KARADENIZ TECHNICAL UNIVERSITY
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

GEOMATICS ENGINEERING DEPARTMENT

A FRAMEWORK FOR ONTOLOGY-BASED SPATIAL DATA QUALITY ASSESSMENT, DESIGN AND DEVELOPMENT

DOCTOR OF PHILOSOPHY THESIS

Cemre YILMAZ, M.Sc. Eng.

SEPTEMBER 2018
TRABZON

KARADENİZ TECHNICAL UNIVERSITY
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

GEOMATICS ENGINEERING DEPARTMENT

A FRAMEWORK FOR ONTOLOGY-BASED SPATIAL DATA QUALITY ASSESSMENT, DESIGN AND DEVELOPMENT

Cemre YILMAZ

This thesis is accepted to give the degree of
DOCTOR OF PHILOSOPHY
By
The Graduate School of Natural and Applied Sciences at
Karadeniz Technical University

The Date of Submission	: 25 /05 /2018
The Date of Examination	: 10 /09 /2018

Thesis Supervisor : Prof . Dr. Çetin CÖMERT

Trabzon 2018

KARADENİZ TECHNICAL UNIVERSITY

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

GEOMATICS ENGINEERING DEPARTMENT

Cemre YILMAZ, M.Sc.Eng

A FRAMEWORK FOR ONTOLOGY-BASED SPATIAL DATA QUALITY ASSESSMENT, DESIGN AND DEVELOPMENT

Has been accepted as a thesis of

DOCTOR OF PHILOSOPHY

after the Examination by the Jury Assigned by the Administrative Board of the Graduate School of Natural and Applied Sciences with the Decision Number 1763 dated 24 / 07 / 2018

Approved By

Chairman	:	Prof. Dr. Çetin CÖMERT	…...…………………….....………
Member	:	Prof. Dr. Mustafa TÜRKER	…...…………....…………………
Member	:	Prof. Dr. Cemal KÖSE	……....……………....……………
Member	:	Assoc. Prof. Dr. Halil AKINCI	....
Member	:	Assoc. Prof. Dr. Emine TANIR KAYIKÇI	....

Prof. Dr. Sadettin KORKMAZ

Director of Graduate School

ACKNOWLEDGEMENTS

This thesis would not have been possible without the help of certain people. With immense gratitude, I acknowledge the support and help of my advisor Prof. Dr. Çetin CÖMERT. I thank to my thesis committee members Prof. Dr. Cemal KÖSE and Assoc. Dr. Emine TANIR KAYIKÇI, for their support.

Also, I would like to express my special gratitude to my colleagues and friends, in particular Asst. Prof. Dr. Gülten KARA, Asst. Prof. Dr. Deniztan ULUTAŞ KARAKOL, Asst. Prof. Dr. Leyla ÇAKIR.

Many thanks to Dr. David VAZ for his support in the Logical Programming part of this thesis.

I thank my dear friend Asst. Prof. Dr. Şebnem ATASARAL for her cheerful attitude and friendship.

Lastly, I owe many thanks to my fiancée Res. Ass. Deniz YILDIRIM for his patience, smiling face and endless support. I would like to thank to my precious siblings Çiğdem and Emre YILMAZ for their help, for those many sleepless nights they stood by my side, and for their constant moral support. I thank to my father and mother for their patience. It is to my family that this thesis is dedicated to.

Cemre YILMAZ

Trabzon 2018

THESIS STATEMENT

I hereby declare that all information in this thesis titled as “A Framework for Ontology-Based Spatial Data Quality Assessment, Design and Development” has been completed under the responsibility of my supervisor Prof. Dr. Çetin CÖMERT and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work. 10/09/2018

Cemre YILMAZ

TABLE OF CONTENTS

PhD. Thesis

SUMMARY

A FRAMEWORK FOR ONTOLOGY-BASED SPATIAL DATA QUALITY ASSESSMENT, DESIGN AND DEVELOPMENT

Cemre YILMAZ

Karadeniz Technical University
The Graduate School of Natural and Applied Sciences
Geomatics Engineering Graduate Program
Supervisor: Prof. Dr. Çetin CÖMERT
2018, 137 Pages

Spatial data is used for operations involving decision-making analysis and spatial calculations, in various domains such as cadastre, agriculture and risk analysis. Institutions produce data and share with the NSDI and should control whether they are conformant with the specifications. Spatial data quality is an essential aspect in domains for the accuracy of the results. Quality assessment is based on the conformance of data to its specifications or fitness for users' purpose. These specifications include the rules and constraints that a dataset should comply with. Traditional quality assessment software and solutions in literature are mostly proprietary and rule-based. The rules are not reusable, shareable or interoperable; every software has its own format of rules. Updates require rewriting of rules. To overcome these shortcomings, a new open source-based methodology is required that can be applicable to spatial data in general, independent from the domain. Purpose of this thesis is to design a domain independent web-based data quality assessment framework. Logic programming and ontology-based methods are proposed. An ontology-based spatial data quality assessment framework is designed. For the purposes of the thesis, two types of ontologies are introduced. These are; the specification ontologies and the Spatial Data Quality Ontology. A specification ontology is to be created by users to define rules according to specifications. Spatial Data Quality Ontology is responsible with quality assessment; it is domain independent and used for quality assessment based on the rules defined by any specification ontology for the related domain. The framework's applicability to a spatial dataset against relevant quality metrics and rules are proven with case studies.

Keywords: Spatial data quality, OWL, Prolog, data quality assessment, SWRL, quality ontology.

Doktora Tezi

ÖZET

KONUMSAL VERİ KALİTESİNİN ONTOLOJİ-TABANLI DEĞERLENDİRİLMESİ İÇİN BİR ÇATI TASARIMI VE GELİŞTİRİLMESİ

Cemre YILMAZ

Karadeniz Teknik Üniversitesi
Fen Bilimleri Enstitüsü
Harita Mühendisliği Anabilim Dalı
Danışman: Prof. Dr. Çetin CÖMERT
2018, 137 Sayfa

Konumsal veriler; kadastro, tarım, risk analizi gibi alanlarda, karar verme analizleri ve konumsal hesaplamaları içeren işlemler için kullanılmaktadır. Kurumlar veri üreterek bu verileri UKVA' da paylaşır ve verilerin mevcut belirtimlere uygunluğunu test etmeleri gerekir. Konumsal veri kalitesi sonuçların doğruluğu açısından birçok alan için önemli bir konudur. Konumsal veri kalitesinin değerlendirilmesi, belirtimlere ve kullanıcı hedeflerine uygunluğu temel almaktadır. Bu belirtimler, veri kümelerinin uyması gereken kural ve kısıtlamaları içerir. Geleneksel yazılımlar ve literatürdeki çözümler ticari ve çoğunlukla kural tabanlıdır. Yeniden kullanılabilirlik ve birlikte işlerlik sağlanmamıştır. Yazılımların kendilerine özgü kural biçemleri vardır. Güncellemeler, kuralların yeniden üretilmesini gerektirmektedir. Bu eksiklikleri gidermek için, konumsal verilere yönelik, alan-bağımsız ve açık kaynak tabanlı bir metodoloji gerekmektedir. Tezin hedefi, alan-bağımsız, İnternet-tabanlı veri kalitesi değerlendirme çerçevesi tasarlamaktır. Mantık ve ontoloji tabanlı yöntemler ortaya koyulmuştur. Ontoloji tabanlı veri kalitesi değerlendirme çerçevesi tasarlanmıştır. Bu çerçevede, tezin hedefine yönelik, iki tür ontoloji ortaya koyulmuştur. Bunlar; belirtim ontolojileri ve Konumsal Veri Kalitesi Ontolojisi'dir. Belirtim ontolojileri, kullanıcılar tarafından, alandaki belirtimlere yönelik kuralları tanımlamak için oluşturulur. Konumsal Veri Kalitesi Ontolojisi, değerlendirmeden sorumludur ve tasarımı, alan-bağımsız olarak, belirtim ontolojileri tarafından içe aktarılıp kalite değerlendirilmesinin doğru bir biçimde gerçekleştirilmesine yöneliktir. Önerilen çerçevenin konumsal verilere uygulanabilirliği, örnek çalışmalarla ortaya koyulmuştur.

Anahtar Kelimeler: Konumsal veri kalitesi, OWL, Prolog, veri kalite değerlendirmesi, SWRL, kalite ontolojisi.

LIST OF FIGURES

<u>Page Nr. aa</u>

LIST OF TABLES

LIST OF ABBREVIATIONS

API	: Application Programming Interface
CSV	: Comma-Separated Values
CWA	: Closed World Assumption
DL	: Description Logic
DE-9IM	: Dimensionally Extended 9-Intersection Model
FOL	: First Order Logic
GeoSPARQL	: A Geographic Query Language for RDF Data
GCM	: Turkish General Command of Mapping
GIS	: Geographic Information Systems
GML	: Geography Markup Language
HTML	: Hypertext Markup Language
IC	: Integrity Constraint
ICA	: International Cartography Association
INSPIRE	: Infrastructure for Spatial Information in Europe
IRI	: Internationalized Resource Identifier
ISO	: International Organization for Standardization
ISO TC 211	: ISO Technical Committee 211
JTS	: Java Topology Suite
LOD	: Linked Open Data
LSMMPR	: Turkish Large-Scale Map and Map Production Regulation
NAF	: Negation as Failure
NSDI	: National Spatial Database Infrastructure
NTDB	: National Topographic Database
OBDA	: Ontology-Based Data Access
ODP	: Ontology Design Pattern
OGC	: Open GIS Consortium
OS	: Ordnance Survey
OSM	: OpenStreetMap
OWA	: Open World Assumption
OWL	: Web Ontology Language

R2RML	: RDB to RDF Mapping Language
RCC8	: Region Connection Calculus 8
RDF	: Resource Description Framework
RDF-S	: RDF Schema
RIF	: Rule Interchange Format
RuleML	: Rule Markup Language
SDI	: Spatial Data Infrastructure
SDQO	: Spatial Data Quality Ontology
SHACL	: Shapes Constraint Language
SfO	: Specification Ontology
SPARQL	: SPARQL Protocol and RDF Query Language
SPIN	: SPARQL Inference Notation
SQUIRL	: Spatial Quality Integration Rules Language
SQWRL	: Semantic Query-Enhanced Web Rule Language
SWRL	: Semantic Web Rule Language
TURTLE	: Terse RDF Triple Language
UDI	: User Defined Indexing
UNA	: Unique Name Assumption
URI	: Uniform Resource Identifier
VGI	: Volunteered Geographic Information
W3C	: World Wide Web Consortium
WFS-T	: Transactional Web Feature Service
XML	: Extensible Markup Language
XSD	: XML Schema Definition

1. GENERAL INFORMATION

1.1. Introduction

Spatial data are among primary objects in the work done by many institutions. These institutions, even if they are part of one government or company, may have independent, occasionally conflicting structure and regulations, despite having common objects of interest. If the spatial data are produced by an organization in a manner not fully compliant to the needs of another organization, the latter organization may have to look for other ways to obtain relevant data. One more point harming their interoperability lies in the de facto solutions in practice; due to ignorance, convenience or the inefficiency of the rules, technicians or engineers may produce their own temporary solutions when they encounter problems.

Many regulations are based on the practices of the past. For high quality work, the regulations need to stay efficient and up to date; furthermore, the work needs to be examined under regular inspections. The world also becomes more social and technologically advanced every day; the interoperability and quality assessment aspects become more and more necessary and at the same time more conveniently applied. Currently, the poor quality in data costs trillions of dollars every year (Hamish, 2012). There is a need to test the compliance of data to the intended usage.

There are proprietary solutions for spatial data quality assessment. 1Spatial's 1Validate and ESRI's ArcMap Data Reviewer can be given as examples (Sanderson et al., 2009; URL-1, 2017). These propose rule-based means to define the rules that a dataset should comply with. Although they propose solution for the same problem, the way of defining rules and the rules categorization are different. To provide interoperability, 1Spatial proposed 1Integrate for ArcGIS. These solutions are proprietary and platform dependent. These are the shortcomings of such solutions.

In literature, there are several studies for quality assessment, Mostafavi et al. (2004); Wang et al. (2005) are examples to ontology -based studies for spatial data quality. Although they ensure reusability for the proposed domain, they are domain-dependent.

Debattista et al. (2016); Fürber and Hepp (2011); Geisler et al. (2016) are examples of ontology-based solutions for data quality management. They propose ontologies for quality management. Although they provide solutions for many domains, they do not propose solution for spatial data. For example, Geisler et al. (2016b) devise ontology for data streams, Debattista et al. (2016b) propose dataset quality ontology for linked data.

The purpose of this thesis is to investigate how to design a reusable, domain-independent, web and open-source based quality assessment framework.

The Closed World Assumption (CWA) implies that no information can be added to a system, anything not declared or known to be true is considered as false. In reality, the world is open; information, one is oblivious to, is not necessarily wrong or true. The world has stagnant elements, but also dynamic elements.

Historically, ontology is the study of being and becoming, the still and the dynamic, and their nature of structure. Not independent from the historical meaning, the same term is used in engineering disciplines and sciences for "specifications of conceptualizations" (Gruber, 1995). Naturally, ontologies in scientific applications, in line with Open World Assumption (OWA).

In 2001, sixteen years after World Wide Web, Tim Berners-Lee, together with Lassila and Hendler, using ontologies and the idea of ontology as backbone, introduced the concept of "Semantic Web", a new worldwide Web, where information is processed at the level of meaning (Berners-Lee et al., 2001). A meaningful web is also more machine-operable enabling standardized solutions to the related tedious problems. These standards for solutions are also established using ontologies. Ontologies are part of the one Semantic Web; any ontology can be linked to the other ones by the way of importing. They fit well with the purpose of producing interoperable, reusable, extensible and customizable solutions.

In this thesis, several systems for quality assessment are introduced, with CWA or OWA. A logical programming-based application using the programming language Prolog is done for inspecting the Closed World approach. For the spatial data quality assessment with ontologies, a Web Ontology Language (OWL) based framework is introduced.

Data quality assessment process in the designed framework has three steps. a) Defining the relevant data quality elements, metrics and rules according to the needs such as, specifications. b) Data quality assessment. c) Creating a data quality report including the results of quality assessment.

Within the framework, two types of ontologies are developed; specification ontologies (SfOs) for specifications with common points, and the Spatial Data Quality Ontology (SDQO). SfOs ontologies are different, depending on the domains and used in accordance with SDQO to provide a general rule-based system for quality assessment. A domain expert can define the rules with the help of a graphical user interface. As the data is expected to be error-prone, the framework is to be robust against expected errors and the causes of such errors are to be exploited. Furthermore, the SfOs are designed to be especially simple and easy to manipulate; they will be primarily used by domain experts that may fail to have expertise in ontologies and specifically OWA.

1.1.1. Definition of the Problem

The starting point of this thesis is the assessment of spatial data quality, conformance to related specifications or users' intended usage. Many institutions in public and private sectors produce spatial data according to their regulations. Conformance of the produced data to its specifications is to be provided by the producers. Rule-based methodology is commonly used for this purpose. Implementation of rule-based methodologies is mostly dependent on the used technologies. Main problem to be addressed in such implementation is to construct a framework in a reusable, interoperable, domain-independent and open source-based way. Improvements in web technologies affect the spatial data management methods and tools, as it so happens in many different domains. Semantic Web technologies are becoming more and more important because of using semantics of data in the applications including the Spatial Database Infrastructures (SDIs). There is an inclination for creating new aspects and technologies for quality assessment of spatial data. Therefore, Semantic Web technologies are considered for the implementation of the framework. Following research points define the scope of this thesis.

- To what extent the framework needs a complete classification of data quality checks.
- How to express constraints in the specifications in a formal way.
- Should one use a logic programming (Prolog) based or ontology-based (OWL) approach?
- How to enable extendibility of the framework?
- How to ensure reusability within the framework?

- Can the framework be implemented with open source components? What are the available tools for web-based implementation?

1.1.2. Objective of the Study

In this study, the purpose is to design and apply a framework with the use of semantic technologies for spatial data quality assessment. In the scope of this thesis, quality assessment aspects are related with the internal data quality that is, conformance of data to its specifications. A framework, applicable to a range of data application schemas is the main quality aspect of the proposed system. Proof of the proposed system, against the relevant data quality elements is subject of this thesis.

1.1.3. Methodology

Following methodology is applied to implement this study.

- Literature research including standards related with Spatial Data Quality (SDQ) and academic studies to determine applicable data quality elements and quality metrics to test spatial data conformance to its specifications.
- Research of the spatial specifications; international and national levels e.g. Infrastructure for Spatial Information in Europe (INSPIRE) data specifications, Large-Scale Map and Map Production Regulation (LSMMPR). Defining the common constraints to be tested independent of the application domain.
- Formalisation of the constraints defined in the regulations; Closed World (Prolog-based) and Open World (OWL-based) applications are applied.
- Research on the current technologies for rules implementation in both Open World (OWL-based) and Closed World (Prolog-based) scenarios in spatial domain.
- Research on the data access/transform technologies for implementation of both Closed World and Open World-based applications.
- Creation of reports for quality results.
- Web based implementation of the proposed framework.

1.2. Preliminaries

In this section, background for the related research in the thesis is explained. Main subjects of the research are; Spatial Data Quality including its related standards, elements and metrics, Open World and Closed World concepts for quality assessment. Semantic Web related definitions are explained with regards to Open World implementation. Logic Programming related definitions are explained with regards to Closed World implementation.

1.2.1. Data Quality

Data quality, depending on the viewpoint, can be defined in various ways; for producers and users as pointed out in researches (Devillers and Jeansoulin, 2013; Herzog et al., 2007; ISO, 2013). Juran (1974) describe the data quality as fitness for use, a common users' side definition. Strong et al. (1997) stress the importance of quality for the contextual aspect also can be defined as external quality. Wand and Wang (1996), Batini and Scannapieca (2006) consider the internal data quality, the producers' side, such as consistency between data and the specifications. Large number of data quality criteria and elements are defined in literature towards the different kinds of users' and producers' needs. The concept and the relation between the "produced data" and its internal and external quality dimensions is shown in Figure 1. In this study, internal data quality is considered.

