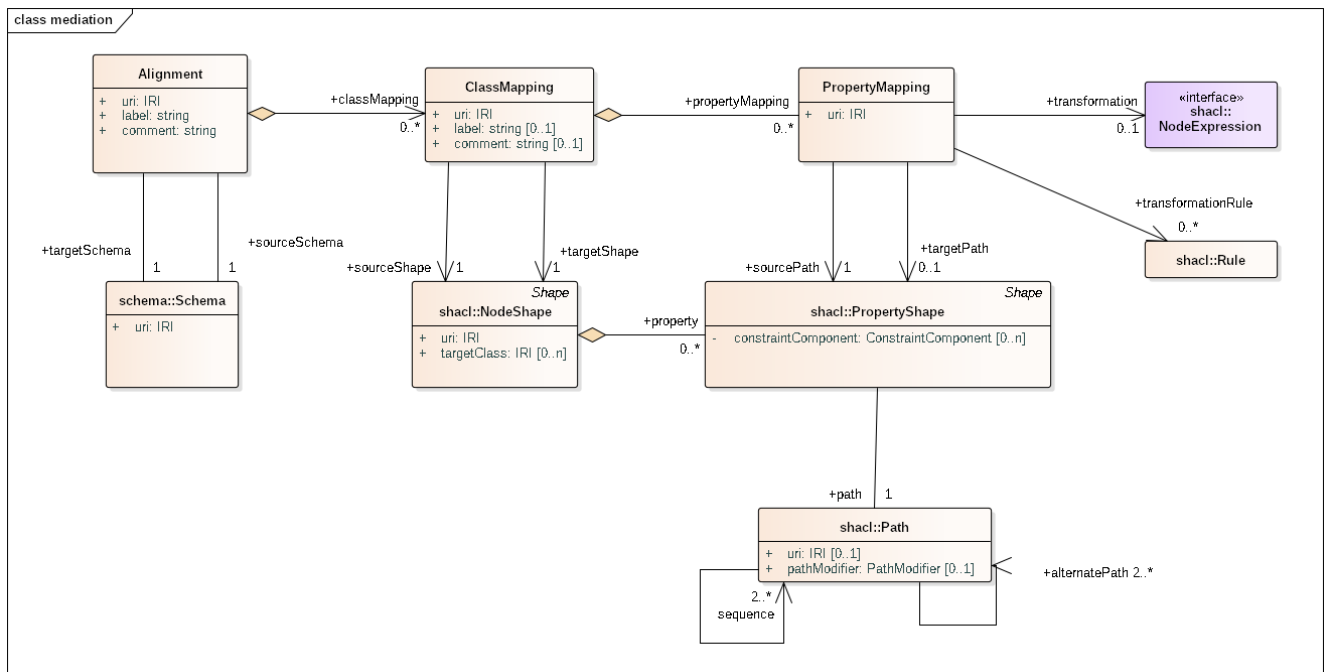


Figure 22. Mediation UML Model

The ontology is defined by the namespace: <http://purl.org/ontology/mediation#> and uses the preferred prefix **mediation**.

11.7.1. Alignment

A semantic alignment (**mediation:Alignment**) defines a mapping between a source (**mediation:sourceSchema**) and a target (**mediation:targetSchema**) schema (**schema:Schema** defined in Testbed-12). A schema can refer to an ontology or an application profile (expressed in SHACL for example). The Application Profile representation is currently being standardized at W3C [16]. A semantic alignment is composed of zero or more class mappings (**mediation:ClassMapping**) using the association (**mediation:classMapping**).

11.7.2. Class Mapping

A **ClassMapping** defines the semantic mapping between two entities types satisfying specific constraints. These constraints on the types and other attributes are defined by SHACL Node Shapes (**shacl:NodeShape**) for the source (**sourceShape** association) and the target entity (**targetShape** association).

A **shacl:NodeShape** is defined by a URI. This URI can be the URI of an **rdfs:Class** or **owl:Class** and the identifier of the shape, which can have one or more **shacl:Target**. In addition, the node shape can be constrained with property shapes (**shacl:PropertyShape**) or other node shape constraint components.

The use of the SHACL Node Shape provides more flexibility than the approach used in Testbed-11 [38]. Node shapes can be defined in application profiles and be reused for the semantic mapping.

11.7.3. Property Mapping

ClassMapping can be associated with zero or more property mapping (**mediation:PropertyMapping**) using the association (**mediation:propertyMapping**). Each **PropertyMapping** refers to a source and target property shape (**shacl:PropertyShape**). The initial version of the ontology was using only **shacl:Path**. However, the approach was not satisfactorily capturing some constraints on the target nodes of the path (such the language of a literal value). **shacl:PropertyShape** requires a **shacl:Path** using the property **shacl:path** and can have one or more **shacl:ConstraintComponents** to express the constraints that the node values needs to satisfied.