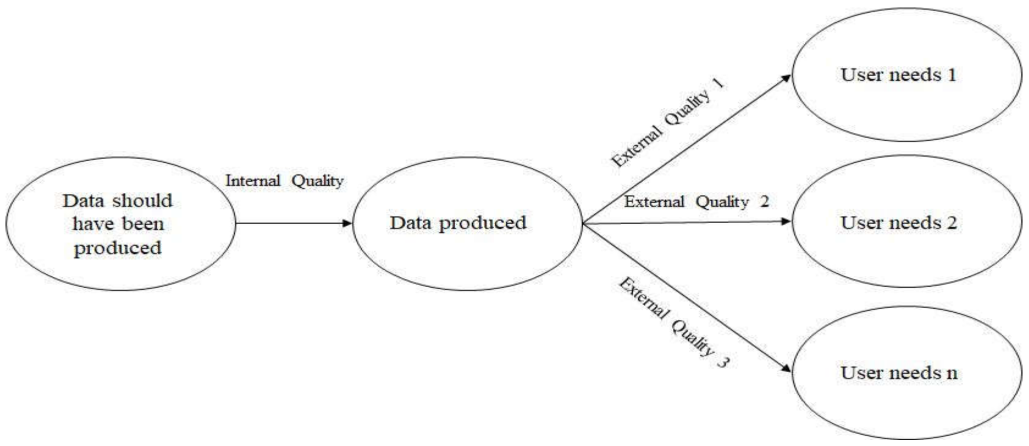


Figure 1. Concepts of internal and external data quality (Devillers and Jeansoulin, 2010).

Data quality elements are to be changed from domain to domain. Volunteered geographic data (VGI), spatial data infrastructures (SDI), linked data (LD) or open data (OD) have common data quality elements but also have different quality aspects and elements.

Michael F Goodchild (1995); Guptill and Morrison (2013); ISO (2013); NCDCDS (1987); Servigne et al. (2000); Veregin (1999) define the internal and external spatial data quality elements for spatial data. Zaveri et al. (2016) review data quality criteria and assessment methods for linked open data (LOD). International Organization for Standardization (ISO) has published several standards for the geospatial quality, latest being ISO 19157:2013 (ISO, 2013). Fonte et al. (2017) discuss the sufficiency of the standard for VGI data quality assessment and classifies additional data quality indicators which are specific to VGI because of its nature. Reliability which is data-based quality indicator, socio-economic quality indicators, contributor indicators are proposed in addition to the ISO 19157:2013 standard classification.

Throughout the recorded history, production of the highest possible quality, more accurate, more consistent, more complete maps and spatial data is desired by geospatial data producers and users. In the modern era, in late 1980s and 1990s, committees had met to define quality aspects for spatial data. United States National Committee on Digital Cartographic Data Standards developed quality related standards for the American Congress of Surveying and Mapping (ACSM) in 1983. The need of specifying the data quality of cartographic datasets and standards for common quality explanation are stated in the standard. In 1987, “A Draft Proposed Standard for Digital Cartographic Data” was published (NCDCDS, 1987) and this standard includes data quality elements. The International Cartography Association (ICA) have met several times for developing data quality criteria, methods for quality evaluation and they established the ICA Commission on Spatial Data Quality (Guptill and Morrison, 2013). These proposals and the ISO 9000 (Quality Management System- Fundamentals and Vocabulary) standard have guided the data quality evaluation. In 2000s, ISO 19113:2002, ISO 19114:2003, ISO 19115:2003 which is currently revised to ISO 19115:2014 and ISO/TS 19138:2006 were published to standardize the principles of spatial data quality, evaluation and measures and publish metadata (ISO, 2002, 2003, 2005, 2013; Joos, 2006). ISO (2013) combines the previous standards; it is the current standard for spatial data quality elements, evaluation methods and quality measures.

Basic concepts, data quality element, data quality scope and the data quality report, for assessing data quality are defined below.

Spatial Data Quality Element*:* Data quality as defined in the 1.2.1 section, has many aspects. Quality element is the distinct aspect for determining the quality of geographic data. There are several categories of spatial data quality elements. ISO 19157 standard is the accepted document in geospatial domain for concepts of quality. The completeness of a data set can be given as example (ISO, 2013).

Spatial Data Quality Scope: The scope of spatial data quality is the extent of data is to be tested. Any common properties of spatial data or relations and temporal or spatial data may be defined for spatial data quality scope (ISO, 2013).

Spatial Data Quality Report: Quality assessment has a result and the result should be documented. The report may have, quantitative, conformance and descriptive results (ISO, 2013).

1.2.1.1. Spatial Data Quality Elements in Literature

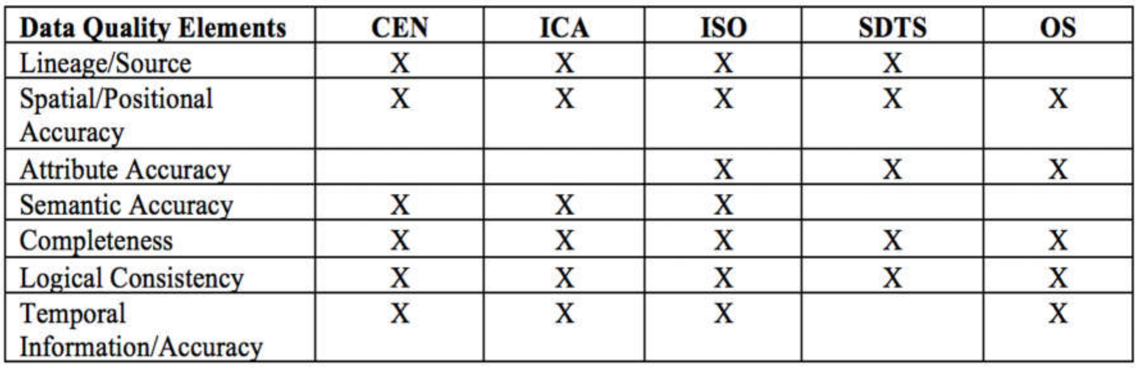
The methods for production and storage of the maps are now different. With technological development, maps have become digitized. This affects the production means, methods, storage and reusability etc. of spatial data. There are many types of spatial data classified according to their production methods such as; SDI, crowdsourcing geographic data (CGI), VGI, LOD and open data are the examples. However, for each type of spatial data have similar quality elements with same meaning sometimes different meaning, there are also common quality elements or totally new quality elements. As the result of these divergencies, academia and organizations such as ISO and Open Geospatial Consortium (OGC) are seeking constantly to standardize the concepts for quality domain. Common quality elements from these studies are listed below.

1.2.1.2. Spatial Data Quality Elements for SDI

National Committee on Digital Cartographic Data Standards (NCDCDS) define five data quality elements (NCDCDS, 1987). In 1995, NCDCDS updated the standards with two more quality elements; temporal accuracy and semantic consistency.

European Committee for Standardization Technical Committee 287 publish standards, in cooperation with the ISO Technical Committee 211 (CEN, 2013; Gouveia et al., 2001), including the data quality standards ISO 19113 from 2003 and ISO 19157 from 2013. Spatial data quality committee of ICA (1995), Spatial Data Transfer Standard from Federal Geographic Data Committee and Ordnance Survey (OS) Master Map data guide (2004) define data quality elements reside in the specifications (Veregin, 1999). Quality elements from these standards are summarized in the Table 1. Definition of the quality elements are listed below.

Table 1. Data quality elements defined by standards or cartographic organizations, modified from Devillers et al. (2005).



Data Quality Elements	CEN	ICA	ISO	SDTS	OS
Lineage/Source	X	X	X	X	
Spatial/Positional Accuracy	X	X	X	X	X
Attribute Accuracy			X	X	X
Semantic Accuracy	X	X	X		
Completeness	X	X	X	X	X
Logical Consistency	X	X	X	X	X
Temporal Information/Accuracy	X	X	X		X

Lineage: Lineage gives information about history of data including, production date, updates, transformations of data until the final product, descriptions of the data source and methods for derivation (Clarke and Clark, 1995).

Spatial/Positional Accuracy: Conformance of the coordinate values in the dataset to the ones in the reference dataset. Positional accuracy is included in the final product after all transformations (J. Drummond, 1995).

Attribute Accuracy: There are two types of attributes; quantitative and qualitative. Examples of qualitative attributes are feature names, types of buildings or roads; quantitative attributes include numerical values as a result of any measurement or analysis, population of a city, width of a road, number of cities in a country. Attribute accuracy refers to accuracy of quantitative attribute and correctness of qualitative attribute (Servigne et al., 2006).

Semantic Accuracy: This criterion is related with some other quality criteria such as completeness, attribute accuracy. It is the accuracy of the perceived reality and modelled reality. Hence, semantic accuracy can be defined as the modelling accuracy of specifications. (Salgé, 1995).

Completeness: This quality element can be applied to the model, the data and objects and attributes. Data completeness is defined as errors of omission (absence of the data) or commission (presence of data more than expected) of certain objects. Model completeness is defined as the suitability of the provided representation for users' requirements (Brassel et al., 1995). Types of completeness are shown in Figure 2.

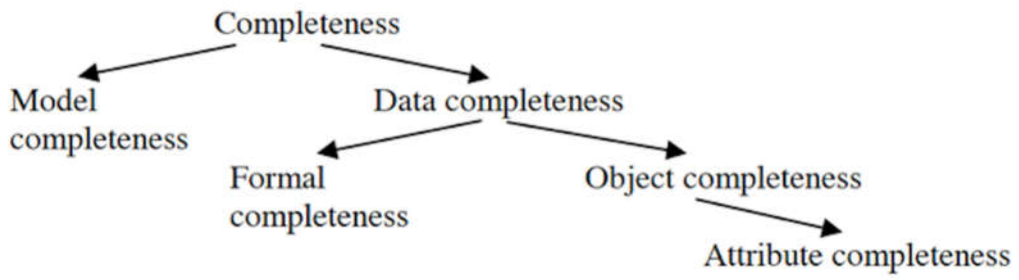


Figure 2. Types of completeness (Brassel et al., 1995).

Logical Consistency: It is described as the conformance of relationships encoded in the structure of database with respect to all the constraints caused by data specifications. It is about the consistency of modelled dataset with the characteristics of intended model with integrity constraints (Servigne et al., 2006).

Temporal Information/Accuracy: This quality element is responsible with providing information about how accurate and consistent the temporal aspect of the data with data observation dates (origin), update periods and time and the data's validity period.

The categorization in Table 1 follows the ISO scheme. There are some differences between the descriptions of data quality elements defined by ISO standards and other literature. For example, completeness is described as only the absence or presence of the data more or less than expected. One more example is the semantic accuracy element. This is not defined in the ISO standard as a distinct quality element. Instead, logical consistency is described. It has similar meaning with slight differences. Institutions usually define the quality elements to be inspected by regulations. For INSPIRE, data specifications for themes such as Transport network, Hydrography, Cadastral Parcels are created (INSPIRE, 2014a, 2014b, 2014c). INSPIRE Thematic Working Group creates the Specifications for various themes and INSPIRE Data Quality Working Group discussed for generating a road map for data quality to be followed. INSPIRE data specifications has a part for Data Quality and they follow (ISO, 2013) standard for quality assessment purposes. Tóth et al. (2013) defines the data quality assessment procedure and required data quality elements for INSPIRE data theme.

1.2.1.3. ISO 19157

ISO 19157 standard defines concepts and includes directives for spatial data quality, quality elements, evaluation methods and reporting the quality result (ISO, 2013). Currently it is followed by many institutions and studies about data quality. Hence, it is described in this part of thesis to describe the accepted definitions.

Logical consistency, completeness, thematic accuracy, temporal accuracy, lineage are the main quality elements defined in the ISO 19157 standard (ISO, 2013). Definitions of the data quality elements represented in the ISO standard is described below.

Completeness is defined as the existence of correct amount of data including attributes, features and relationships (ISO, 2013). It has two more quality elements; commission (excess of data) and omission (missing of data). If there are ten houses instead of twelve which exist in universe of discourse is the example of commission of data.

Logical consistency is the conformance level to the logical rules which are used to construct a dataset. It has four quality elements;

Conceptual consistency is defined as the conformance of a dataset to its conceptual schema (ISO, 2013). If there is inconsistency between the rules defined for the conceptual schema and the dataset then there is conceptual inconsistency. Invalid overlap of road and building can be given as example.

Domain consistency is the conformance of the attribute values of the dataset to the defined values in the related domain (ISO, 2013). If the speed limit of a road is 80 instead of 50 which is defined for the domain, there is domain inconsistency.

Format consistency is the conformance of the format that is defined for the physical structure of the dataset (ISO, 2013). Storing spatial data in wrong format, GML format instead of shapefile is a sample inconsistency.

Topological consistency is defined as the correctness of the topological rules declared for the dataset (ISO, 2013). The self- intersection of a polygon type feature can be given as example for the topological inconsistency.

Thematic accuracy is defined as the correctness of the quantitative and non-quantitative attributes, and classification of features in the dataset. (ISO, 2013). It has three sub elements. Classification correctness is defined as how accurate the classification of the features and attributes is compared to a reference data set. Non-quantitative attribute correction is defined to measure if a non-quantitative attribute is given accurately or not. Quantitative attribute correction is defined to determine how close the attribute value of data to the reference value of the attribute.

Temporal quality is defined as the correctness of the temporal attributes and temporal relations of the features (ISO, 2013). It has three quality elements. Accuracy of a time measurement, the closeness of the temporal attribute values to the reference attributes. Temporal consistency is the correctness of the ordered events. If there are four events but not ordered correctly can be given as example. Temporal validity is the validity of the temporal attribute.

Positional Accuracy is defined as how accurate the position values of a dataset are based on a spatial reference system (ISO, 2013). Absolute or external accuracy is defined as the closeness of the positional values according to a reference dataset. Relative or internal accuracy is defined as the closeness of the positional values according to an internal reference in a dataset.

Finally, usability refers to how a given dataset can meet specific user requirements that cannot be described using the quality elements described above (ISO, 2013).

1.2.1.4. Logical Consistency

Kainz (1995) define logical consistency as "the logical rules of structure and attributes rules for spatial data and describes the compatibility of a datum with other data in a data set". These rules also apply to the relationships and composition of relationships between objects.

Servigne et al. (2000) consider logical consistency and semantic accuracy together. The topo-semantic consistency concerns the correctness of the topological relationship between two objects according to their semantics. For instance, a building inside another building is certainly an error whereas a building inside a parcel is not an error.

1.2.1.5. VGI and Linked Data

VGI is a new source of data which is contributed by citizens. VGI data may be in different forms such as; photographs with geotags on websites e.g. Panoramio and Flickr, online maps on websites such as OpenStreetMap (OSM) and Wikimapia. In addition, there are 3D VGI such as OSM-3D and OSM2World (Fonte et al., 2017). OpenStreetMap is the first VGI project that allows volunteers to collect road data. The OSM data model is constituted of three data types or objects: nodes, ways (polygons and polylines) and relations (logical collections of ways and nodes). There are many tools for citizens to contribute spatial data. The main tool is the websites. Some of them are Wikimapia, OSM, LinkedGeoData, Google Map Maker and GeoNames (See et al., 2017). Wikimapia provides means to describe places in the world and can be accepted as a gazetteer service (Michael F. Goodchild, 2007). It gives the possibility of uploading and categorizing photos, adding descriptions, marking places to the volunteers. OSM website allows to contribute data to the project. LinkedGeoData is the semantic form of OSM data. Google Map Maker allows to map features such as point of interest (POIs) and roads. GeoNames is another gazetteer service which people can add or edit place names to the website of GeoNames (See et al., 2017).

Data acquisition of VGI is different from the SDI. Data for SDI is produced by institutions and conventional methods. While the volunteered GIS arise, quality is the main issue. There are many studies to propose quality management methods, quality elements, quality assessment. Because of the divergence in the data types, some quality dimensions and assessment methods of VGI are different from the ones for SDIs.

Ballatore et al. (2013) define the methods and quality components of crowdsourced spatial data quality. It classifies the quality evaluation methods into four types. The evaluation can be manual, within-knowledge-base, between-knowledge-base and application-based.

1.2.2. Semantic Web

Today, most of the information on the web is coded in such a way that humans can interpret it. The size of the information on the web, and the rapid increase in the need for information make it necessary that the information can be interpreted by not only people, but also computers. Realizing the need for this, Berners-Lee and Hendler (2001) introduced the "Semantic Web". Tim Berners-Lee defines "Semantic Web" as the "Web of Data". The idea underlying the "Semantic Web" is embellishing the web-accessible data with a semantic meaning through ontologies. This will help the content of the web document to be read and understood by the computers without much human intervention.

The Semantic Web is a web-based development of the semantic content of web pages, software agents who navigate between pages can easily to perform complex tasks (Berners-Lee and Hendler, 2001). According to Guha et al. (2003), Semantic Web makes discovery and reuse of information and various media and automation more effective. Semantic Web data involves not only web pages or images, but also people, places and organizations (Guha et al., 2003). The initial Semantic Web architecture is shown in Figure 3 (Begam and Ganapathy, 2011; Berners-Lee, 2000).

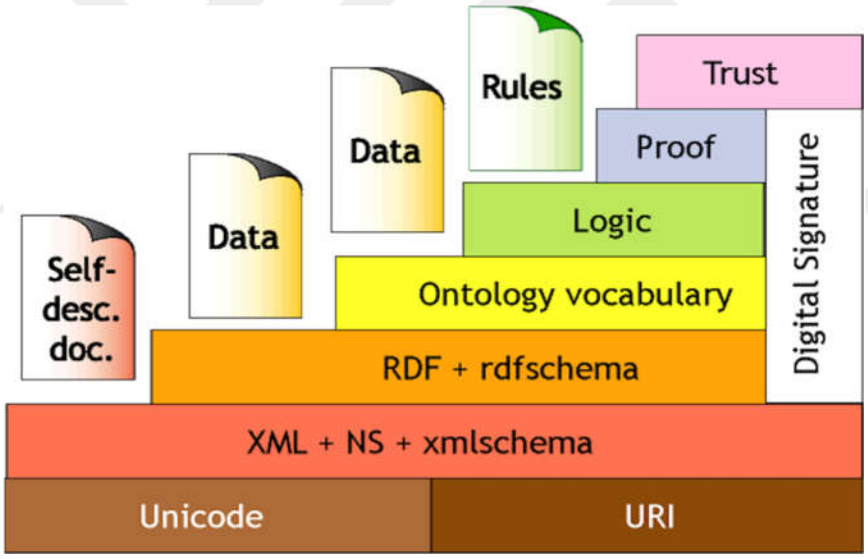


Figure 3. Semantic Web layered architecture (Berners-Lee, 2000).

The newer version of the Semantic Web layer cake is shown in Figure 4 (Berners-Lee, 2006). As seen in the figures, ontologies are at the heart of Semantic Web. In the newer version, there are W3C standards for queries and rules, SPARQL and Rule Interchange Format (RIF), respectively.

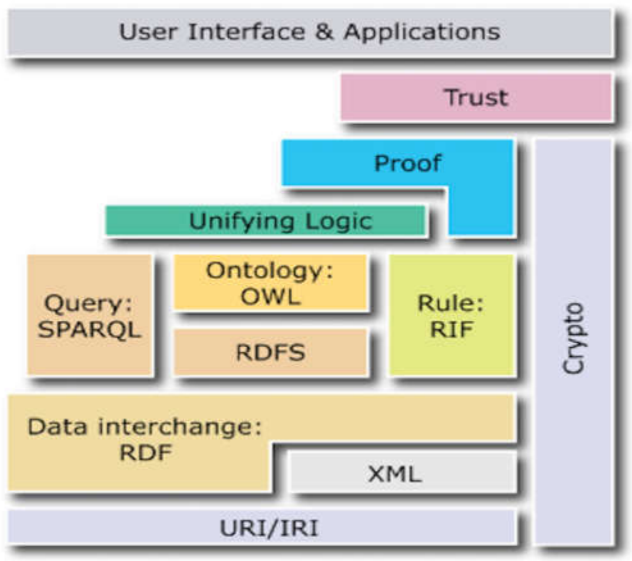


Figure 4. Semantic Web layer cake (Berners-Lee, 2006).

Ontology is a common word that can be used as a standard for modelling domain. Real world objects or concepts can be expressed along with their properties and relationships. With ontologies, semantic relations are established between concepts; this way, the concepts are expressed in a formalized manner. Ontologies describe hierarchies using the subclass relations. Man and Woman can be ontological classes that are subclasses of a class named Person.

1.2.2.1. The Resource Description Framework

The Resource Description Framework (RDF) is the primary representation framework for developing Semantic Web and uses graph structure to represent data which provides linking between web resources (W3C, 2004c). RDF uses triples, describing an abstract version of (directed) graphs (Antoniou and Harmelen, 2008; W3C, 2014c).

A typical rectangle, polygon or similar shapes have vertices (nodes) and edges. Some of the vertices are connected by lines or curves that are called edges. These edges can be of different length or colour. They can also be given directions, creating directed graphs.

In RDF, the geometric graphs are generalized. Vertices are now arbitrary resources (still can be called nodes), and the colours describe relations between them. With a subject-predicate-object triple, there is an "edge" from the subject to the object, "coloured" according to the predicate. Statements such as “ex:cemre ex:writes ex:thisThesis” are represented as RDF subject-predicate-object triples in Figure 5. To represent each node and edge, RDF uses Uniform Resource Identifiers (URIs), which are used with prefixes in various RDF serialization formats (W3C, 2004a). URIs are sequences of characters identifying resources (Berners-Lee et al., 2005).

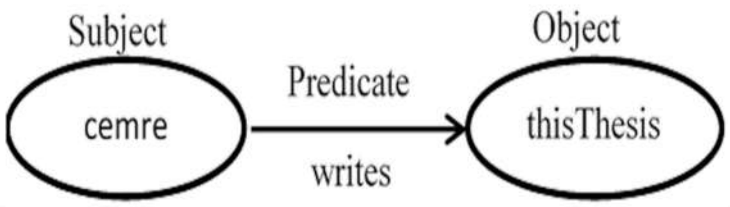


Figure 5. An RDF triple.

A knowledge base about the author is given in Figure 6. A sample knowledge base about Cemre YILMAZ is represented as a graph in that figure.

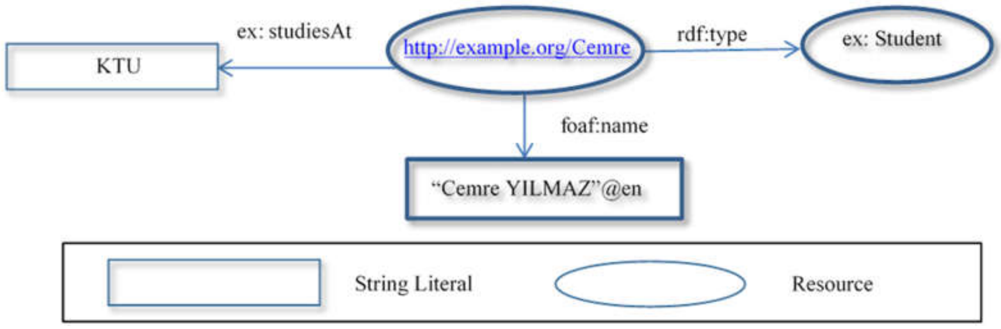


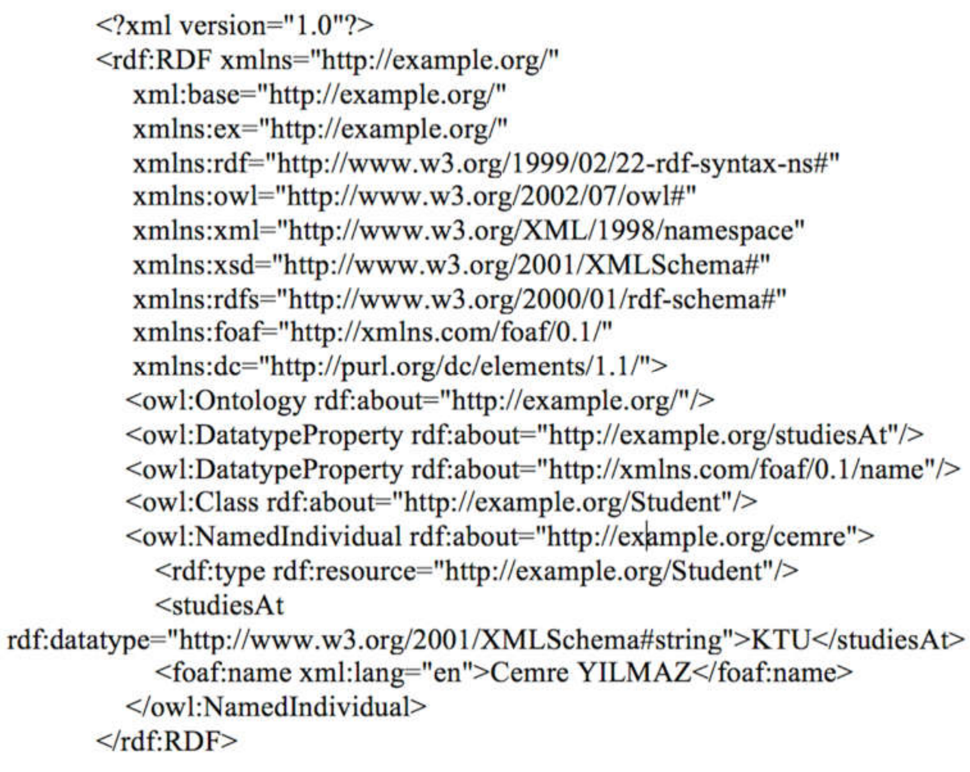
Figure 6. RDF graph

RDF has several serializations, called RDF syntaxes. Using these serializations, RDF graphs can be published. RDF/XML and RDFa formats are W3C standards (W3C, 2004b, 2015). In this thesis, the Turtle format is preferred. It is a very commonly used format and a W3C recommendation since 2014 (W3C, 2014b). Other formats such as N3 and N Triples exist (W3C, 2011a, 2014a). N3 format is, in terms of expressivity, equivalent to RDF/XML, but easier to “scribble” (URL-2, 2005). Every valid Turtle document is a valid N3 document.

N3 has additional capability. Furthermore, every valid N-Triples document is a valid Turtle document; N-Triples does not use prefixes (W3C, 2014a).

RDF uses URIs, which are not necessarily Uniform Resource Locators (URLs), even if they contain strings such as “http://”. Typically, the URIs do not exist on the web as a website. RDFa format is for Hypertext Markup Language (HTML) documents (W3C, 2008a). RDFa enables adding structured data to HTML documents, and therefore Internet websites. It is also used to publish RDF URIs as URLs describing the related RDF documents on the website given by the URL.

RDF/XML syntax uses the Extensible Markup Language (XML) for representing RDF. XML is used widely, for a wide range of purposes. Even the docx MS Word document format is zipped container of XML files. The RDF/XML syntax is standardized by the W3C (2004b). RDF/XML file for the knowledge base in the last figure is given in Figure 7.



```
<?xml version="1.0"?>
<rdf:RDF xmlns="http://example.org/"
    xml:base="http://example.org/"
    xmlns:ex="http://example.org/"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xml="http://www.w3.org/XML/1998/namespace"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    xmlns:dc="http://purl.org/dc/elements/1.1/">
   <owl:Ontology rdf:about="http://example.org/"/>
   <owl:DatatypeProperty rdf:about="http://example.org/studiesAt"/>
   <owl:DatatypeProperty rdf:about="http://xmlns.com/foaf/0.1/name"/>
   <owl:Class rdf:about="http://example.org/Student"/>
   <owl:NamedIndividual rdf:about="http://example.org/cemre">
      <rdf:type rdf:resource="http://example.org/Student"/>
      <studiesAt
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">KTU</studiesAt>
      <foaf:name xml:lang="en">Cemre YILMAZ</foaf:name>
   </owl:NamedIndividual>
</rdf:RDF>
```

Figure 7. Sample RDF/XML document.

URI base strings for RDF are used with prefixes. The canonical prefixes for RDF are “rdf:”, “xml:”, “xsd:” and “rdfs”. OWL ontology documents use prefix “owl:”, as well.

These prefixes and the URIs they represent are shown in Table 2. For instance, “owl:Ontology” is short for "http://www.w3.org/2002/07/owl#Ontology". Each RDF document may introduce some other prefixes for URI base strings, such as “sdqo:”. The Prefix.cc website[1] lists many URI prefixes.

Table 2. URIs and prefixes.

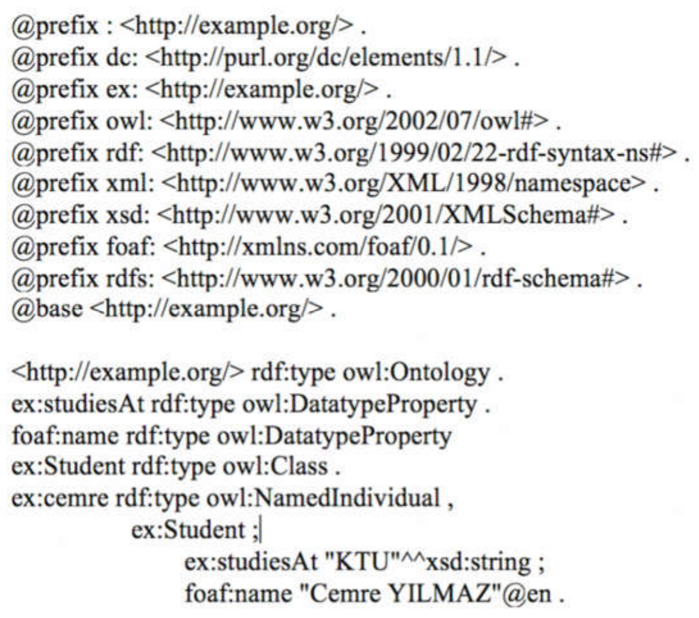
Prefix	URI	Note
rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#	RDF documents
xml:	http://www.w3.org/XML/1998/namespace	RDF documents
xsd:	http://www.w3.org/2001/XMLSchema#	RDF documents
rdfs:	http://www.w3.org/2000/01/rdf-schema#	RDF documents
owl:	http://www.w3.org/2002/07/owl#	OWL documents
ogc: or geo:	http://www.opengis.net/ont/geosparql#	OGC GeoSPARQL prefix

As opposed to XML, Turtle format is specially designed for RDF documents (W3C, 2014b). Turtle stands for “Terse RDF Triple Language”. Similar to the Structured Query Language (SQL), triples with the same subject can be combined, even for blank nodes. If the subject is a blank node, square brackets are used. [predicate object] is a triple where the subject is a blank node.

Turtle is well suited for prefix mapping. In general, Turtle documents are terser than RDF/XML documents. Contents of Turtle file equivalent of the RDF/XML file shown in the previous figure is shown in Figure 8.

The ontology editor Protégé uses the Manchester syntax for presentation and editing (Horridge et al., 2006).

[1] http://prefix.cc



```
@prefix : <http://example.org/> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix ex: <http://example.org/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@base <http://example.org/> .

<http://example.org/> rdf:type owl:Ontology .
ex:studiesAt rdf:type owl:DatatypeProperty .
foaf:name rdf:type owl:DatatypeProperty
ex:Student rdf:type owl:Class .
ex:cemre rdf:type owl:NamedIndividual ,
            ex:Student ;|
                ex:studiesAt "KTU"^^xsd:string ;
                foaf:name "Cemre YILMAZ"@en .
```

Figure 8. Sample Turtle document.

1.2.2.2. RDF Vocabulary Description Language

The standard URI does not allow the whole range of Unicode characters. To overcome this, Internationalized Resource Identifiers (IRIs) are introduced. IRIs are uniform resource identifiers that, as strings, can contain Unicode characters (Dürst and Suignard, 2004), generalizing URIs. In RDF. anything that is not a blank node refers to a resource. Resources in RDF are identified either by (IRI) or literals.

Literals are strings that can be language-tagged and that contain datatype IRIs, such as xsd:integer and xsd:string. To describe the integer 1, the literal "1"^^xsd:integer is used. To describe the string “KTU”, the literal "KTU"^^xsd:string is used. For strings, language tags such as @en can be added.

RDF needs a method to define classes and attributes for specific domains besides defining classes, properties and values with resources. This is developed by extending RDF. RDFS is the W3C standard for semantic schema definition language for RDF (W3C, 2014c). RDFS introduces some vocabulary, some classes and the class hierarchy (W3C, 2014c). Any RDF document automatically has these RDF and RDFS constructs and relevant entailments. In RDF, an entailment is a triple that follows from previous triples. rdf:type is an instance of rdf:Property that describes “is-a” relations. It also has entailments. Subject1 being an instance of object1, equivalently subject1 being of type object1 is represented by the triple “subject1 rdf:type object1”. Subject1 being of type object1 should imply that object1 is a type, a class, and this is automatically achieved with the entailments. The triple “subject1 rdf:type object1” has the entailment “object1 rdf:type rdfs:Class”.

The RDF and RDFS classes are shown in Table 3 (W3C, 2014c). rdfs:Class denotes the class of all classes and rdfs:Resource denotes the class of all resources, so IRIs and literals.

With the property rdfs:subClassOf connecting classes, the class hierarchy is introduced. Any class, including rdfs:Class and rdfs:Resource, is an instance of rdfs:Class and a subclass of rdfs:Resource. rdfs:Literal, the class of all literals, is a superclass to rdf:langString, rdf:HTML, rdf:XMLLiteral. Each instance in rdfs:Datatype is a subclass of rdfs:Literal. The class rdfs:Container is a superclass to rdf:Seq, rdf:Bag and rdf:Alt.

The RDF Schema properties are listed in Table 4 (W3C, 2014c).

Table 3. RDF and RDFS classes (W3C, 2014c)

Class name	**Comment**
rdfs:Resource	The class of resources, everything except blank nodes.
rdfs:Literal	The class of literal values.
rdf:langString	The class of language-tagged string literal values.
rdf:HTML	The class of HTML literal values.
rdf:XMLLiteral	The class of XML literal values.
rdfs:Class	The class of classes.
rdf:Property	The class of RDF properties.
rdfs:Datatype	The class of RDF datatypes.
rdf:Statement	The class of RDF statements (subject-predicate-object).
rdf:Bag	The class of unordered containers.
rdf:Seq	The class of ordered containers.
rdf:Alt	The class of containers of alternatives.
rdfs:Container	The class of RDF containers.
rdfs:ContainerMembershipProperty	The class of container membership properties, which are sub-properties of 'member'.
rdf:List	The class of RDF Lists.

Table 4. RDF common properties with their domains and ranges.

Property name	Comment	Domain	Range
rdf:type	The subject is an instance of a class.	rdfs:Resource	rdfs:Class
rdfs:subClassOf	The subject is a subclass of a class.	rdfs:Class	rdfs:Class
rdfs:subPropertyOf	The subject is a subproperty of a property.	rdf:Property	rdf:Property
rdfs:domain	A domain of the subject property.	rdf:Property	rdfs:Class
rdfs:range	A range of the subject property.	rdf:Property	rdfs:Class
rdfs:label	A human-readable name for the subject.	rdfs:Resource	rdfs:Literal
rdfs:comment	A description of the subject resource.	rdfs:Resource	rdfs:Literal
rdfs:member	A member of the subject resource.	rdfs:Resource	rdfs:Resource
rdf:first	The first item in the subject RDF list.	rdf:List	rdfs:Resource
rdf:rest	The rest of the subject RDF list after the first item.	rdf:List	rdf:List
rdfs:seeAlso	Further information about the subject resource.	rdfs:Resource	rdfs:Resource
rdfs:isDefinedBy	The definition of the subject resource.	rdfs:Resource	rdfs:Resource
rdf:value	Idiomatic property used for structured values.	rdfs:Resource	rdfs:Resource
rdf:subject	The subject of the subject RDF statement.	rdf:Statement	rdfs:Resource
rdf:predicate	The predicate of the subject RDF statement.	rdf:Statement	rdfs:Resource
rdf:object	The object of the subject RDF statement.	rdf:Statement	rdfs:Resource

1.2.2.3. Web Ontology Language

Web Ontology Language (OWL) is a family of Semantic Web languages for describing ontologies (URL-3, 2004). Primary OWL languages are OWL Full, OWL DL and OWL Lite.

OWL ontologies are just collections of axioms corresponding to explicit logical assertions about concepts and individuals in a domain. OWL ontologies are serialized in RDF formats such as Turtle, where axioms are some RDF triples with additional restrictions. A valid OWL document is a valid RDF document, but the reverse is not necessarily true. RDF documents in general do not contain OWL constructs such as owl:Ontology.

Each OWL ontology is itself a resource or blank node that is an instance of owl:Ontology. Named OWL ontologies have IRIs, anonymous ones don’t have IRIs (no literals). owl:imports is an instance of rdf:Property; it is a transitive property between two

OWL ontologies denoting the importing relation. When an OWL ontology imports another OWL ontology (therefore also any ontology imported by that ontology), it is as if the latter ontology is subsumed in the former ontology.

In an ontology, individuals, concepts (classes) and properties exist. owl:Class is a subclass of rdfs:Class denoting the ontology classes.