The **mediation:PropertyMapping** also supports transformation of values with the SHACL **NodeExpression** defined in the SHACL Advanced Feature specification. This allows the transformation of source values to new target values such as concatenation of multiple fields. For example, **foaf:surname** and **foaf:firstName** properties could be concatenated to a model that accepts only a fullname property such as **schema:fullname..**

Finally, property mapping may require some inference rules to add additional triples into the target entities. For example, if a source entity models a rectangle with a width and height property, we may want to infer in the target ontology that the rectangle is a square if the width and height are equals. SHACL provides an example mechanism to define rules using **SPARQLRule** or simple **shacl:TripleRule**. An **mediation:PropertyMapping** can be associated with zero or more rules using the property **mediation:transformationRule**.

Due to the short duration for this Testbed, the implementation of a mediation engine supporting

this model was not implemented. This should be addressed in future work.

11.8. REST API

A semantic mediation service was implemented during Testbed-12. The service managed schemas and schema mapping, and performed transformation from one schema to another. The service followed REST principles and used JSON-LD and the Hypermedia Application Language (HAL). It is anticipated that a variety of clients may use the Semantic Fusion Service, and as a consequence it is difficult to accommodate the needs of every type of client. The REST API will evolve and be modified as more requirements and features are added to the service. As long as the clients are using the semantics of the link relation types, the hypermedia-driven API provides a robust approach to evolve the API without breaking the client ecosystem. For this reason, the service has adopted a hypermedia-driven RESTful API by default, which meets level 3 of the [Richardson Maturity Model](http://martinfowler.com/articles/richardsonMaturityModel.html) [http://martinfowler.com/articles/richardsonMaturityModel.html].

For this Testbed, a REST-based Fusion Service API was implemented with a POST endpoint to perform correlation between two datasets expressed in different ontologies. Future extensions of the service will implement Create-Read-Update-Delete (CRUD) operations to manage LinkSetSpecification, Alignment and Fusion Policies, and to perform mediation and fusion operations. This could be addressed in future Testbeds.

The POST operation for correlation was accessible at the endpoint /correlate. The endpoint accepted a multipart/form-data with the following key-value pairs that are used to configure the correlation engine described above, in [Correlation Engine](#).

Table 8. Request parameters

parameter name	type	description	Card.
<i>includeStatement</i>	boolean	Boolean indicating if the response include the statements for each correlated pairs	0..1
<i>includeTargets</i>	boolean	Boolean if the closure of correlated node are included in the response	0..1
<i>correlationModel</i>	RDF Model	RDF Model including the LinkSetSpecification to use for correlation	1
<i>sourceModel</i>	RDF Model (RDF/XML, TTL, NTriples)	Source Model to correlate	0..1
<i>targetModel</i>	RDF Model (RDF/XML, TTL, NTriples)	Target Model to correlate	0..1

The response of the POST operations returns an RDF Model with similarity links and optionally the statements and targets node closure (according the setting of the request parameters).

Sample NRCan Data

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix hy: <http://geosciences.ca/def/hydraulic#>.

<https://geoconnex.ca/id/waterbody/60c56a06be4911d892e2080020a0f4c9> a
hy:HY_WaterBody;
  rdfs:label "Riviere Richelieu"@fr,
            "Richelieu River"@en.

<https://geoconnex.ca/id/catchment/020J*CF> a hy:HY_Catchment;
  rdfs:label "Riviere des Iroquois "@fr,
            "Iroquois River"@en.

<https://geoconnex.ca/id/waterbody/1e041853be2d11d892e2080020a0f4c9> a
hy:HY_WaterBody;
  rdfs:label "Riviere L'Acadie"@fr,
            "Acadia River"@en.

<https://geoconnex.ca/id/waterbody/3effba23be2a11d892e2080020a0f4c9> a
hy:HY_WaterBody;
  rdfs:label "Riviere des Hurons"@fr,
            "Huron River"@en.
```

LinksetSpecification NRCan to DBpedia

```
@prefix : <http://www.imagemattersllc.com/testbed15/hydro/correlationModels#> .
@prefix corr: <http://purl.org/ontology/correlation#> .
@prefix void: <http://rdfs.org/ns/void#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix ks: <http://www.usersmarts.com/ont/2005/06/ks#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix prm: <http://www.smartrealm.com/ont/ks/param#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix kspf: <http://www.knowledgesmarts.com/ontologies/pf#> .
@prefix ksfun: <http://www.usersmarts.com/ont/2005/06/ks/functor#> .
@prefix shacl: <http://www.w3.org/ns/shacl#> .
@prefix metric: <http://purl.org/ontology/metrics#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix sim: <http://purl.org/ontology/fusion/similarity#> .
@prefix dash: <http://datashapes.org/dash#> .
@prefix ksd: <http://www.usersmarts.com/ont/2007/02/ks/descriptor#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .

: a owl:Ontology ;
  shacl:declare [ shacl:namespace "http://dbpedia.org/ontology/" ;
                 shacl:prefix "dbo"
               ] ;
  shacl:declare [ shacl:namespace "http://dbpedia.org/resource/" ;
```

```

        shacl:prefix      "dbr"
      ] ;

  shacl:declare [ shacl:namespace "http://www.w3.org/2000/01/rdf-schema#" ;
                  shacl:prefix   "rdfs"
                ] .

:WaterBodyShape a      shacl:NodeShape ;
  shacl:targetClass <http://geosciences.ca/def/hydraulic#HY_WaterBody>,
  <http://geosciences.ca/def/hydraulic#HY_Catchment>.

:RiverInCanadaShape a shacl:NodeShape ;
  shacl:target :RiverInCanadaSPARQLTarget .

:RiverInCanadaSPARQLTarget
  a      shacl:SPARQLTarget ;
  shacl:prefixes : ;
  shacl:select "SELECT ?this \n WHERE { ?this a dbo:BodyOfWater;\n
dbo:sourceCountry dbr:Canada . }" .

:nrcan-dbpedia-correlation-set
  a      corr:CorrelationRuleSet ;
  rdfs:comment "Rule Set to map NRCAN Hydro ontology to DBPedia" ;
  rdfs:label "CorrelationRuleSet NRCAN Hydro-Dbpedia" ;
  corr:correlationRule :RiverCorrelationRule ;
  corr:sourceSchema <http://geosciences.ca/def/hydraulic#> ;
  corr:targetSchema <http://dbpedia.org/ontology/> .

:RiverCorrelationRule
  a      corr:CorrelationRule ;
  corr:sourceShape :WaterBodyShape ;
  corr:targetShape :RiverInCanadaShape ;
  sim:similarityMethod :NameComparison .

:NameComparison a      sim:Comparison ;
  sim:comparator [ a      sim:Comparator ;
                  metric:metric metric:Levenshtein
                ] ;
  sim:sourceProperty [
    sh:path rdfs:label
  ];
  sim:targetProperty [
    sh:path rdfs:label
  ];
  sim:threshold "0.91"^^xsd:double .

metric:Levenshtein a metric:Metric .

:NRCAN-Wikidata-LinkSetSpec
  a      corr:LinkSetSpecification ;
  corr:correlationRuleSet :nrcan-dbpedia-correlation-set ;

```

```

corr:sourceDataset      <http://nrcan.org> ;
corr:targetDataset     <http://dbpedia.org> ;
corr:linkType owl:sameAs.

<http://nrcan.org> a      void:Dataset ;
                    void:sparqlEndpoint <http://localhost:33335/contexts/nrcan/sparql> .

<http://dbpedia.org>
a      void:Dataset ;
void:sparqlEndpoint <https://dbpedia.org/sparql> .

```

Figure 23 shows the correlation of the NRCAN sample with DBPedia without including the DBPedia triples.

The screenshot shows a REST client interface with the following details:

- URL:** localhost:3030/correlation/correlate
- Method:** POST
- Body:** A JSON-LD document with the following structure:


```

{
  "@context": {
    "corr": "http://purl.org/ontology/correlation#",
    "void": "http://rdfs.org/ns/void#",
    "owl": "http://www.w3.org/2002/07/owl#",
    "ks": "http://www.useragents.com/ont/2005/06/ks#",
    "xsd": "http://www.w3.org/2001/XMLSchema#",
    "prn": "http://www.smartrealis.com/ont/xs/param#",
    "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
    "ks:fn": "http://www.knowledgecommons.com/ontologies/ks/func#",
    "ks:fun": "http://www.useragents.com/ont/2005/06/ks/function#",
    "shacl": "http://www.w3.org/ns/shacl#",
    "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
    "sim": "http://purl.org/ontology/union/similarity#",
    "metrics": "http://purl.org/ontology/metrics#",
    "dash": "http://datashapes.org/dash#",
    "ks:desc": "http://www.useragents.com/ont/2007/02/ks/descriptor#",
    "dc": "http://purl.org/dc/elements/1.1/"
  },
  "corr:linkSet": {
    "@id": "http://www.imagemattersllc.com/testbed15/hydro/correlationModels#NRCAN-WikiData-LinkSetSpec",
    "a": "void:Dataset",
    "owl:sameAs": "http://dbpedia.org/resource/Richelieu_River",
    "corr:score": "1.8",
    "corr:source": "http://geoconex.ca/id/waterbody/68c56a86b4911d892e28802ba0f4c9",
    "corr:target": "http://dbpedia.org/resource/Acadia_River"
  },
  "corr:linkSetSpecification": "http://www.imagemattersllc.com/testbed15/hydro/correlationModels#NRCAN-WikiData-LinkSetSpec"
}

```
- Response:** A JSON-LD document with the following structure:


```

{
  "@context": {
    "geoconex": "http://geoconex.ca/id/waterbody/68c56a86b4911d892e28802ba0f4c9",
    "owl:sameAs": "http://dbpedia.org/resource/Richelieu_River"
  },
  "corr:linkSetSpecification": "http://www.imagemattersllc.com/testbed15/hydro/correlationModels#NRCAN-WikiData-LinkSetSpec",
  "a": "void:Dataset",
  "owl:sameAs": "http://dbpedia.org/resource/Acadia_River"
}

```

Figure 23. Request example 1

Figure 24 shows the correlation of the NRCAN sample with DBPedia with enriched DBPedia triples.



Figure 24. Request example 2

11.9. WPS

The REST API for the fusion service can be integrated with a WPS interface. At the time of publishing this ER, the integration was not performed due to lack of time. However, the participants are planning on writing a WPS plugin in GeoServer that will connect to the REST API correlation endpoint of the fusion service. However, the WPS service will not be able to support full CRUD operations. Using the REST API is preferable for making the resource resolvable and to integrate with web clients.

11.10. Web Crawling

This section discusses web crawling of linked data information. While this capability was not implemented during the Testbed due to time constraints and focus on solving data fusion problem, the participants captured some of the requirements and ideas of what a semantic web crawler would look like in the context of data discovery and data integration. The participants assume that a semantic crawler is configured to harvest linked data information about some entities of interests expressed in a given ontology.

The first problem to overcome is how do you determine what data sources to crawl. The initial step is to find if there is a semantic mapping specification from the entity ontology to other ontologies (see below model about LinkedSet specification). These semantic mappings can be managed in a semantic registry (as demonstrated in Testbed-12 for Semantic Mediation Service) or in a vocabulary management service. Once all the mapped ontologies have been identified, we need to identify data sources that use these ontologies. The search of these data sources (datasets) can be done again using the Semantic Registry which describes metadata about the datasets including the spatial scoping, temporal scoping, ontologies used and access information (SPARQL endpoint for example).

Once the data sources with relevant ontologies have been identified, the crawler will query. For each entity to enrich, a query is formulated to each data source using the associated linkedSet specification to return candidate entities to be correlated and fuse. When an item has an owl:sameAs link to an equivalent entity, the crawler can be configured to follow the link and fetch additional triples from the resolvable URL. The triples are aggregated to be further analyzed and fused. The results of the crawler and fusion process can be stored in an RDF repository or returned through an API in an asynchronous manner.

Chapter 12. Future Work

The scope of this Testbed was very broad and attempted to tackle the hard problem of semantic fusion from heterogeneous data sources. A thorough analysis of the integration pipeline was performed, and we managed to formalize the different phases (mediation, correlation, fusion) needed to produce semantic coherent fused entities. Unfortunately, due to time limitations, only the formalization of the correlation and mediation ontology was finished along with the implementation of a SHACL engine and correlation engine. More work is needed to demonstrate the complete integration pipeline. The following are recommendations for future work.

12.1. Semantic Mediation Engine

The implementation of a mediation engine supporting the mediation ontology should be addressed in future work. In particular, the OGC should investigate the transformation from one ontology to another, an application profile to another profile, and be able to transform source query to a target query on the fly by using query rewriting techniques leveraging the semantic mapping specification.

12.2. Fusion Ontology and Fusion Engine

The analysis of conflict resolution needed in the fusion phase was performed in this Testbed. In future Testbeds, the formalization of the different types of conflict and resolution strategies should be formalized. This is so they can be used by a fusion engine to implement the fusion of similar entities coming from multiple sources. The fusion process needs to produce fused entities that are semantically coherent. To be consistent with the other ontologies designed during this Testbed and that favor interoperability and reusability, the ontology should use the SHACL standard. To demonstrate the usage of the fusion ontology, a fusion engine should be implemented to demonstrate the fusion of entities coming out from the correlation phase in the fusion pipeline.

12.3. Integrated Fusion Pipeline

For a future Testbed, a complete fusion pipeline should be demonstrated by leveraging the data to semantic mapping, correlation, mediation and fusion ontologies to perform the integration of entities from multiple sources defined in different ontologies.

12.3.1. Fusion REST Service

For this Testbed, a REST-based Fusion Service API implemented a POST endpoint to perform correlation between two datasets expressed in different ontologies. Future extensions of the service will implement CRUD operations to manage LinkSetSpecification, Alignment and Fusion Policies and perform mediation and fusion operation. This could be addressed in a future Testbed.

12.3.2. Integration of Semantic Data Cubes with Conversational Agent

The focus of this task was to perform the integration of semantic information. The assumption of this Testbed was to use linked data sources (exposed as SPARQL endpoints). For future Testbeds, we

propose integrating Natural Language Processing (NLP) with semantic techniques by integrating semantic data with conversational agents (such as Google Assistant or Alexa). Conversational agent accepts text or speech input and convert the utterances into intents with entities (called slots). The slots can be mapped to semantic concepts used by linked datasets and transformed to GeoSPARQL queries. The results of the query can be transformed into natural language utterances. This integration of linked data with Natural Language understanding will lower the bar to access geospatial data. There is a large volume of data that is available as data cubes and OGC/W3C are considering extension of the W3C RDF Cube standard to support geospatial operations (<https://www.w3.org/TR/qb4st/>). This model can be used as a starting point for future Testbed and be extended to general linked datasets.

Appendix A: Appendix A

A.1. Metric Ontology