In any OWL document, two special OWL classes exist, owl:Thing and owl:Nothing. When a resource is declared to be an OWL class, several entailments follow. In particular, all OWL classes are subclasses of owl:Thing and superclasses to owl:Nothing. OWL named individuals are IRI resources that are instances of OWL classes and each OWL named individual is an instance of owl:Thing. The only subclass of owl:Nothing is itself, and owl:Nothing does not contain any individuals in a (consistent) ontology.

In OWL ontologies, Open World logic is preeminent and Unique Name Assumption (UNA) does not exist. Unless declared explicitly, two resources are not necessarily different. owl:differentFrom is used for declaring two resources as different. owl:AllDifferent is a shortcut for making all individuals in the given OWL class different from each other.

In OWL, a distinction exists between properties. owl:ObjectProperty and owl:DatatypeProperty are two subclasses of rdf:Property. Object properties are properties between IRI resources, and datatype properties are from an IRI resource to a literal resource. They are necessarily disjoint in OWL DL and sublanguages, but not so in OWL Full.

Using the OWL languages, properties, object and data types, class operations, domains and ranges of properties associations can be defined (W3C, 2014c).

OWL Full has exactly the same constructs with OWL DL. It has fewer restrictions and is closer to RDF than any other OWL language. OWL Full is especially useful, when one wants to use classes as instances. This is not allowed in OWL DL or its more restrictive sublanguages. A primary concern is decidability and reasoning efficiency. The OWL vocabulary is listed in Table 5 (URL-3, 2004).

All reasoning processes eventually end with OWL DL and its sublanguages. In fact, OWL DL, being associated with Description Logic (DL), is the most expressive OWL language that is still decidable (Horrocks, 2002). Reasoners such as Pellet are implemented for OWL DL and its sublanguages (Sirin et al., 2007).

The sublanguages of OWL DL are created by adding several decidability-friendly capabilities to the attributive language AL and by manipulating restrictions.

OWL Lite is one of the sublanguages of OWL DL. It is designed for cases requiring mainly classification hierarchy and simple constraints. For example, OWL Lite allows quantitative restrictions but allows only the values 0 and 1 for quantity. OWL Lite allows more efficient reasoning algorithms. The computational complexity is above polynomial time even for OWL Lite.

Under the constraints of computational completeness and logical decidability, OWL DL has the most expressive possible structure, being associated with Description Logic. It allows logical reasoners, as any reasoning will end eventually; all true or false statements can be proven to be as such.

Table 5. OWL vocabulary.

owl:AllDifferent	owl:DataRange	owl:imports
owl:allValuesFrom	owl:differentFrom	owl:incompatibleWith
owl:AnnotationProperty	owl:disjointWith	owl:intersectionOf
owl:backwardCompatibleWith	owl:distinctMembers	owl:InverseFunctionalProperty
owl:cardinality	owl:equivalentClass	owl:inverseOf
owl:Class	owl:equivalentProperty	owl:maxCardinality
owl:complementOf	owl:FunctionalProperty	owl:minCardinality
owl:DatatypeProperty	owl:hasValue	owl:Nothing
owl:ObjectProperty	owl:oneOf	owl:onProperty
owl:Ontology	owl:OntologyProperty	owl:priorVersion
owl:Restriction	owl:sameAs	owl:someValuesFrom
owl:SymmetricProperty	owl:Thing	owl:TransitiveProperty
owl:unionOf	owl:versionInfo	

OWL 2, a W3C recommendation since 2012, the current OWL version, is backwards compatible with OWL (1). It adds new functionality, such as property chains and qualified cardinality restrictions. OWL 2 also introduces three additional profiles. OWL 2 EL, OWL 2 QL and OWL 2 RL are the three profiles of OWL2 (W3C, 2012a). OWL 2 EL enables polynomial time algorithms for reasoning. For very large ontologies with many properties and classes, OWL 2 EL is preferable. OWL 2 QL is well suited for situations where a lot of queries are processed, possibly because of the largeness of the number of instances. OWL 2 RL enables scalable reasoning with rule-based reasoners (W3C, 2012a).

In the following lines an example of OWL ontology concepts is explained with Turtle syntax based on the Example ontology and ontology named A Geographic Query Language for RDF Data (GeoSPARQL).

Class of “Feature” is subclass of “SpatialObject” class axiom can be declared as in the following lines;

```
ogc:Feature  rdf:type  owl:Class ;   rdfs:label "Feature"@en ;
                               rdfs:subClassOf  ogc:SpatialObject .
```

Object Property “hasGeometry” relates “f0” individual of “Feature” class with the “geom0” individual of “Geometry” class ;

```
ex:f0 rdf:type ogc:Feature , owl:NamedIndividual ;
                              ogc:hasGeometry ex:geom0 .
```

Datatype Property “asWKT” can be declared as follows in an ontology,

```
ogc:asWKT rdf:type owl:DatatypeProperty ;
                           rdfs:label "asWKT"@en .
```

Assertion of the class of a geometric individual “geom1” with its coordinates values can be declared as follows,

```
ex:geom rdf:type ogc:Geometry,
            owl:NamedIndividual ;
            ogc:asWKT "Polygon (( 0 0, 1 0, 1 1, 0 1))"^^ogc:wktLiteral
```

Domain and range restriction for properties make it possible to state the type of object and subject resources of the property. Formally, (x rdfs:domain y) and (x rdfs:range y) triples state the object and subject resources of a property. (x rdfs:domain y) states that resources which have a property x belong to class y. (x rdfs:range y) states that the value for the property x belongs to the class y.

While the domain expression of a property makes it possible to classify subjects, the range expression type objects of the property in an RDF triple.

For example, hasGeometry property in GeoSPARQL ontology has its objects from the class Feature and its property value from the class Geometry. It is declared as,

```
ogc:hasGeometry rdfs:domain ogc:Feature
ogc:hasGeometry rdfs:range  ogc:Geometry
```

1.2.2.3.1. Cardinality Restrictions

A cardinality restriction is used to restrict the number of values can be asserted for the property of an instance. This restriction is used to specify the number of value assertions for a property or to restrict that there is no property value. There are three types of cardinality restrictions; owl:maxCardinality, owl:minCardinality and owl:Cardinality. However, OWL-Lite supports all three types of cardinalities, it is only allowed to have values 0 or 1.

owl:maxCardinality Restriction

This OWL built-in relates the values of restriction class for property of OWL to the XML Schema datatype nonNegativeInteger. owl:maxCardinality constraint can be used to restrict the number of values that can take at most N semantically different values for the property concerned. These values may belong to individuals or values and N is the value of the cardinality constraint.

owl:minCardinality Restriction constraint can be used to restrict the number of values that can take at most N semantically different values for the property concerned. These values may belong to individuals or values and N is the value of the cardinality constraint. If there is owl:minCardinality restriction for any property as one or more, this means that property must have a value.

A class of individuals that can take at most one GeographicID.

The following example describes a class of individuals that have at most two parents:

```
<owl:Restriction>
  <owl:onProperty rdf:resource="#hasGeographicID" />
  <owl:maxCardinality
              rdf:datatype="&xsd;nonNegativeInteger">2</owl:maxCardinality>
</owl:Restriction>
```

1.2.2.3.2. OWL Formal Semantics

OWL DL is designed according to the description logic SHOIN. Several varieties for description logics are designed with different expressivity and scalability, such as ALC, SHIQ, SHOIN and SROIQ (Hitzler et al., 2009). ALC (Attributive Language with Complement) is the basic description logic and contains classes, individuals and roles (properties) as basic building blocks that can be related with each other. Statements in ALC are divided into two groups, ABox statements and TBox statements. The TBox statements are about class inclusion axioms (subclass or equal class situations) and the ABox statements are about individuals and their relationship with classes and roles. Under OWA, with ALC, one cannot express that a class has exactly a given set of elements, so one cannot declare closed classes (nominals). The cardinality cannot be given upper bounds.

SHOIN is an extension to ALC that allows closed class expressions and cardinality restrictions. Furthermore, with SHOIN, equality and inequality between individuals, role hierarchies, inverse-ness between roles, transitivity, symmetry, functionality and inverse functionality of roles can be expressed. SHOIN(D) logic is based on SHOIN and allows the use of data values, elements of datatypes.

SHOIN can also be further extended staying within decidable description logic limits. SROIQ is an extension to SHOIN allowing complex role inclusion axioms (propagation of one property along another one) (Horrocks et al., 2006). Furthermore, roles can be declared reflexive, antisymmetric, irreflexive, universal. Roles can also be expressed to be disjoint from or negated form of other roles. Qualified cardinality restriction exist; the individuals can be restricted to be within a class and to follow a cardinality restriction (Baader et al., 2003; Horrocks et al., 2006).

Below the syntaxes and semantics of DL components are explained.

SHOIN = S + H + O + I + N = ALCR+HOIN

S : ALC with transitive roles R+

R+ : transitive role, e.g., trans(isPartOf)

H : Role inclusion axioms, R1 vR2, e.g., isComponentOf vs. Partof

O : Nominals (singleton class), (a), e.g., ∃hasChild.mary

I : Inverse role, R−, e.g., isPartOf = hasPart−

N : Number restrictions, (≥ n R) and (≤ n R), e.g., (≥ 3 hasChild) (has at least 3 children)

D: datatype

Q: qualified cardinality restrictions

F: role functionality. (clearly implied if N exists, restricted number=1)

OWL Lite: SHIF(D), OWL DL: SHOIN(D), OWL 2 DL : SROIQ (D) . For SROIQ (D), one also uses RBox for roles (Krötzsch et al., 2012).

1.2.2.4. SPARQL Protocol and RDF Query Language

SPARQL Protocol and RDF Query Language (SPARQL) (W3C, 2008b), is a query language for RDF. SPARQL queries form triple patterns similar to RDF triples. It is like SQL, the Structured Query Language for relational databases, with similarly named terms such as SELECT, FROM, WHERE, ORDER BY, DISTINCT, FILTER, *, OPTIONAL, CONSTRUCT, ASK and DESCRIBE, but designed for RDF. Software such as D2RQ, can in fact be used to SPARQL query data, by first converting to RDF format. Within SPARQL, resources are represented with URIs that can use prefixes. Therefore, SPARQL queries often start with prefix declarations.

Simple example:

```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE {
   ?person foaf:name ?name .
}
```

This basic query lists all the people (subjects in the RDF source in triples where the predicate is foaf:name) alongside their names (objects in the RDF source in triples where the predicate is foaf:name).

ARQ, OpenLink Virtuoso, Redland's SPARQL Rasqal, SNORQL[2] are some of the SPARQL query editors.

1.2.2.4.1. SPIN and SHACL

SPIN is a 2003 W3C submission for modelling SPARQL queries also enabling functions for SPARQL in a web-friendly syntax, fit for further re-use (Knublauch et al., 2009; URL-12, 2017). The SPARQL queries can be defined and used using RDF triples with RDF (Fürber and Hepp, 2010). The select count query "SELECT COUNT (?object)

[2] https://github.com/kurtjx/SNORQL

WHERE ?this ?arg1 ?object" is represented as a node called ex:count as follows, which can later be reused (W3C, 2011c).

```
"ex:count  a       sp:Select ;
    sp:resultVariables ([ a       sp:Count ;
                    sp:expression sp:_object
                  ]) ;
    sp:where ([ sp:object sp:_object ;
              sp:predicate spin:_arg1 ;
              sp:subject spin:_this
            ])."
```

Another SPIN element ex:cardinality can be defined as a function, using ex:count as follows (W3C, 2011b).

```
"ex:cardinality a spin:Function;
rdfs:subClassOf spin:Functions;
spin:constraint [ a spl:Argument; spl:predicate sp:arg1];
spin:body ex:count."
```

This ex:cardinality can also be used in further queries (W3C, 2011b). Also, one can make use of paths, for instance for hierarchy of subclasses.

Six years after SPIN, developers of SPIN introduced Shapes Constraint Language (SHACL) (W3C, 2017). SHACL is created for validating RDF data against the rules that they should comply with. SHACL uses shape graphs to validate RDF data. It works as a successor to SPIN. SHACL became a W3C recommendation following meetings with the developers of alternatives to SPIN (W3C, 2017). Among the features SHACL allows that is not possible with direct OWL is IRI pattern matching (Tomaszuk, 2017).

A SHACL shape graph is seen as the description of the data graphs that do satisfy the conditions. Validating is done with Closed World reasoning. As in SPIN, queries can be saved or used made into functions which can be used in the later queries. With SHACL, JavaScript will be able to be implemented. Another improvement of SHACL over SPIN is with the usage of parameters such as $this, which have been given special status. SHACL is well-adapted for the anonym parts declared between brackets. Constraint definition example with SHACL to validate data to find out the buildings which have less than two floors.

```
ex:Building a rdfs:Class,
sh:NodeShape ;
rdfs:label «Building»;
rdfs:subClassOf ex:Feature ;
  sh:sparql
 [ sh:message "Must have at least 2 numberOfFlats,
but has number of flats «{?flats
}» ;
  sh:prefixes ex:hasNumberOfFlats ;
          sh:select """ SELECT  $this ?flats
WHERE
{ $this ex:hasNumberOfFlats ?flats .

          FILTER (?flats< 2) . }""" ]
.
```

1.2.2.4.2. GeoSPARQL

GeoSPARQL is a standard developed by OGC to provide a standard way to define and query geospatial elements in RDF (OGC, 2012). The GeoSPARQL standard follows a modular design and consists of one core component for top level classes and five additional components (OGC, 2012). The other components are topological vocabulary component, geometry component, geometry topology, query rewrite and RDFS entailment components as shown in Figure 9 (OGC, 2012).

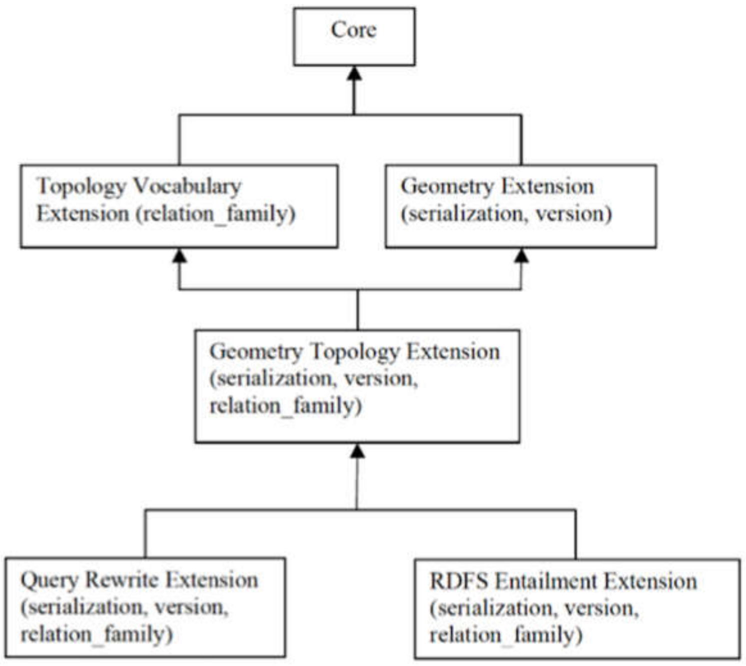

Core
Topology Vocabulary Extension (relation_family)
Geometry Extension (serialization, version)
Geometry Topology Extension (serialization, version, relation_family)
Query Rewrite Extension (serialization, version, relation_family)
RDFS Entailment Extension (serialization, version, relation_family)

Figure 9. GeoSPARQL components (OGC, 2012a).

Some of the existing triple stores that implement GeoSPARQL are Parliament[3], USeekM[4] , Strabon[5] , Virtuoso[6] and GraphDB[7].

1.2.2.5. Semantic Web Rule Language

Semantic Web Rule Language (SWRL) is the combination of OWL DL, OWL Lite and Unary / Binary Datalog RuleML (W3C, 2004d). SWRL syntax is expanded with OWL for more advanced inference capabilities. SWRL atom types are shown in Table 6.

[3] http://parliament.semwebcentral.org/
[4] https://www.openhub.net/p/useekm
[5] http://test.strabon.di.uoa.gr/euhydro/
[6] https://virtuoso.openlinksw.com/
[7] http://graphdb.ontotext.com/

Table 6. SWRL atom types.

SWRL Atom Type	**Example Atom**
Class Atom	Feature(?x), Geometry(?x)
Individual property atom	hasGeometry(?x,?y)
sameAs atom / differentFrom atom	sameAs(?x,?y)/ differentFrom(?x,?y)
Data Valued property atom	hasId(?x, “5”)
Built-in Atom	swrlb:lessThan (?x, 7)
Data range atom	xsd:double(?X)

SWRL rules are “If-Then” rules for OWL. It means if some statements are true and the resulting statement should also hold true. SWRL rule axiom consists of an antecedent (body) and a consequent (head). Each of antecedent and consequent consist of set of atoms (W3C, 2004d). If all statements in the antecedent clause are proven to be true, then all statements in the consequent clause are also proven as true. With the rules and existing statements, new statements are inferred involving individuals in the ontology.

The OWL 2 language is not able to express all relations; it is not possible to define relations between individuals which with individuals already have relations in OWL 2. Widely used uncle example can be given as example. If Cemre has Gülcen as parent and Gülcen has Yusuf as brother, then Cemre has Yusuf as uncle. This rule is given below in First Order Logic (FOL) syntax.

hasParent(?x1,?x2) ∧ hasBrother(?x2,?x3) ⇒ hasUncle(?x1,?x3)

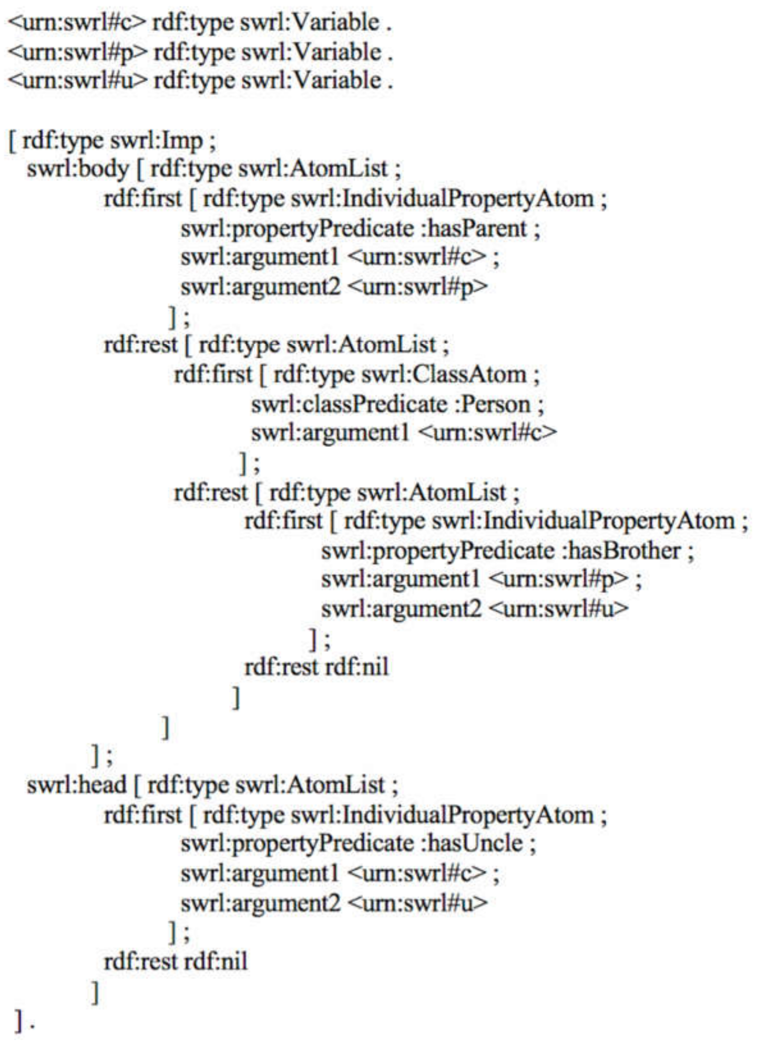
Properties and classes can be used together in a SWRL rule, head of the rule may compose of properties and classes together. A SWRL rule may consists of unary, binary and n-ary predicates; respectively for classes and data types, properties, and some special built-ins. Default SWRL built-ins are separated into seven categories. There are built-ins for comparisons, mathematical or string operations, date-time-duration manipulation and usage, Boolean operator, URIs and lists.[8] The default built-ins for lists are applicable for OWL Full.

[8] https://www.w3.org/Submission/SWRL/#8

Uncle SWRL rule:

Person(?c) ^ hasParent(?c,?p) ∧ hasBrother(?p,?u) ⇒ hasUncle(?c,?u)

Figure 10 shows how the SWRL rule for the uncle relation, as represented in Turtle syntax. As it can be seen in the figure, RDF lists are used successively to represent conjunctions.



```
<urn:swrl#c> rdf:type swrl:Variable .
<urn:swrl#p> rdf:type swrl:Variable .
<urn:swrl#u> rdf:type swrl:Variable .

[ rdf:type swrl:Imp ;
  swrl:body [ rdf:type swrl:AtomList ;
          rdf:first [ rdf:type swrl:IndividualPropertyAtom ;
                  swrl:propertyPredicate :hasParent ;
                  swrl:argument1 <urn:swrl#c> ;
                  swrl:argument2 <urn:swrl#p>
                ] ;
          rdf:rest [ rdf:type swrl:AtomList ;
                  rdf:first [ rdf:type swrl:ClassAtom ;
                          swrl:classPredicate :Person ;
                          swrl:argument1 <urn:swrl#c>
                        ] ;
                  rdf:rest [ rdf:type swrl:AtomList ;
                          rdf:first [ rdf:type swrl:IndividualPropertyAtom ;
                                  swrl:propertyPredicate :hasBrother ;
                                  swrl:argument1 <urn:swrl#p> ;
                                  swrl:argument2 <urn:swrl#u>
                                ] ;
                          rdf:rest rdf:nil
                        ]
                ]
        ] ;
  swrl:head [ rdf:type swrl:AtomList ;
          rdf:first [ rdf:type swrl:IndividualPropertyAtom ;
                  swrl:propertyPredicate :hasUncle ;
                  swrl:argument1 <urn:swrl#c> ;
                  swrl:argument2 <urn:swrl#u>
                ] ;
          rdf:rest rdf:nil
        ]
 ] .
```

Figure 10. SWRL Uncle Rule in Turtle syntax.

SWRL rules are in general not DL-safe because they can make deductions based on OWL individuals that are not explicitly known. As a result, this may cause undecidability. Hence, for DL safety, SWRL rules should only applied to named individuals, assertion part of data. This is possible with the way reasoner deals with SWRL rules. Pellet supports DL-

Safe SWRL Rules. Following knowledge base can be given as an example to illustrate the how DL-Safe rule affect the inferred result (Skillen et al., 2013).

Axioms

Parent hasChild some Person
Parent (a), Parent(b), Parent(c), Person(d)
hasChild(a,d)
Rule
hasChild(?x,?y) -> personWithChild(?x)

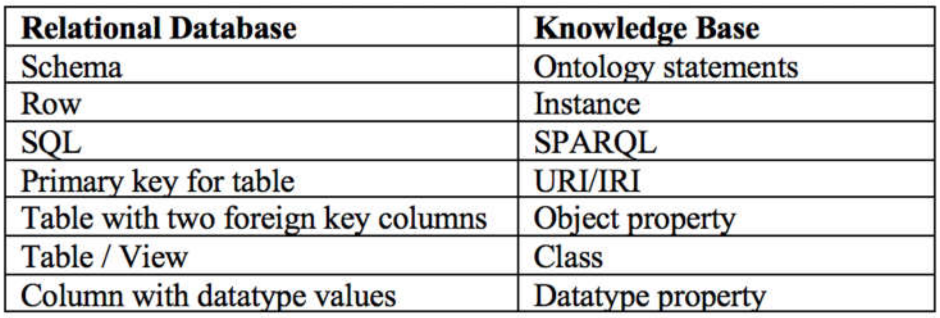
Consequence with DL-Safety
personWithChild(a)
Possible consequence without DL-Safety
personWithChild(a), personWithChild(b), personWithChild(c).

Semantic Query-Enhanced Web Rule Language (SQWRL) is built on SWRL for querying capability (O'Connor and Das, 2009).

1.2.2.6. Ontologies and Relational Databases

In various domains such as Data Integration and Semantic Web, ontologies are widely used for representing the shared conceptualization of domain interest (URL-4, 1998). These applications mostly require using data residing within databases. Accessing data with ontologies is be done with ontology-based data access (OBDA) systems. Main aim of OBDA is to access data without the necessity of storing data in the ontology (Poggi et al., 2008). This need by the applications in this domain as with different approaches as described in Calvanese et al. (2009). A DL-Lite approach is implemented for the OBDA system QUONTO (Acciarri et al., 2005). There are some other applications developed for ontology based data access such as D2RQ Protégé plug-in and Ontop Framework (Bagosi et al., 2014). A comparison between relational databases and ontological knowledge bases is shown in Table 7 (Hebeler et al., 2011; Sicilia and Lytras, 2008).

Table 7. Comparison between relational databases and knowledge bases.



Relational Database	Knowledge Base
Schema	Ontology statements
Row	Instance
SQL	SPARQL
Primary key for table	URI/IRI
Table with two foreign key columns	Object property
Table / View	Class
Column with datatype values	Datatype property

1.2.3. Tools for Semantic Web and OWL

1.2.3.1. Pellet and Openllet

Pellet is a Java-based reasoner for OWL 2 DL based ontologies supporting SWRL rules (Sirin et al., 2007). Initially, it was open source and had a free plugin ready for Protégé. Pellet 3.0, closed source version, is embedded in Stardog RDF database[9] and has a commercial licence. Openllet is an open source continuation (Rattanasawad et al., 2018). SROIQ Description Logic, user-defined datatypes and DL-safe rules are supported by Openllet (Sirin et al., 2007). DL-Safe SWRL rules are to be interpreted which are only applicable to the named individuals in the ontology. With restrictions and other relations, anonymous entities can be introduced in the ontology. Non-DL-safe SWRL rules would endanger decidability. Pellet ascertains DL-safety.

With Pellet and Openllet, by default all SWRL built-ins except the ones for lists and time-date-duration can be applied. For every implementation, one can also introduce and make use of custom built-ins. In Figure 11, main components of Pellet reasoner can be seen.

[9] https://www.stardog.com/

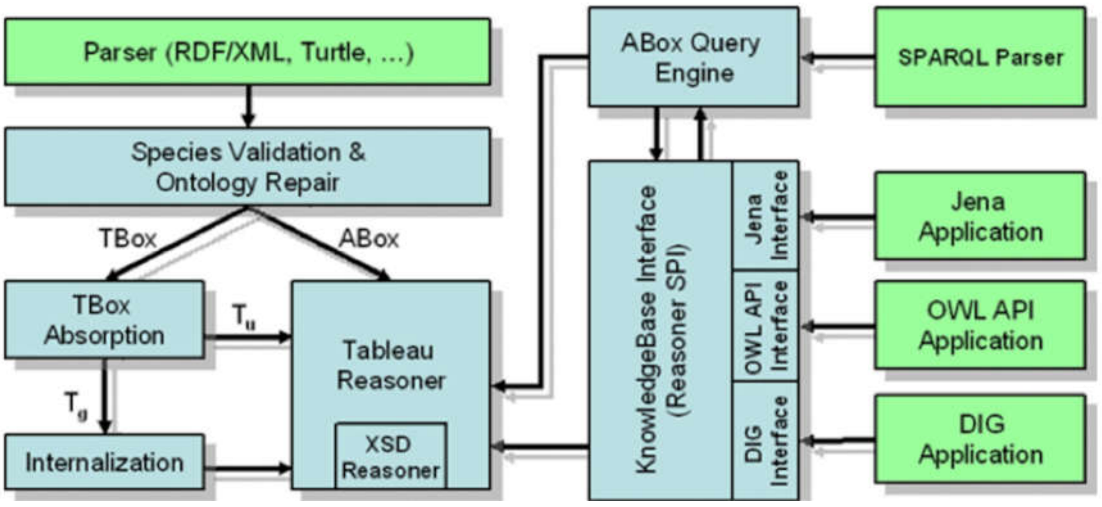


Figure 11. Main components of Pellet reasoner (Sirin et al., 2007).

1.2.3.2. OWLAPI

The OWLAPI is an open source Application Programming Interface (API) in Java for OWL ontology-based applications. Operations that can be implemented with the use of OWLAPI include ontology manipulation, creation and serialization (Horridge and Bechhofer, 2011) . The popular ontology reasoners such as Pellet and Fact++ are supported.

1.2.3.3. Java Topology Suite

The Java Topology Suite (JTS) is a Java API that implements spatial data operations with robust geometric algorithms. It implements Dimensionally Extended 9 Intersection Model (DE-9IM) for computing spatial relationships of two geometries (Clementini et al., 1993). Metric methods such as area, length and withinDistance, overlay methods and buffering are the main functions supported by JTS (URL-5, 2015). JTS is used in the development of applications that support the validation, cleaning, integration and querying of spatial datasets.

1.2.4. Logic

1.2.4.1. First Order Logic

In logic, one deals with statements. A statement (or proposition) is a meaningful sentence that is declarative and has a truth value either (T (true), or F (false) in Boolean logic). Statements are denoted using letters such as p and q. “Run, Forrest Run!” is a sentence. “A man is a human” is a statement. This statement is equivalent (↔) to the statement “Being man implies being human”, which is in propositional logic (“zeroth order logic”) form. There are literals (atomic constants or variables or their negations), and basic constructs (conjunctives) (Hilbert and Ackermann, 1999) such as implication (→), negation (¬), disjunction (OR, ∨), conjunction (AND, ∧), exclusive disjunction (XOR, ⊕), negated conjunction (NAND, ↑) and negated disjunction (NOR, ↓). These conjunctives can be described using the truth tables (Anellis, 2012).

Truth value tables for the propositional conjunctives are listed in Table 8. Tautology (denoted by ⊤) is the state of being always true and contradiction (⊥) is the state of being always false. Negation (¬) switches the truth value. Applying negation twice gives identity. Negation has operator precedence over conjunction and disjunction, which have precedence over implication.

Table 8. Truth tables.

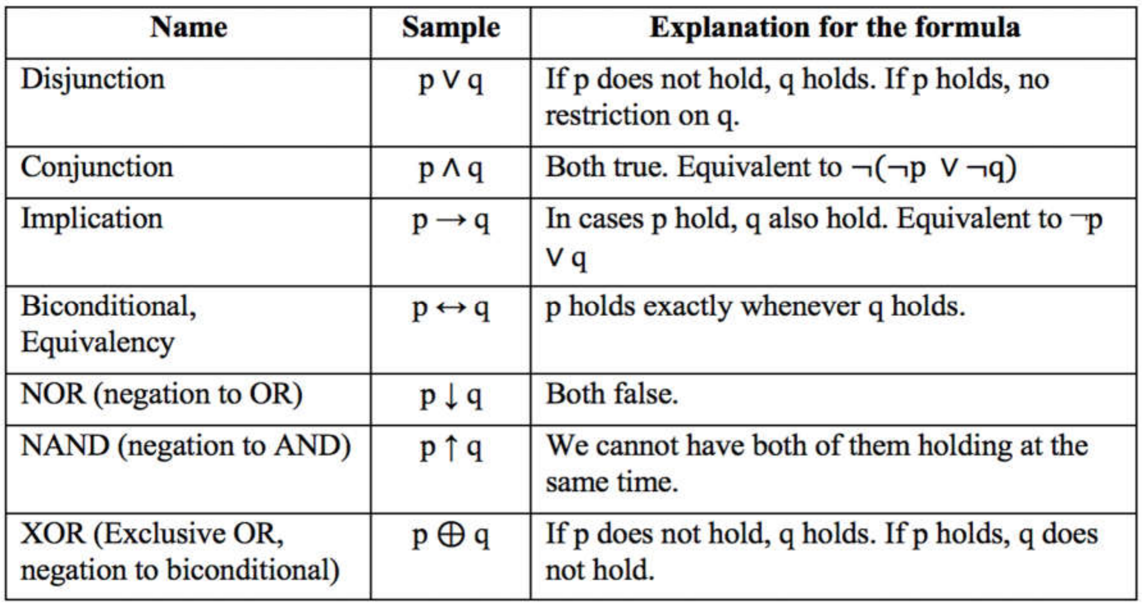
	Unary					**Binary**						
p	**¬p**	**⊤**	**⊥**	**p**	**q**	**p∨q**	**p∧q**	**p→q**	**p↓q**	**p↑q**	**p⊕q**	**p↔q**
T	F	T	F	**T**	**T**	T	T	T	F	F	F	T
F	T	T	F	**F**	**T**	T	F	T	F	T	T	F
				T	**F**	T	F	F	F	T	T	F
				F	**F**	F	F	T	T	T	F	T

A statement can be written in many equivalent forms, called formulas. In formulas, one still uses the same notations as logical constructs. p ∨ q is “p or q” as a logical construct and “either or p is true or q is true” as a statement formula (p ∨ q is Tautology). Sample formulas are given in Table 9.

$p \rightarrow q$ as a formula means that the cases where p holds true, are among the cases q holds true. $p \rightarrow q$ is equivalent to $\neg p \vee q$. If q is false, then $\neg p$ must be true (p must be false). The latter form is the clause form. In a clause, only negations and disjunctions exist. This is also one of the Horn clauses. Only one literal (q) is not negated in disjunction. $p \wedge q \rightarrow r$ is equivalent to $\neg p \vee \neg q \vee r$.

If all of the literals are negated or if only one of them is not negated in the series of disjunctions, it is a Horn clause (Horn, 1951). $p \uparrow q$ has Horn clause form $\neg p \vee \neg q$. Horn clauses are important in logical programming and inferencing. Horn clauses establish rules or facts.

Table 9. Sample formulas in propositional logic using the conjunctives.



Name	Sample	Explanation for the formula
Disjunction	$p \vee q$	If p does not hold, q holds. If p holds, no restriction on q.
Conjunction	$p \wedge q$	Both true. Equivalent to $\neg(\neg p \vee \neg q)$
Implication	$p \rightarrow q$	In cases p hold, q also hold. Equivalent to $\neg p \vee q$
Biconditional, Equivalency	$p \leftrightarrow q$	p holds exactly whenever q holds.
NOR (negation to OR)	$p \downarrow q$	Both false.
NAND (negation to AND)	$p \uparrow q$	We cannot have both of them holding at the same time.
XOR (Exclusive OR, negation to biconditional)	$p \oplus q$	If p does not hold, q holds. If p holds, q does not hold.

"Every feature has a geometry", that can also be written as "for every feature, there exists a geometry", is a statement in FOL form. A predicate such as hasGeometry can describe this situation.

FOL is an extension to propositional logic (Rosen, 2011). In FOL, predicates are used to describe properties of literals. hasFather(ayse, mehmet), is a predicate assigning a property to the literal "ayse", using literal "mehmet", which humans can interpret as ayse having mehmet as a father.

Additionally, there are (cardinality) quantifiers in FOL. “For all” (∀) is the universal quantifier; “There exists” (∃) is the existential quantifier; “There exists exactly one” (∃!) is the uniqueness quantifier. In FOL, these quantifiers apply on the literals (constants or variables). In higher orders of logic, they may apply on predicates (Enderton, 2015). In the study, higher order logic is not within the scope. The level of expressivity of Description Logics variants fall between propositional logic and FOL.

1.2.4.2. OWA and CWA

Hustadt (1994) discuss the different assumptions followed by knowledge representation formalisms and database systems. Database systems are based on three assumptions; CWA domain closure assumption and Unique Name Assumption (UNA). CWA is the assumption of what is not true for a database is considered as false. Domain closure assumption creates a world that contains only the objects exist in that database and every object in that world represented with different names are different entities is the result of UNA.

Knowledge representation systems support OWA, there is no UNA and open-domain is followed. With these assumptions, OWA-based Knowledge representation systems have more freedom. Everything may be possible until explicitly restricted with logical constructs, missing facts are assumed as unknown instead of false. Open-domain assumption is the way to say there can be more entities in the universe than the ones contained in the knowledge base. Any objects which have different names may be represent the same object with no UNA assumption.

The basic constructs of Semantic Web are ontologies and they follow OWA. Several URIs might be assigned to the same resource if the reverse is not explicitly expressed because of non-existence of UNA. Open-domain assumption and non-existence of UNA can be constrained within a knowledge base if needed. This need may arise with the applications such as quality assessment.

1.2.4.3. Negation

Depending on the viewpoint, there are several notions or approximate notions to negation. They base on the logical negation as described above. In the logical programming, a relevant notion is Negation as Failure (NAF), which is used to implement negation under the CWA (Ling, 1990). For NAF, Prolog uses cut operators (Blackburn et al., 2006). YAP Prolog has the cut operator !, which blocks backtracking at that point .(Santos Costa et al., 2002). To have the negation to the predicate man(), the following formula is used.

```
not_man(X) :- man(X) , !, fail.
```

If the fact man(adam), query not_man(adam), then it passes through the cut operator !, encounters the fail operator, but cannot backtrack because of the cut operator. Without the cut operator, it would backtrack and try to find another solution. Because of the cut operator, not_man() cannot be true for the adam atom, even if further rules afterwards suggest otherwise.