```
# baseURI: http://purl.org/ontology/metrics
# prefix: metrics

@prefix : <http://purl.org/ontology/metrics#> .
@prefix metrics: <http://purl.org/ontology/metrics#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://purl.org/ontology/metrics>
  a owl:Ontology ;
  owl:versionInfo "Created with TopBraid Composer" ;
  .
metrics:DistanceMetric
  a owl:Class ;
  rdfs:label "Distance metric" ;
  rdfs:subClassOf metrics:Metric ;
  .
metrics:Metric
  a owl:Class ;
  rdfs:label "Metric" ;
  rdfs:subClassOf owl:Thing ;
  rdfs:subClassOf [
    a owl:Restriction ;
    owl:maxCardinality "1"^^xsd:nonNegativeInteger ;
    owl:onProperty metrics:isNormalized ;
  ] ;
  .
metrics:MetricType
  a owl:Class ;
  rdfs:label "Metric type" ;
  rdfs:subClassOf owl:Thing ;
  .
metrics:MetricUnit
  a owl:Class ;
  rdfs:label "Metric unit" ;
  rdfs:subClassOf owl:Thing ;
  .
metrics:SimilarityMetric
  a owl:Class ;
  rdfs:label "Similarity metric" ;
  rdfs:subClassOf metrics:Metric ;
  .
metrics:isNormalized
```

```

a owl:DatatypeProperty ;
rdfs:label "is normalized" ;
rdfs:range xsd:boolean ;
.
metrics:metric
a owl:ObjectProperty ;
rdfs:label "metric" ;
rdfs:range metrics:Metric ;
.
metrics:metricType
a owl:ObjectProperty ;
rdfs:label "metric type" ;
rdfs:range metrics:MetricType ;
.
metrics:metricUnit
a owl:ObjectProperty ;
rdfs:label "metric unit" ;
rdfs:range metrics:MetricUnit ;
.
metrics:preferredMetricUnit
a owl:ObjectProperty ;
rdfs:label "preferred metric unit" ;
rdfs:range metrics:MetricUnit ;
rdfs:subPropertyOf metrics:metricUnit ;
.

```

A.2. Similarity Ontology

```

# baseURI: http://purl.org/ontology/fusion/similarity
# imports: http://www.w3.org/ns/shacl#
# prefix: sim

@prefix : <http://purl.org/ontology/fusion/similarity#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix sim: <http://purl.org/ontology/fusion/similarity#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://purl.org/ontology/fusion/similarity>
a owl:Ontology ;
rdfs:comment "Similarity Ontology used to perform correlation task" ;
rdfs:label "Similarity Ontology" ;
owl:imports <http://www.w3.org/ns/shacl#> ;
owl:versionInfo "v1" ;
.
sim:Aggregation
a owl:Class ;
rdfs:label "Aggregation" ;

```

```

rdfs:subClassOf sim:SimilarityMethod ;
rdfs:subClassOf [
  a owl:Restriction ;
  owl:cardinality "1"^^xsd:nonNegativeInteger ;
  owl:onProperty sim:aggregator ;
] ;
rdfs:subClassOf [
  a owl:Restriction ;
  owl:minCardinality "0"^^xsd:nonNegativeInteger ;
  owl:onProperty sim:component ;
] ;
.
sim:AggregationComponent
  a owl:Class ;
  rdfs:label "Aggregation component" ;
  rdfs:subClassOf owl:Thing ;
  rdfs:subClassOf [
    a owl:Restriction ;
    owl:cardinality "1"^^xsd:nonNegativeInteger ;
    owl:onProperty sim:similarityMethod ;
  ] ;
  rdfs:subClassOf [
    a owl:Restriction ;
    owl:maxCardinality "1"^^xsd:nonNegativeInteger ;
    owl:onProperty sim:missingWeight ;
  ] ;
  rdfs:subClassOf [
    a owl:Restriction ;
    owl:maxCardinality "1"^^xsd:nonNegativeInteger ;
    owl:onProperty sim:weight ;
  ] ;
.
sim:Aggregator
  a owl:Class ;
  rdfs:label "Aggregator" ;
  rdfs:subClassOf owl:Thing ;
.
sim:Average
  a sim:Aggregator ;
  rdfs:label "Average" ;
.
sim:Comparator
  a owl:Class ;
  rdfs:label "Comparator" ;
  rdfs:subClassOf owl:Thing ;
.
sim:Comparison
  a owl:Class ;
  rdfs:label "Comparison" ;
  rdfs:subClassOf sim:SimilarityMethod ;
  rdfs:subClassOf [

```

```

    a owl:Restriction ;
    owl:cardinality "1"^^xsd:nonNegativeInteger ;
    owl:onProperty sim:comparator ;
  ] ;
  rdfs:subClassOf [
    a owl:Restriction ;
    owl:cardinality "1"^^xsd:nonNegativeInteger ;
    owl:onProperty sim:sourceProperty ;
  ] ;
  rdfs:subClassOf [
    a owl:Restriction ;
    owl:cardinality "1"^^xsd:nonNegativeInteger ;
    owl:onProperty sim:targetProperty ;
  ] ;
.
sim:GeometricMean
  a sim:Aggregator ;
  rdfs:label "Geometric mean" ;
.
sim:Maximum
  a sim:Aggregator ;
  rdfs:label "Maximum" ;
.
sim:Minimum
  a sim:Aggregator ;
  rdfs:label "Minimum" ;
.
sim:QuadraticMean
  a sim:Aggregator ;
  rdfs:label "Quadratic mean" ;
.
sim:SimilarityMethod
  a owl:Class ;
  rdfs:comment "Abstract base class for similarity methods. There are two kinds of
similarity methods: Comparison and Aggregation. This is class is needed to allow
nested composition." ;
  rdfs:label "Similarity method" ;
  rdfs:subClassOf owl:Thing ;
  rdfs:subClassOf [
    a owl:Restriction ;
    owl:maxCardinality "1"^^xsd:nonNegativeInteger ;
    owl:onProperty sim:threshold ;
  ] ;
.
sim:aggregator
  a owl:ObjectProperty ;
  rdfs:label "aggregator" ;
  rdfs:range sim:Aggregator ;
.
sim:comparator
  a owl:ObjectProperty ;

```

```

    rdfs:label "comparator" ;
    rdfs:range sim:Comparator ;
.
sim:component
  a owl:ObjectProperty ;
  rdfs:label "component" ;
  rdfs:range sim:AggregationComponent ;
.
sim:missingWeight
  a owl:DatatypeProperty ;
  rdfs:label "missing weight" ;
  rdfs:range xsd:decimal ;
.
sim:similarityMethod
  a owl:ObjectProperty ;
  rdfs:label "similarity method" ;
  rdfs:range sim:SimilarityMethod ;
.
sim:sourceProperty
  a owl:ObjectProperty ;
  rdfs:label "source property" ;
  rdfs:range <http://www.w3.org/ns/shacl#PropertyShape> ;
.
sim:targetProperty
  a owl:ObjectProperty ;
  rdfs:label "target property" ;
  rdfs:range <http://www.w3.org/ns/shacl#PropertyShape> ;
.
sim:threshold
  a owl:DatatypeProperty ;
  rdfs:label "threshold" ;
  rdfs:range xsd:decimal ;
.
sim:weight
  a owl:DatatypeProperty ;
  rdfs:label "weight" ;
  rdfs:range xsd:decimal ;
.

```

A.3. Mediation Ontology