In the ontologies, instead of negation, there are complements for OWL classes using owl:complementOf. Any individual that, for certain, cannot be in a class, is put within the complement of the class.

As an example, let C be the class of all features for which the numberOfFloors attribute is less than 7. If in an ontology, the numberOfFloors is a functional datatype property, then the complement of C consists of all features for which the numberOfFloors is defined and is not less than 7. It contains features for which numberOfFloors is not a number. Complement of C does not contain features for which the numberOfFloors is not defined, due to OWA.

If the numberOfFloors datatype property is not functional, then complement of C is empty. As previously, it does not contain features for which numberOfFloors is not defined. Additionally, even if numberOfFloors is defined and is bigger than 7, in the future, it may be defined again with a number less than 7.

1.2.4.4. Defining Constraints in OWL

Domain value related assessment for different schemas in geospatial domain can be done by general rules and using the assertions as consistency facts. How to make OWL behave efficient for integrity constraint (IC) checking is researched and discussed in several studies (Bravo and Rodriguez, 2012; Mäs et al., 2005; Sirin, 2010; Sirin et al., 2008; Sirin and Tao, 2009; Tao et al., 2010; Vallières et al., 2005). It is hard to implement because of the two major properties of the OWL languages. These are the support of OWA and absence of Unique Name Assumption. OWA does not dictate the model having a specific number of instances which can be restricted with the domain necessities. There is not a limit of knowledge for the modelled domain. Everything is possible unless stated otherwise (N. Drummond and Shearer, 2006). OWL does not follow UNA. These properties are efficient for reusable and extensible models for a domain. They also require strict attention. Two instances are not necessarily different or same, unless they are explicitly declared so.

Tao et al. (2010) discuss the integrity constraints in OWL and how to use such constraints for data validation. While OWL is useful for data integration and analysis, not suitable for data validation.

In relational databases integrity constraints are used for data validation. For example, if the datatype of a concept is defined as “string”, it prevents the insertion of different type of data. Expressed statements in OWL may constitute a schema for any domain. Expressions of the schema are expected to be used as constraints for instance data, to help find out the inconsistencies. As the result of OWA and UNA properties, what used as constraints in relational databases causes new inferences in OWL based applications. An example is explained below to illustrate this problem. Sirin and Tao (2009) applied a survey to the people from OWL community to determine their need of integrity constraints for OWL. As a result of the survey, the most needed constraints are summarized as; typing constraints, participation constraints and uniqueness constraints.

Typing constraints, such as domain and range are applied on properties (relations) between resources that are certain classes. owl:allValuesFrom is another typing constraint from OWL. It restricts the values of property should be from a certain class.

An example knowledge base which consists of Parcel and Owner classes and relations between them is shown below.

Every Parcel instance should be related by hasOwner property to the instances from Owner class. This constraint can be defined with Domain and Range property restrictions of OWL. Domain and Range for the hasOwner property is defined as, Parcel and Owner respectively (1). A relation between individuals is asserted with (2). Suppose that this is in Closed World IC. If the individuals are not explicitly defined as in (3), this will cause inconsistency.

```
<owl:ObjectProperty rdf:ID="hasOwner">                                    (1)
```

```
<rdfs:domain rdf:resource="#Parcel" />
<rdfs:range  rdf:resource="#Owner" />
</owl:ObjectProperty>

hasOwner(parcel1, owner 1)                                            (2)

Parcel(parcel1)
Owner(owner1)                                                         (3)
```

If the knowledge base is supposed to be in Open World semantics of OWL, there is no need to explicit assertion of (3). As the result of assertions (1) and (2), (3) will be inferred. Although domain and range constraints supposed to be as ICs, they are used to infer new knowledge (3) in OWL.

Participation constraints is one of the ICs. A property restriction, someValuesFrom, of OWL can be given as example to this IC. Every individual in a class should have at least one relation with other class. “Parcel hasOwner someValuesFrom Owner” is an example of this restriction.

The owl:someValuesFrom constraint is like existential quantifier of Predicate logic. The Following assertions should be asserted for Integrity constraint definition.

```
<owl:Restriction>
  <owl:onProperty rdf:resource="#hasOwner" />
  <owl:someValuesFrom rdf:resource="#Owner" />
</owl:Restriction>

Parcel(parcel2)                                                       (4)
```

Adding (4) may cause inconsistency for the knowledge base in IC, so the following lines (5) should be asserted to constitute a consistent knowledge base.

```
Owner(owner2)
hasOwner(parcel2, owner2)                                             (5)
```

Uniqueness constraint can be explained as the restriction of a property which allows only one assertion of an individual with the same property. Functional property can be used to assert such constraint with OWL.

```
<owl:ObjectProperty rdf:ID="hasOwner">
  <rdfs:domain rdf:resource="#Parcel" />
  <rdfs:range  rdf:resource="#Owner" />                                    (6)
</owl:ObjectProperty>

<owl:FunctionalProperty rdf:about="#hasOwner" />

hasOwner(parcel3; ownerX)
hasOwner(parcel3; ownerY)                                                  (7)
```

(6) defines the hasOwner property as a Functional Property. Following (7) can be used as IC and causes invalidity. In OWL semantics this causes new inference; ownerX and ownerY inferred as same individuals.

As explained with the above example, what is supposed to define constraints may cause new inferences following the OWA. Thus, studies research for a way to implement integrity constraints with OWL.

1.3. Related Work

This part of thesis comprises discussion about the research and approaches in the area of data quality management. In the first part, the proprietary software for spatial data quality management are introduced and examined. Afterwards, the approaches for ontology-based and rule-based data quality management frameworks are examined with comparison to this study.

1.3.1. Software

There are two prominent software solutions in the field of spatial data quality management. These are ArcGIS Data Reviewer from ESRI and 1Integrate from 1Spatial. Both of them support rule-based approach and both of them are proprietary software. They have differences in the types of rules they support. For volunteered data such as OpenStreetMap data, several solutions exist, again proprietary, such as osmValidation[10].

1.3.1.1. ArcGIS Data Reviewer

ArcGIS is a well-known proprietary software for Geographic Information Systems from ESRI company. Data Reviewer extension is produced for making automated quality evaluation with the support of well-known rules. These rules are classified into several categories according to the types of validation checks and are used to create a rule base for the quality checks in terms of consistency of the data to the specifications. The classification of the rules can be seen in Table 10. These are general rules that can be applied to any database independent from its application domain. While geometric validation of the data is categorized as default checks, the main checks such as validating topology and validating domains are categorized as database validation checks. It has a similar approach to parts in the implementation of the study in terms of generalization of the rules while being independent from its application domain. No ontological reasoning is done. It follows Closed World calculations.

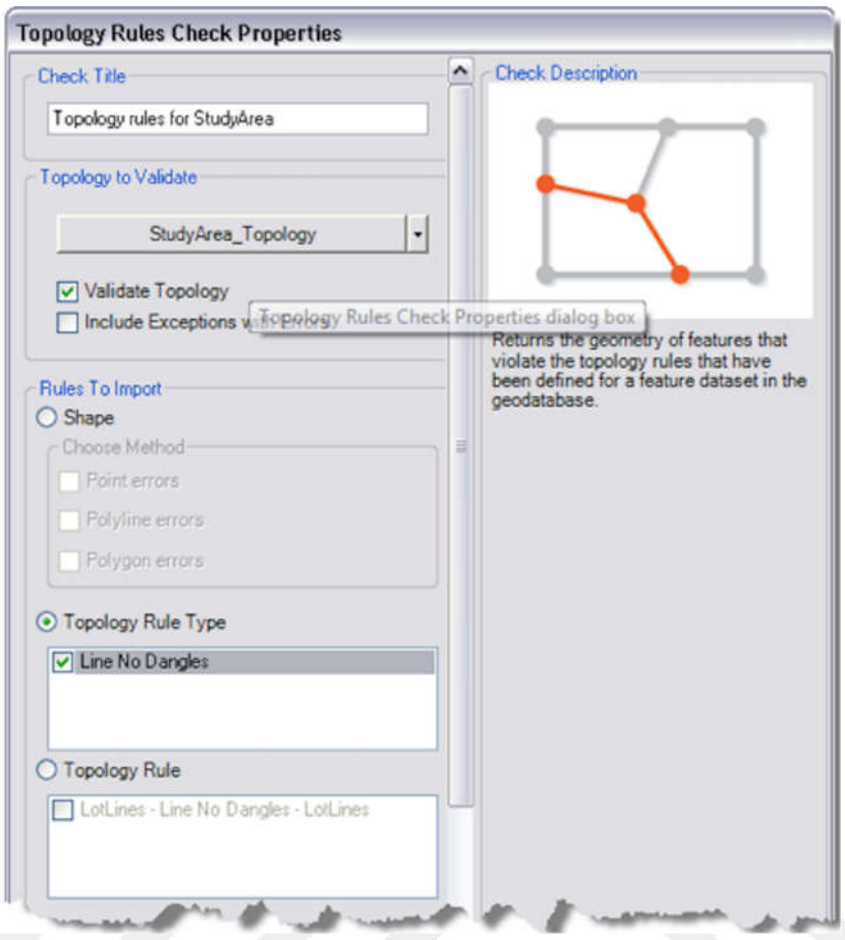
Properties in a relationship class define how objects in the origin relate to objects in the destination. Validating relationship rules is defined to test their consistency. Relationship categorization exists. For instance, “Cardinality” type relations are used to define whether the relationship between two classes is one-to-one, one-to-many or many-to-many. A relationship often needs to be defined in more restrictive terms. In a relationship of parcels and buildings, one may require that each building is associated with a parcel or limit the number of buildings in a parcel.

[10] https://www.npmjs.com/package/osmvalidation

Table 10. Data Reviewer validation categories- Modified from (URL-1, 2017).

Validation checks category	Rule types
Database Validation checks	Network connectivity, validating domains, checking for invalid subtypes and validating relationships, validating a topology against topology rules
Default checks	Finding invalid geometry, finding multipart line features, finding multipart polygons, finding nonlinear segments, finding polylines or paths that close on themselves.
Duplicate Geometry checks	Finding duplicate geometry, finding duplicate vertices
Event checks	Invalid event
Feature on Feature checks	Geometry on geometry, intersection on geometry, polygon overlap/gap is sliver
Polygon checks	Finding polygon slivers, finding invalid whole features, evaluating polygon perimeters and areas, checking cutbacks in lines and polygons, evaluating segment part/path, polyline lengths.
Polyline checks	Checking cutbacks in lines and polygons, evaluating segment part/path and polyline lengths.
Spatial parameter evaluation	Evaluating a features part count, evaluating the intersection count, evaluating feature extents, evaluating vertex counts.
Table checks	Meta characters used to build regular expressions, validating string values, checking for unique ids, comparing table attributes, finding features with a SQL query
Z Value checks	Finding unclosed rings in polygons, evaluating z-values, searching for different z-values at intersections, searching for adjacent vertex elevation changes

Validating a topology of data against topology rules is another similar test with the ones existing in the implementation of the study. There are topology rules classified according to the geometry types of features. For example, if a topology is defined that includes the road and building feature classes, it can be indicated that roads and buildings should not cross. This means that any road crosses with building would be returned by the check. The user interface is represented in Figure 12.



Topology Rules Check Properties

Check Title: Topology rules for StudyArea

Topology to Validate: StudyArea_Topology

[✓] Validate Topology
[ ] Include Exceptions with Errors

Rules To Import
○ Shape
Choose Method
[ ] Point errors
[ ] Polyline errors
[ ] Polygon errors

● Topology Rule Type
[✓] Line No Dangles

○ Topology Rule
[ ] LotLines - Line No Dangles - LotLines

Check Description

Returns the geometry of features that violate the topology rules that have been defined for a feature dataset in the geodatabase.

Figure 12. The Topology Rules Check Properties dialog box.

One uses domain rule validation to ensure that all constraints for a field are met. For instance, the number of flats in building type features can be restricted to be in a given range or enumerated set. The features for which the number of flats is not in the given range or set, are marked as faulty.

1.3.1.2. 1Spatial 1Integrate

1Spatial Management Suite, is a data management software by 1Spatial company. 1Spatial 1Integrate application supports quality assessment with rules-based processing. The group that created Radius Studio for the Laser-Scan company split to create the 1Spatial software. In Radius Studio, initially, fact-pattern-action followed where data sources were represented as facts, and the rules were represented as patterns and actions were represented as the result of the rules (Woodsford, 2007). Initial architecture of Radius Studio is represented in Figure 13.

Watson (2007) discuss the needs of a rule language for quality assessment and criticizes the existing languages in the aspect of fulfilling the desired necessities. Sanderson et al. (2009) have developed a “standard-based rules language”, Spatial Quality Integration Rules (SQUIRL) in the light of the desired necessities. This group redesigned Radius Studio software to implement SQUIRL for 1Spatial group. In INSPIRE Annex 1 testing process, consistency of the datasets to the specifications have been tested. The same group implements such a rules-based approach with 1Integrate. As specified for Data Reviewer it has the similar approach for data quality test providing general rules independent from the domain.

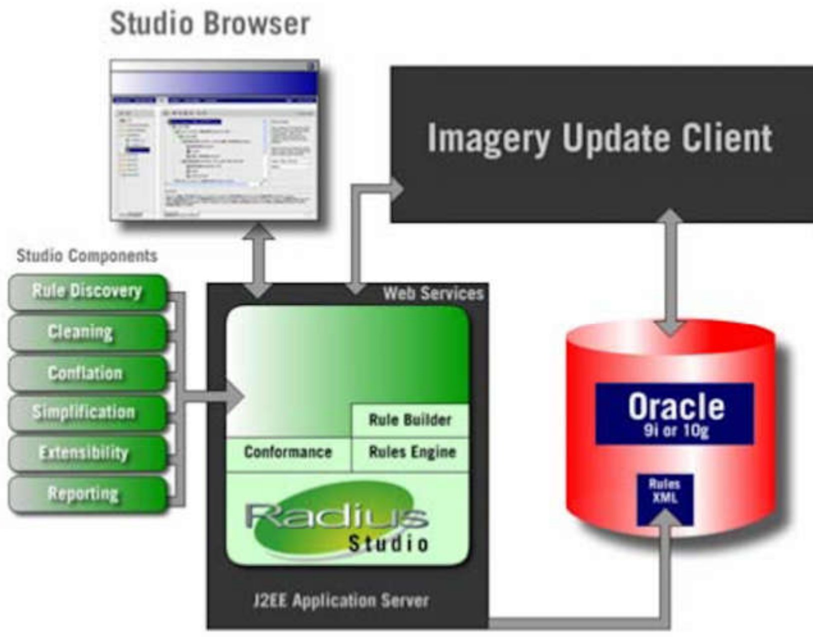


Figure 13. Radius Studio as a multi-tier architecture (Woodsford, 2007).

1Spatial 1Integrate has a limited set of free rules. These are shown in Table 11. Custom rules require paid credits. Rule components consist of parent and child nodes, geometries, conditions, relationships, values, class labels, object labels, aggregate functions and built-in functions. Relationships (scalar or spatial) and value components are used to assess topological consistency or domain consistency as specified in this thesis. The rest of the section describes these components in detail. Parent and child nodes are used to create a rule condition supporting three types of geometries; point, line and polygon.

Table 11. Free rules of 1Spatial.

Essential geometry checks	Duplicate Features Check
	Kickbacks Check
	Duplicate Points Check
	OGC Valid Feature Check
	OGC Simple Feature Check
Network Checks	Closed Lines Check
	Crossing Lines Check
Polygon Checks	Part Intersection Check
	Ring Intersection Check

For 1Integrate, a condition is a high-level, logical test that defines the syntax for a rule or action. A condition can be a comparison of values, a logical operator, looping construct or a combination of them, called a collection.

Two types of relations are applied; scalar and spatial relations. Scalar relations compare any two values (Boolean, integers, real numbers (float/double) or literal strings) with equal, not equal etc. This type of relations is used for attribute tests. Spatial relations are, “beyond”, “contains”, “covered by”, “covers”, “cross”, “disjoint”, “equal”, “intersects”, “overlaps”, “touches”, “within” and “within distance”. 1Spatial classification of spatial relations are demonstrated in Figure 14.

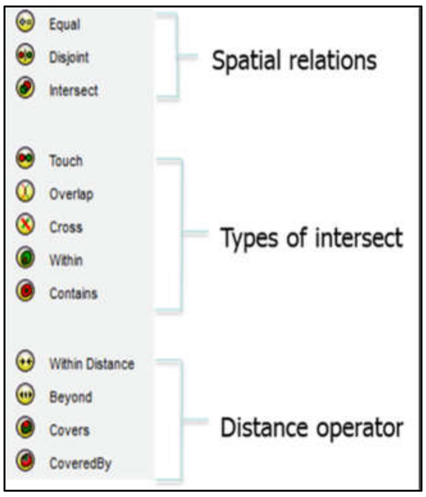


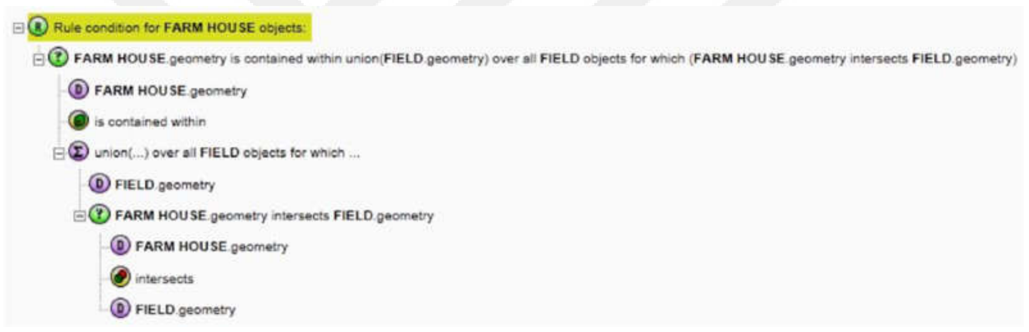
Figure 14. 1Spatial classification of spatial relations.

Values are compared using relationships. The class label identifies classes of objects to be processed. An object label uniquely identifies objects in the same class that are tested against each other in clauses and sub-clauses of a rule.