```

# baseURI: http://purl.org/ontology/mediation
# imports: http://www.w3.org/ns/shacl#

@prefix : <http://purl.org/ontology/mediation#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix sh: <http://www.w3.org/ns/shacl#> .

```

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
```

```
<http://purl.org/ontology/mediation>
```

```
  a owl:Ontology ;  
  owl:imports sh: ;  
  owl:versionInfo "1.0" ;
```

```
.
```

```
:Alignment
```

```
  a owl:Class ;  
  a sh:NodeShape ;  
  rdfs:comment "An alignment defines the rule of transformation from one schema to  
  another. A schema can be an ontology or an application profile" ;
```

```
  rdfs:label "Alignment" ;  
  rdfs:subClassOf owl:Thing ;  
  rdfs:subClassOf [  
    a owl:Restriction ;  
    owl:cardinality "1"^^xsd:nonNegativeInteger ;  
    owl:onProperty :sourceSchema ;  
  ] ;
```

```
  rdfs:subClassOf [  
    a owl:Restriction ;  
    owl:cardinality "1"^^xsd:nonNegativeInteger ;  
    owl:onProperty :targetSchema ;  
  ] ;
```

```
  rdfs:subClassOf [  
    a owl:Restriction ;  
    owl:minCardinality "0"^^xsd:nonNegativeInteger ;  
    owl:onProperty :classMapping ;  
  ] ;
```

```
.
```

```
:ClassMapping
```

```
  a owl:Class ;  
  rdfs:label "Class mapping" ;  
  rdfs:subClassOf owl:Thing ;  
  rdfs:subClassOf [  
    a owl:Restriction ;  
    owl:cardinality "1"^^xsd:nonNegativeInteger ;  
    owl:onProperty :sourceNode ;  
  ] ;
```

```
  rdfs:subClassOf [  
    a owl:Restriction ;  
    owl:cardinality "1"^^xsd:nonNegativeInteger ;  
    owl:onProperty :targetNode ;  
  ] ;
```

```
  rdfs:subClassOf [  
    a owl:Restriction ;  
    owl:minCardinality "0"^^xsd:nonNegativeInteger ;  
    owl:onProperty :propertyMapping ;  
  ] ;
```

```
.
```

```
:PropertyMapping
```

```

a owl:Class ;
  rdfs:comment "A property mapping maps values from a source path to target path. The
values can be calculated using transformation" ;
  rdfs:label "Property mapping" ;
  rdfs:subClassOf owl:Thing ;
  rdfs:subClassOf [
    a owl:Restriction ;
    owl:cardinality "1"^^xsd:nonNegativeInteger ;
    owl:onProperty :targetProperty ;
  ] ;
  rdfs:subClassOf [
    a owl:Restriction ;
    owl:maxCardinality "0"^^xsd:nonNegativeInteger ;
    owl:onProperty :transformation ;
  ] ;
  rdfs:subClassOf [
    a owl:Restriction ;
    owl:maxCardinality "1"^^xsd:nonNegativeInteger ;
    owl:onProperty :sourceProperty ;
  ] ;
  rdfs:subClassOf [
    a owl:Restriction ;
    owl:minCardinality "0"^^xsd:nonNegativeInteger ;
    owl:onProperty :transformationRule ;
  ] ;
  rdfs:subClassOf [
    a owl:Restriction ;
    owl:minCardinality "0"^^xsd:nonNegativeInteger ;
    owl:onProperty sh:rule ;
  ] ;
.
:classMapping
  a owl:ObjectProperty ;
  rdfs:label "class mapping" ;
  rdfs:range :ClassMapping ;
.
:propertyMapping
  a owl:ObjectProperty ;
  rdfs:comment "propertyMapping can be used to specify that a class mapping has a
given property mapping" ;
  rdfs:domain :ClassMapping ;
  rdfs:label "property mapping" ;
  rdfs:range :PropertyMapping ;
.
:sourceNode
  a owl:ObjectProperty ;
  rdfs:label "source node" ;
  rdfs:range [
    a owl:Class ;
    owl:unionOf (
      sh:NodeShape

```

```

        owl:Class
    ) ;
] ;
.
:sourceProperty
  a owl:ObjectProperty ;
  rdfs:comment "Source values defined a SHACL Node expression or Resource" ;
  rdfs:label "source path" ;
  rdfs:range rdfs:Resource ;
.
:sourceSchema
  a owl:ObjectProperty ;
  rdfs:label "source ontology" ;
  rdfs:range owl:Ontology ;
.
:targetNode
  a owl:ObjectProperty ;
  rdfs:label "target node" ;
  rdfs:range [
    a owl:Class ;
    owl:unionOf (
      sh:NodeShape
      owl:Class
    ) ;
  ] ;
.
:targetProperty
  a owl:ObjectProperty ;
  rdfs:comment "Target values defined a SHACL Node expression or a Property" ;
  rdfs:label "target path" ;
  rdfs:range rdfs:Resource ;
.
:targetSchema
  a owl:ObjectProperty ;
  rdfs:label "target ontology" ;
  rdfs:range owl:Ontology ;
.
:transformation
  a owl:ObjectProperty ;
  rdfs:comment "SHACL Node expression transforming source value nodes to target nodes.
Accept a constant or function expression" ;
  rdfs:label "transformation" ;
  rdfs:range rdfs:Resource ;
.
:transformationRule
  a owl:ObjectProperty ;
  rdfs:label "transformation rule" ;
  rdfs:range sh:Rule ;
.

```


Appendix B: Revision History

Table 9. Revision History

Date	Editor	Release	Primary clauses modified	Descriptions
2019-05-31	E. Kok	.1	all	initial version
2019-10-31	E. Kok	1.0	all	issue for submission as pending to OGC TC Meeting
2019-11-11	E. Kok	1.1	all	revision based on internal feedback
2019-12-12	G. Hobona	1.1	all	OGC staff final review and edits