Aggregate functions calculate a single result from one or more input values, specified as the result of a sub-condition.

Built-in functions are used to apply condition rules to tested classes. They include geometric functions, conversion, mathematical functions etc. Geometric functions include functions for geometric calculations such as area, boundary and convex hull.

Rule components make it possible to create custom rules for any domain or intended task, an example rule creation with 1Integrate can be seen in Figure 15.

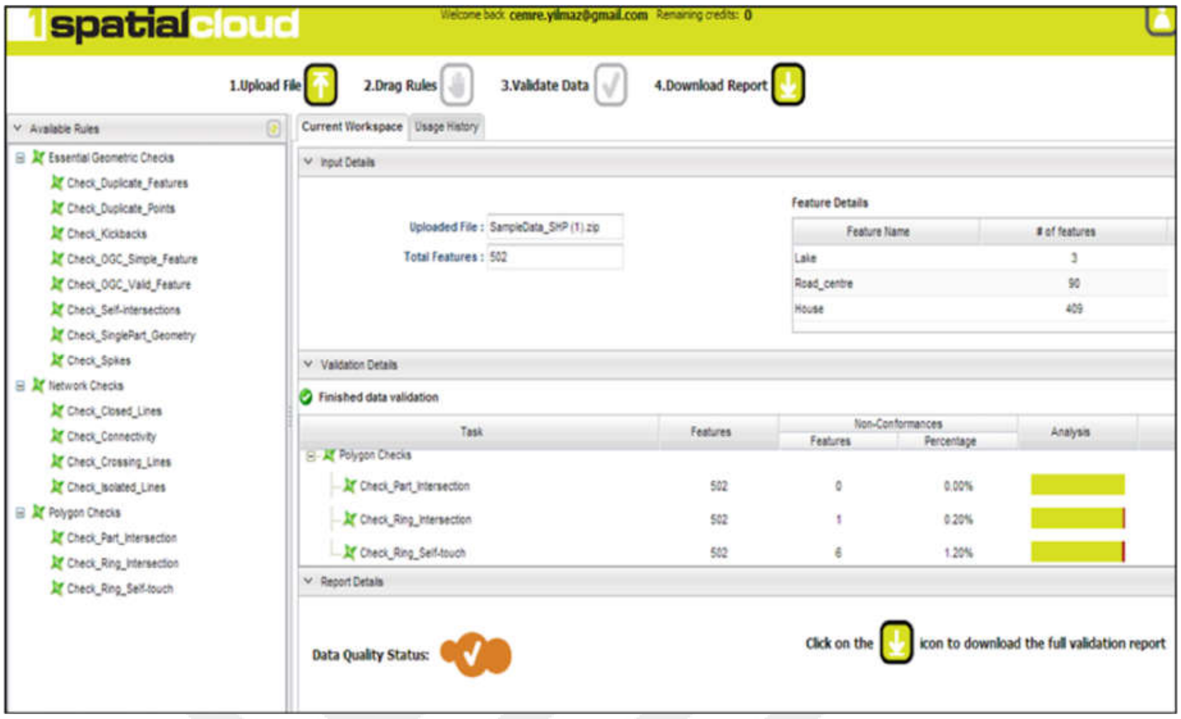


```
⊟ Rule condition for FARM HOUSE objects:
  ⊟ FARM HOUSE.geometry is contained within union(FIELD.geometry) over all FIELD objects for which (FARM HOUSE.geometry intersects FIELD.geometry)
      FARM HOUSE.geometry
      is contained within
    ⊟ union(...) over all FIELD objects for which ...
        FIELD.geometry
      ⊟ FARM HOUSE.geometry intersects FIELD.geometry
          FARM HOUSE.geometry
          intersects
          FIELD.geometry
```

Figure 15. A rule to check “a farm house is contained within some fields.

1Integrate data quality evaluation process follows three steps. Uploading the dataset to 1Integrate online validation system is the initial part of quality assessment. Once dataset uploaded to the system, rules are defined and created according to the specification or the intended task. Then rules-based test is applied to the dataset and a quality report is created as a result of the evaluation process. Total number of features in the dataset and number of failed features are represented in the report. User interface of 1Spatial data quality cloud interface is can be seen in Figure 16.

.



Welcome back cemre.yilmaz@gmail.com Remaining credits: 0

- Essential Geometric Checks
  - Check_Duplicate_Features
  - Check_Duplicate_Points
  - Check_Kickbacks
  - Check_OGC_Simple_Feature
  - Check_OGC_Valid_Feature
  - Check_Self-intersections
  - Check_SinglePart_Geometry
  - Check_Spikes
- Network Checks
  - Check_Closed_Lines
  - Check_Connectivity
  - Check_Crossing_Lines
  - Check_Isolated_Lines
- Polygon Checks
  - Check_Part_Intersection
  - Check_Ring_Intersection
  - Check_Ring_Self-touch

Uploaded File : SampleData_SHP (1).zip
Total Features : 502

Feature Details

Feature Name	# of features
Lake	3
Road_centre	90
House	409

Finished data validation

Task	Features	Non-Conformances		Analysis
		Features	Percentage	
Polygon Checks				
Check_Part_Intersection	502	0	0.00%	
Check_Ring_Intersection	502	1	0.20%	
Check_Ring_Self-touch	502	6	1.20%	

Data Quality Status:

Click on the icon to download the full validation report

Figure 16. 1Spatial's Cloud user interface for free rules.

1.3.2. Literature

In literature, there are conventional spatial data quality and ontology-based assessment approaches.

Mostafavi et al. (2004) can be considered as the first attempt of researching the advantages of using ontologies for rule-based quality assessment. The study is implemented for National Topographic Database (NTDB) of Canada for the compliance test of data against its specification (Canada, 1997). This specification defines the entities, supported geometries and the spatial relations allowed between entities in NTDB. It includes four relations; connection, sharing, adjacency and superimposition. Connection and sharing relations are separated in categories, depending on whether the related entities belong to the same theme or different themes.

The ontology of the NTDB was defined according to the specifications. The logical programming language Prolog is used to formalize ontology and creation of rules for inconsistencies. As a result, knowledge base is created with combining data ontology, dataset and inconsistencies are implemented as Prolog rules as depicted in Figure 17.

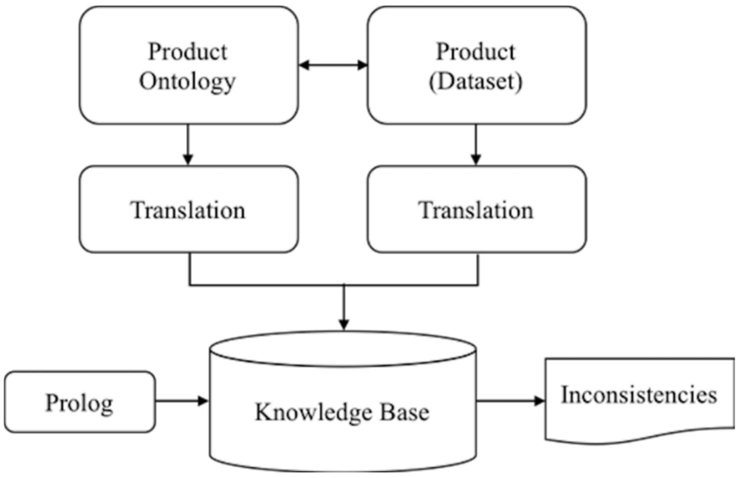
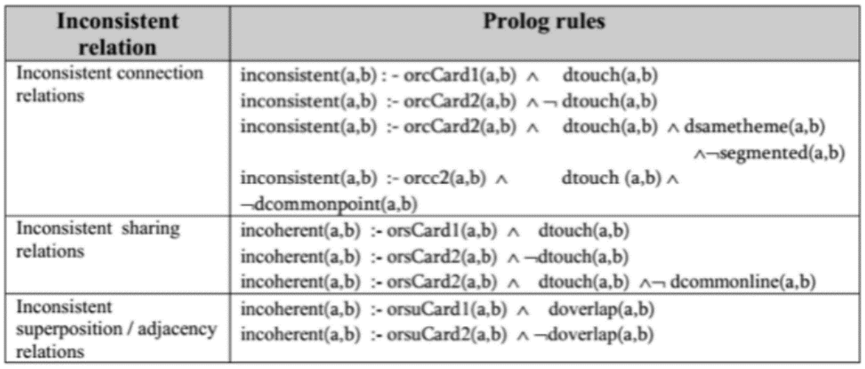


Figure 17. Consistency study for spatial databases (Mostafavi et al., 2004).

Figure 18 includes Prolog rules created to test inconsistent spatial relations (Mostafavi et al., 2004). Data level inconsistencies of relations are starting with ‘o’ represents the ontological level, the letter ‘d’ denote the data level. As a result of the test, they represent the result as the number of inconsistent elements according to the related entity (Mostafavi et al., 2004).



Inconsistent relation	Prolog rules

Inconsistent connection relations	inconsistent(a,b) : - orcCard1(a,b) ∧ dtouch(a,b)
	inconsistent(a,b) :- orcCard2(a,b) ∧ ¬ dtouch(a,b)
	inconsistent(a,b) :- orcCard2(a,b) ∧ dtouch(a,b) ∧ dsametheme(a,b) ∧¬segmented(a,b)
	inconsistent(a,b) :- orcc2(a,b) ∧ dtouch (a,b) ∧ ¬dcommonpoint(a,b)

Inconsistent sharing relations	incoherent(a,b) :- orsCard1(a,b) ∧ dtouch(a,b)
	incoherent(a,b) :- orsCard2(a,b) ∧ ¬dtouch(a,b)
	incoherent(a,b) :- orsCard2(a,b) ∧ dtouch(a,b) ∧¬ dcommonline(a,b)

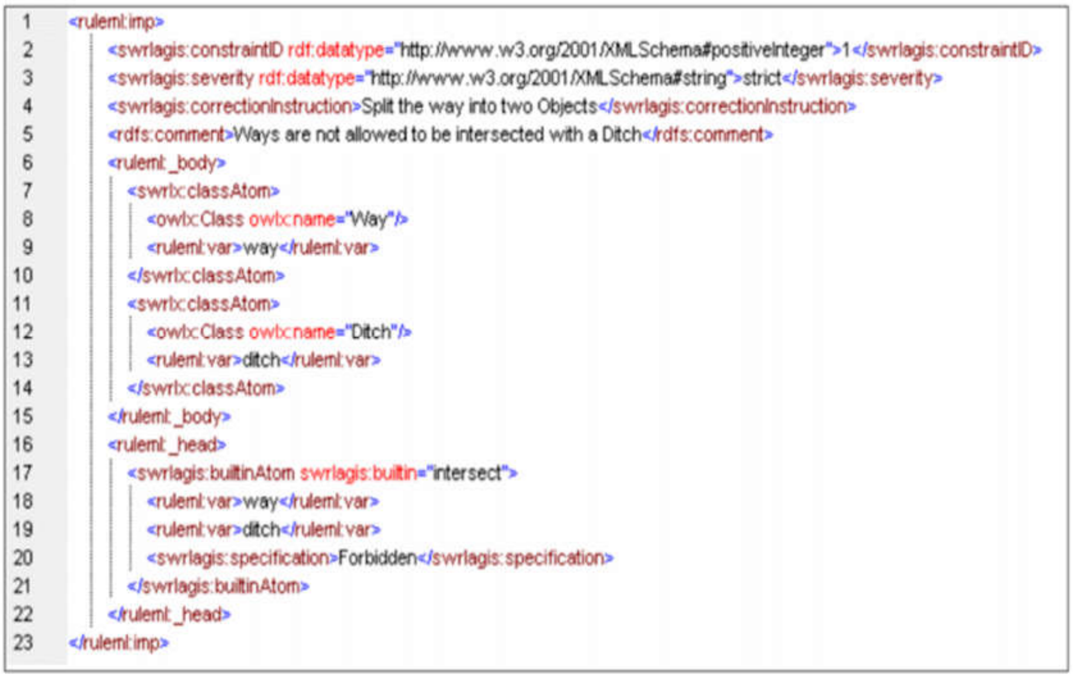
Inconsistent superposition / adjacency relations	incoherent(a,b) :- orsuCard1(a,b) ∧ doverlap(a,b)
	incoherent(a,b) :- orsuCard2(a,b) ∧ ¬doverlap(a,b)

Figure 18. Prolog rules of the study (Mostafavi et al., 2004).

This study was done to validate NTDB against the schema at the time. Despite making use of ontologies, the Closed World notion is prevalent in the study. Although it does the intended task, logical consistency test of data, it is not designed for reusable purposes. The user can locate the data that do not follow that schema; they cannot customize the rules or make similar changes. Changes in the schema require a complete rewrite. In the Prolog part of the study, inconsistency rules are implemented for some of the sources of faults in data.

Wang et al. (2005) developed a system for mobile platforms which is used for specific quality checks to spatial data while gathering from field and warns the user if there is any inconsistency due to newly added data. This system is based on WFS-T services and it serves data as GML. Quality check is mainly done between the data model and the data. Thus, they extend GML application schema with the constraints between features using SWRL. SWRL was chosen because it can be serialized in XML (Wang et al., 2005). Another study by the same team, describes how ontology and SWRL used to define such integrity constraints within GML application schema (Mäs et al., 2005). This is done by extending GML, so that Horn clauses can be implemented. SWRL is not solely about Horn clauses. If the rules are not chosen carefully, the possibly unintended Open World reasoning lead to incorrect results.

Parts of the data model, spatial, topological and semantic quality constraints and combinations of them are defined to express the rules for the domain. As a case study, they consider a geological database. "A hiking way is not allowed to be intersected with a ditch" is one of the constraints created for that study as in Figure 19 (Mäs et al., 2005). When the data collector enters a ditch or a hiking way data to the system, it warns the collector, if there is a forbidden intersection.



```
<ruleml:imp>
    <swrlagis:constraintID rdf:datatype="http://www.w3.org/2001/XMLSchema#positiveInteger">1</swrlagis:constraintID>
    <swrlagis:severity rdf:datatype="http://www.w3.org/2001/XMLSchema#string">strict</swrlagis:severity>
    <swrlagis:correctionInstruction>Split the way into two Objects</swrlagis:correctionInstruction>
    <rdfs:comment>Ways are not allowed to be intersected with a Ditch</rdfs:comment>
    <ruleml:_body>
      <swrlx:classAtom>
        <owlx:Class owlx:name="Way"/>
        <ruleml:var>way</ruleml:var>
      </swrlx:classAtom>
      <swrlx:classAtom>
        <owlx:Class owlx:name="Ditch"/>
        <ruleml:var>ditch</ruleml:var>
      </swrlx:classAtom>
    </ruleml:_body>
    <ruleml:_head>
      <swrlagis:builtinAtom swrlagis:builtin="intersect">
        <ruleml:var>way</ruleml:var>
        <ruleml:var>ditch</ruleml:var>
        <swrlagis:specification>Forbidden</swrlagis:specification>
      </swrlagis:builtinAtom>
    </ruleml:_head>
</ruleml:imp>
```

Figure 19. Forbidden intersection with a strict severity encoded in SWRL (Mäs et al., 2005).

This research investigates data validation in situ, while collecting and entering data from the field using mobile devices. They use SWRL to implement rules that would be not expressible solely by GML This study investigates data quality assessment of produced data according to its specifications. In the central ontology part, SWRL rules follow OWA. Reusability and robustness against ontological inconsistency identifying root of the problem are prime aspects in the study. The data is likely to be already collected by any organization.

Degbelo (2012) create an ontology design pattern (ODP) to model quality aspects of semantic sensor networks. It builds upon ontologies for sensors and observations such as Stimulus Sensor Observation Ontology (Janowicz and Compton, 2010). This pattern is for isolated content ontologies in the same fields. Also, a semantic sensor network data quality content ontology following the proposed pattern is devised and implemented for a single attribute, 'resolution'. The resolution is not given in the sensor characteristics. It is calculated using other characteristics and checked whether the resulting number is above or below a given value.

In this study, various sources of data and regulations exist. There are several comparatively simple SfOs around a common ontology. These SfOs describe different sets of regulations. This study has similarities with the study; both use ontologies as the assessment component of the proposed system. This system also uses SWRL rules to infer the result of quality component. On the other hand, this system supports only one quality component, resolution, and has no spatial predicate and geometric support. SDQO in the thesis study, has some commonality with the concepts in the pattern proposed by Degbelo (2012). In both, data quality processes are modelled as individuals. Importing both ways makes ontologies equivalent. SDQO is ignorant of the ontologies importing it. The design needs to take this into account. The commonality in the regulations and more are reflected in the SDQO.

Fürber (2015) propose a system called Semantic Data Quality Manager (SDQMgr). This framework consists of three components. A user wiki is created for users to define the quality requirements among the predefined data quality dimensions. Second component is Data Quality Management (dqm) vocabulary which is devised for two basic purposes; to represent users' requirements in a formal way and to relate the quality results with the data quality elements (Fürber and Hepp, 2010, 2011). dqm:DataRequirement class is defined as superclass of all data requirements. In case of task-dependent applications the property dqm:appliesFor connects specific requirements with the class dqm:Task. The rules defined in the ontology are executed with associated SPARQL rules to calculate the values of metrics. Metrics implemented with this system are related to attributes. Some of the metrics are; missing values, functional dependency violation, syntax violation (only letters allowed), out of range value (upper limit), out of range value (lower limit), unique value violation. This system implements rule-based system with SPARQL (Fürber and Hepp, 2010). The framework proposed in the thesis has similarities with this system, as the user should define the requirements among the predefined rules. Both systems apply generic quality dimensions. In the framework, quality requirements of the user are to be created by users' requirements. These requirements are based on spatial specifications and a user interface is created to define the SfO. SDQMgr system, makes the assessment of data with SPARQL queries. In the framework, general rules created with SWRL for quality assessment; occasionally SPARQL queries are used. As the result of assessment, data quality values of the inconsistent instances are inferred as new knowledge in the thesis study.

Geisler et al. (2016) propose an ontology-based approach for data quality management of data streams. Noting that ontologies are reusable, human and machine-friendly, they choose to use ontologies as configurations in automatic quality evaluation for data streams. This is the primary commonality with the thesis study. They focus on the problems caused by interruptions in data streams. They create ontology to define quality concepts and relations between them, to model quality assessment process and make it reusable for quality assessment. The ontology created for this study can be seen in Figure 20 (Geisler et al., 2016).