Appendix C: Bibliography

1. Fellah, S.: OGC® Testbed-11 Incorporating Social Media in Emergency Response Engineering Report. Open Geospatial Consortium, https://portal.opengeospatial.org/files/?artifact_id=64385 (2015).
2. Fellah, S.: Testbed-12 Semantic Portrayal, Registry and Mediation Engineering Report. Open Geospatial Consortium, <http://docs.opengeospatial.org/per/16-059.html> (2016).
3. Gobe Hobona, R.B.: Testbed-12 Catalogue and SPARQL Engineering Report. Open Geospatial Consortium, <http://docs.opengeospatial.org/per/16-059.html> (2016).
4. Holger Knublauch: OWL and SHACL Compared, <https://spinrdf.org/shacl-and-owl.html>, (2017).
5. Knublauch, H., Kontokostas, D.: Shapes constraint language (SHACL). W3C Recommendation. 20, (2017).
6. Grimes, S.: Unstructured data and the 80 percent rule. Carabridge Bridgepoints. 10 (2008).
7. Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805. (2018).
8. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners. OpenAI Blog. 1, (2019).
9. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692. (2019).
10. Berners-Lee, T., Hendler, J., Lassila, O., others: The semantic web. Scientific american. 284, 28–37 (2001).
11. Bizer, C., Heath, T., Berners-Lee, T.: Linked data: The story so far. In: Semantic services, interoperability and web applications: emerging concepts. pp. 205–227. IGI Global (2011).
12. Tim Berners-Lee: Linked Data, <https://www.w3.org/DesignIssues/LinkedData.html>, (2019).
13. Hobona, G., Brackin, R.: OWS-8 CCI Semantic Mediation Engineering Report. Open Geospatial Consortium, https://portal.opengeospatial.org/files/?artifact_id=46342 (2011).
14. Hobona, G., Brackin, R.: OWS-9 CCI Semantic Mediation Engineering Report. Open Geospatial Consortium, https://portal.opengeospatial.org/files/?artifact_id=51840&version=1 (2012).
15. Ingo Simonis, S.F.: OGC® Testbed 10 Cross Community Interoperability (CCI) Ontology Engineering Report. Open Geospatial Consortium, https://portal.opengeospatial.org/files/?artifact_id=58974 (2014).
16. Atkinson, Coyle, Isaac, Car: Profile Guidance W3C Editor’s Draft 18 October 2019, <https://w3c.github.io/dxwg/profiles/>, (2019).
17. Volz, J., Bizer, C., Gaedke, M., Kobilarov, G.: Silk-a link discovery framework for the web of data. LDOW. 538, 53 (2009).
18. Isele, R., Jentzsch, A., Bizer, C.: Efficient multidimensional blocking for link discovery without losing recall. In: WebDB (2011).
19. Ngomo, A.-C.N., Auer, S.: LIMES—a time-efficient approach for large-scale link discovery on the web of data. In: Twenty-Second International Joint Conference on Artificial Intelligence (2011).

20. Kleanthi Georgala: LIMES, <http://aksw.org/Projects/LIMES.html>, (2018).
21. Ngomo, A.-C.N.: Link discovery with guaranteed reduction ratio in affine spaces with minkowski measures. In: International Semantic Web Conference. pp. 378–393. Springer (2012).
22. Ngomo, A.-C.N.: On link discovery using a hybrid approach. Journal on Data Semantics. 1, 203–217 (2012).
23. Ngomo, A.-C.N.: ORCHID–reduction-ratio-optimal computation of geo-spatial distances for link discovery. In: International Semantic Web Conference. pp. 395–410. Springer (2013).
24. Ngomo, A.-C.N.: Helios–execution optimization for link discovery. In: International Semantic Web Conference. pp. 17–32. Springer (2014).
25. Ngomo, A.-C.N., Lyko, K.: Eagle: Efficient active learning of link specifications using genetic programming. In: Extended Semantic Web Conference. pp. 149–163. Springer (2012).
26. Sherif, M.A., Ngomo, A.-C.N., Lehmann, J.: Wombat - a generalization approach for automatic link discovery. In: European Semantic Web Conference. pp. 103–119. Springer (2017).
27. Bleiholder, J., Naumann, F.: Data Fusion. ACM Comput. Surv. 41, 1:1–1:41 (2009).
28. Garcia-Molina, H.: Database systems: the complete book. Pearson Education India (2008).
29. Bleiholder, J., Naumann, F.: Declarative data fusion–syntax, semantics, and implementation. In: East European Conference on Advances in Databases and Information Systems. pp. 58–73. Springer (2005).
30. Arenas, M., Bertossi, L., Chomicki, J.: Consistent query answers in inconsistent databases. In: PODS. pp. 68–79. Citeseer (1999).
31. Fellah, S.: OGC Testbed-14: Characterization of RDF Application Profiles for Simple Linked Data Application and Complex Analytic Applications Engineering Report. Open Geospatial Consortium, <http://docs.opengeospatial.org/per/18-094r1.html> (2018).
32. Microsoft: TypeScript Language, <https://www.typescriptlang.org/>, (2019).
33. Kotlin Foundation: Kotlin Language, <https://kotlinlang.org/>, (2019).
34. Michelfeit, J., Knap, T., Nečaský, M.: Linked data integration with conflicts. arXiv preprint arXiv:1410.7990. (2014).
35. Alexander, K., Cyganiak, R., Hausenblas, M., Zhao, J.: Describing linked datasets with the void vocabulary. (2011).
36. Hodgson, R., Keller, P.J., Hodges, J., Spivak, J.: QUDT-quantities, units, dimensions and data types ontologies. USA Available <http://qudt.org> March. (2014).
37. wikipedia: Levenshtein Distance, https://en.wikipedia.org/wiki/Levenshtein_distance, (2019).
38. Fellah, S.: Testbed-11 Implementing Linked Data and Semantically Enabling OGC Services Engineering Report. Open Geospatial Consortium, https://portal.opengeospatial.org/files/?artifact_id=64405 (2015).