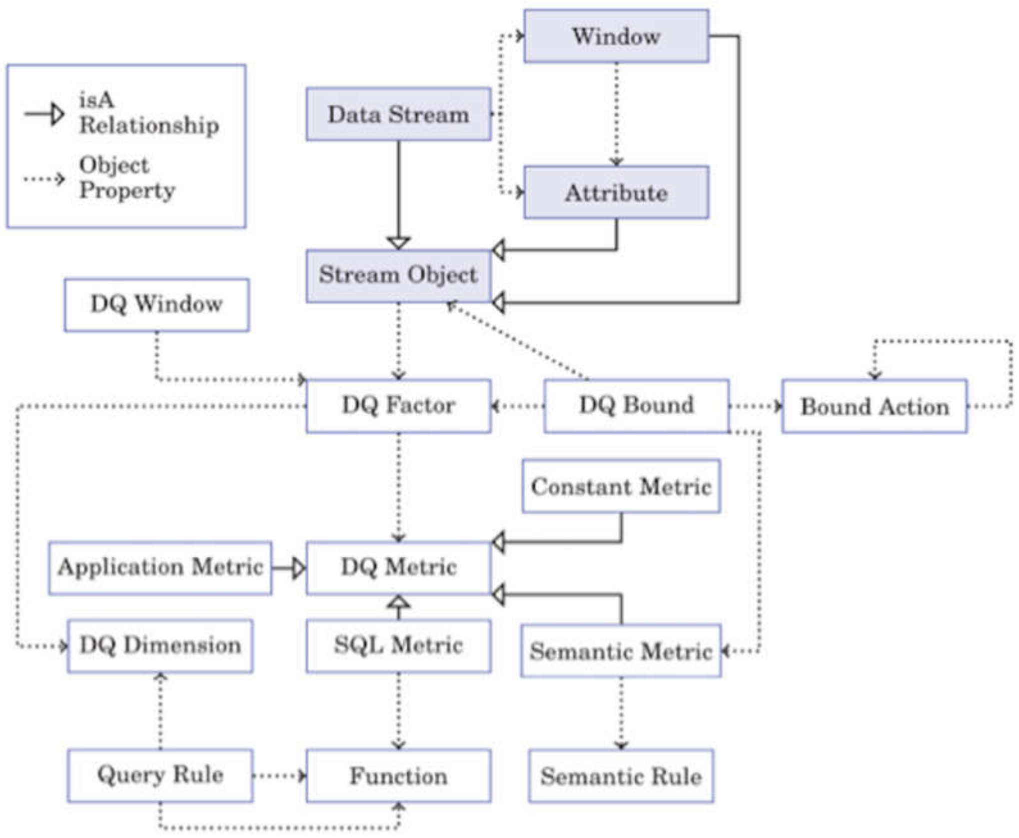


Figure 20. Visualization of the data quality ontology of Geisler et al., (2016b).

The proposed system uses SQL queries for quality assessment. They provide three services, for quality assessment purpose. These are namely query-based, content-based and application-based service. The query-based service is offline. The queries are rewritten to ready them for the other services. For instance, using aggregate functions, attributes such as volume are calculated and viewed. The content-based service applies the quality metrics on the data, also avoiding problems caused by aggregation. With the application-based service, users can create other variables with SQL aggregations that are to serve as data quality metrics. Architecture of the implementation is seen in Figure 21 (Geisler et al., 2016).

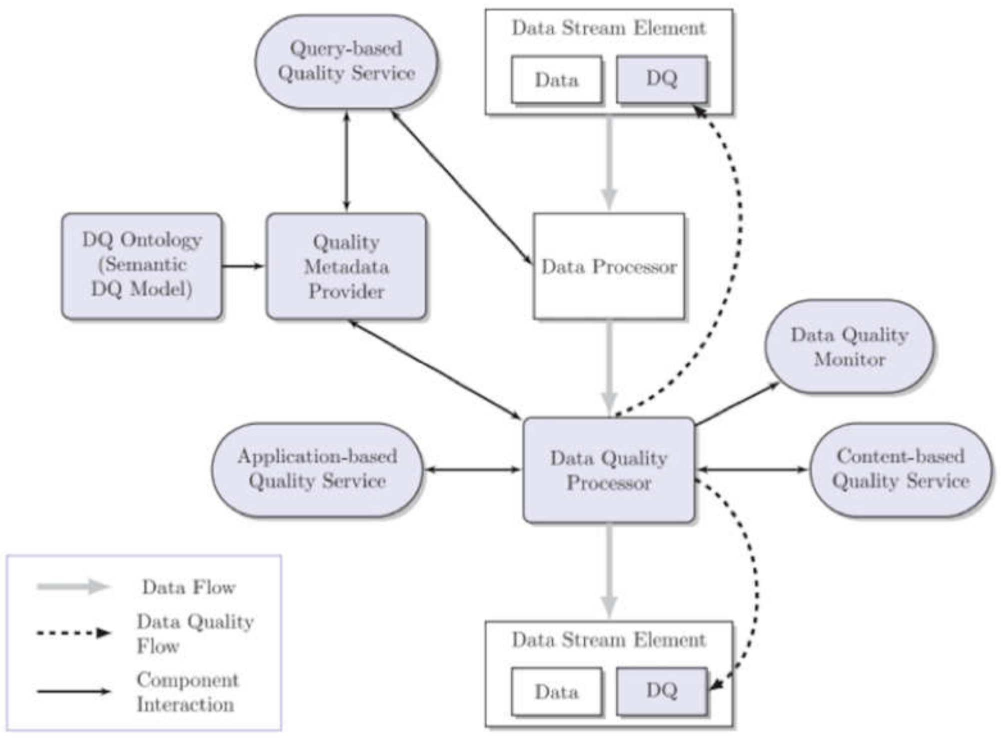


Figure 21. Architecture of the ontology based quality framework (Geisler et al., 2016b).

Ontology created for this system has some common concepts with one of the ontologies in the framework proposed in the thesis, such as quality metrics, quality dimensions and functions. Functions for the quality assessment are implemented as instances in the system with correspondent SPARQL query constructs. There is not a system for users to define their context dependent rules. The metrics and functions for the metrics are certain, system only implements the predefined rules for quality assessment, instead. The results of the queries are calculated with real-time execution and a monitoring system used for viewing results. In the framework, users can define their requirements through SfOs and the framework is designed for assessment of produced data.

Debattista et al. (2016) propose Luzzu, a linked data quality assessment framework that uses an ontology for the back-end (Debattista et al., 2016). It becomes more user friendly with the Luzzu Quality Metric Language (LQML), the domain specific language they design for data quality assessment (Debattista et al., 2015). There are terms in the LQML that correspond to 27 metrics. These are selected from the linked data quality metrics listed in Zaveri et al. (2016). Representational-conciseness category is about how well the data is represented in terms of common best practices and guidelines. Representational

Conciseness, Interoperability, Interpretability and Versatility are the quality dimensions implemented within Luzzu. Keeping URIs short (RC1), Minimal Usage of RDF Data Structures (RC2) and Re-use of Existing Terms (I01) are some of the metrics implemented for this dimension. RC1 is a metric calculates the length of a URI whether longer than a constant number (in this case, 80) (Debattista et al., 2016). Users can invoke these metrics with the associated LQML terms. Similarly, users should define the constraints with the devised SfOs in the proposed framework of this thesis. The results are published according to the Dataset Quality Ontology (daQ) (Debattista et al., 2014). One of the aims of the daQ ontology is to make quality results available for datasets to make comparisons among existing datasets (Debattista et al., 2014). Ontologies are used throughout the assessment process and specification representation. Quality assessment architecture which is represented in Figure 22 (Debattista et al., 2016), is similar with the one implemented in this thesis. User of the system chooses the metrics for the task at hand, and system implements this with the necessary calculations. As a result, a quality report produced including the inconsistent data.

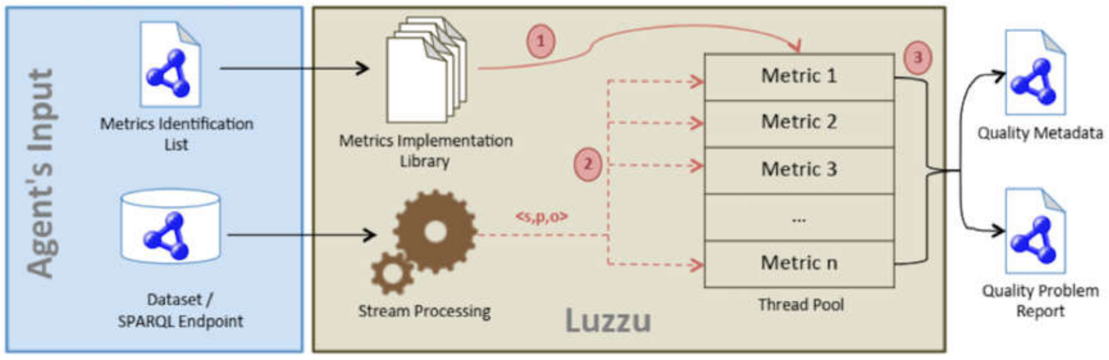


Figure 22. Quality Assessment Workflow in Luzzu System (Debattista et al., 2016).

Henriksson and Kauppinen (2007) propose an ontology-based system to formalize the quality assessment process according to ISO Spatial Data Quality Standards ISO (2002). Although these standards are mainly accepted documents for quality evaluation, there is no practical way for their application. Main purpose of this study is to define a formal way for the application of ISO standards. This study differs from above explained studies with respect to the application purpose.

Zhu (2014) propose ontology-based quality assessment framework for a specific clinical domain (organ transplant). The quality assessment approach followed in the study consists of three basic steps; define, assess and improve. Two types of ontology designed; one for representing data model of the Multi-Organ Transplant (MOT) Electronic Health Record (HER), (MOT HER) ontology, the other is the ontology (Data quality criteria ontology) designed for quality dimensions. Dimensions to test data are determined as; completeness, consistency and logic consistency. Logical consistency has three subclasses; Value in Range, Correct Temporal Sequence, Correct Events According to Clinical Knowledge. They created generic SPARQL queries for the implementation of quality dimensions and criteria for quality assessment. The resultant quality report includes the number of total features and the percentage of faulty instances. This study has similarities with the thesis study; both of them proposes two different ontologies for quality assessment. Their Data quality criteria ontology has similar purpose with the SDQO proposed in this study. The main difference is, while in SDQO, general rules are implemented with SWRL rules, they implement the quality metrics as SPARQL queries in their ontology. In the thesis study, framework is designed to assess spatial data and support of geometries and spatial relations. The system does not support such tools as in the thesis study.

Nash, Wiebensohn, et al. (2011) discuss the automatization of specification rules in agriculture domain. They classify the rules according to value types in the specifications such as obligation, prohibition. Obligation is a value for rules that mandating some action and prohibition is a value for some action that prohibiting some action. Nash, Nikkilä, et al. (2011) explain the implementation of geospatial rules as Interchangeable Rule Format (RIF) and proposes, GeoRIF. RIF is extended with geospatial relations and geometry. This study is similar as they formalize the rules of specifications using ontology and rules. They create rules of specifications as RIF rules, this thesis study implements such rules with SWRL. Although RIF is a W3C Standard in the new Semantic Web layer cake, there are not many reasoners supporting RIF. RIF helps bringing a common ground for interchanging rules. In the framework, similar SfOs are already connected through the SDQO. They use almost the same rule set. Furthermore, RIF embeddings in OWL is possible for the OWL 2 RL profile (W3C, 2013). OWL 2 RL profile has small scope for the purpose of the thesis study.

2. METHODOLOGY AND FINDINGS

Spatial data is increasing in importance with applications to a wide range of domains such as hydrography, transportation, disaster management and agriculture. High quality data is a must for these applications to have reasonable and correct results. This makes data validation important for both producers and users. Currently, there is no consensus in the community on the definition of data quality. For some researchers, data quality relates to how exempt from error that product is; according to some others, conformity with the related specifications defines the quality. Quality is also associated with how far the consumer expectations are met (Devillers and Jeansoulin, 2013). ISO (1986) define quality as the "totality of characteristics of a product that bear on its ability to satisfy stated and implied needs". In this study, the focus is more on the providers' side.

The purpose of this study is to implement rule-based quality assessment methodology for geospatial data with a domain independent way. Rule based approach is applied in three case studies. First study uses Closed World reasoning with Prolog. The other case studies are based on ontologies and Open World paradigm. The second study uses a hybrid approach with queries. These queries are SPARQL queries which can be translated to SPIN and SHACL. The last case study is implemented with the Semantic Web Rule Language (SWRL).

Trabzon city is selected as the study area. Dataset of the base map of Trabzon city is provided by the municipality of Trabzon. For the implementation of the three case studies, a number of layers are used. Cadastre, planning lot, building and road layers are used among the whole dataset. The distinct rules are applied to validate the efficiency of methods. In Figure 23, the study area and the selected layers for implementation are represented.

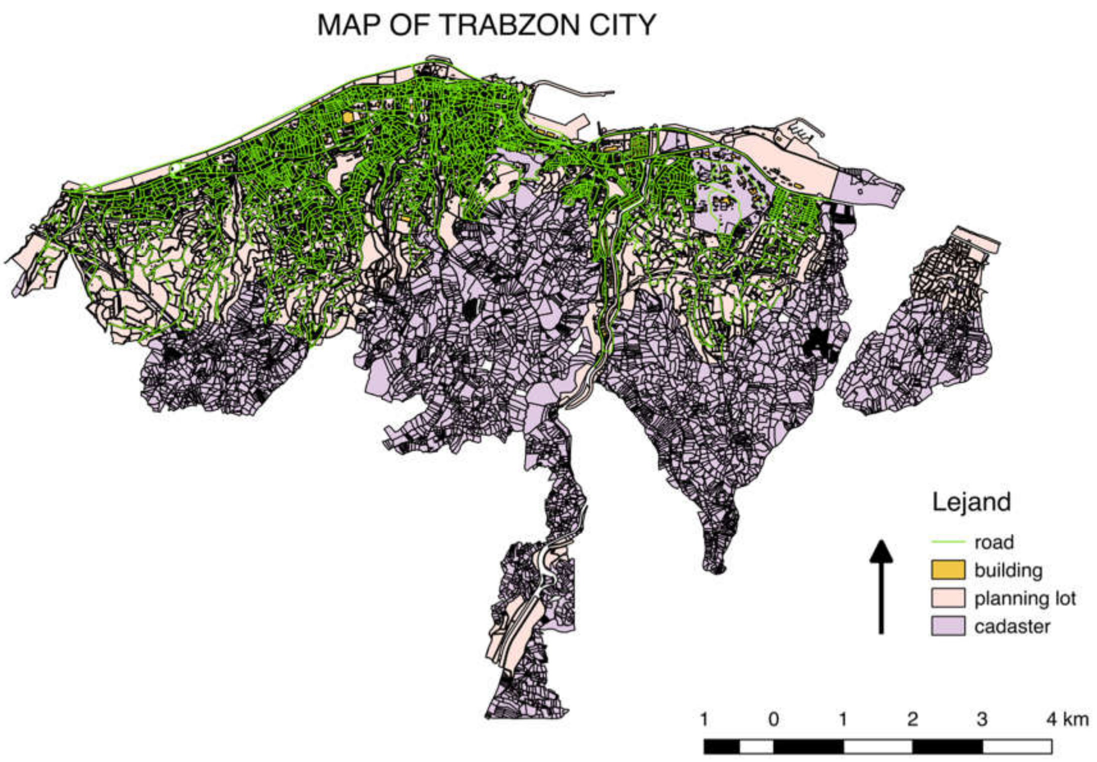


Figure 23. Map of Trabzon, the study area for case studies.

The layers used for the case studies are explained in the related sections. For example, for the first case study, cadastre, planning lot and building layers are used. Information about the layers is given in Table 12.

Table 12. Information about layers used for case studies.

Layer	Number of Features	Size
Road	4395	0.5 MB
Building	20595	5.4 MB
Planning Lot	4901	3.2 MB
Cadastre	33640	18.1 MB

2.1. Closed World Spatial Quality Evaluation with Logical Programming

Prolog which stands for Programming in Logic, is the flagship logical programming language and incorporates Horn clauses within FOL (Blackburn et al., 2006; Clocksin and Mellish, 2012; Sterling and Shapiro, 1994; Van Emden and Kowalski, 1976). FOL encompasses the expressivity of the Description Logic family, which is also related with Datalog programming language (Baader et al., 2003; Ceri et al., 1989; Horrocks, 2002; Hustadt et al., 2004). It is desired to implement rule-based approach using Prolog for quality assessment.

In this study, Prolog is mainly used for formalism of rules related with any domain in a declarative way, creating constraint rules as specified in the data specification. Prolog has a declarative way of rule formalization; it has been used in geospatial domain for various purposes such as quality evaluation (Mostafavi et al., 2004), data management (Srinivasan and Richards, 1993), geospatial risk analysis (Vaz et al., 2010). Once one creates the knowledge base with rules, the next step is to implement a program for the desired goals. In this case study, the goals are related with discovering about the inconsistent data.

Prolog has several kinds of implementations such as SWI Prolog, XSB Prolog, GNU Prolog and YAP Prolog (Costa et al., 2012). Vaz et al. (2007) develop Spatial-Yap, a library for spatial operations, for YAP Prolog. Due to the need of geospatial execution, YAP Prolog is chosen in the first case study. Furthermore, YAP Prolog implementation has spatial User Defined Indexing (UDI) capability (Vaz et al., 2009). Spatial-YAP, UDI and the implementation of a case study with those components are explained in the subsections below.

2.1.1. Spatial-YAP

Spatial-YAP is an extension built on top of YAP Prolog, to support geospatial terms and geospatial predicates to work with these terms, all these conforming to the OGC standards.

The architecture of Spatial-YAP on top of YAP Prolog is shown in Figure 24 (Vaz et al., 2007). It has two main components; YAP Prolog system and MySQL RDBMS coupled with MYDDAS interface (MySQL/Yap Deductive Database System). MYDDAS is fundamentally responsible for translation of logic queries into SQL statements and conversion into YAP terms of MySQL attributes. It explores the YapTAB tabling engine for solving recursive queries involving database goals (Vaz et al., 2007). MYDDAS interface also supports MySQL geometry types shown in Figure 25. Besides, it supports spatial operators and visualization predicates. Spatial operators are based on Geos Library (URL-6, 2008). Geos library aims to contain the complete functionality of JTS in C++. It supports all the functions defined in OGC (1999) as spatial predicate functions and spatial operators, as well as specific JTS enhanced topology functions. Visualization predicates are supported with the Open Graphics Library (OpenGL) (Segal et al., 2010), allegro[11] and postscript [12] libraries.

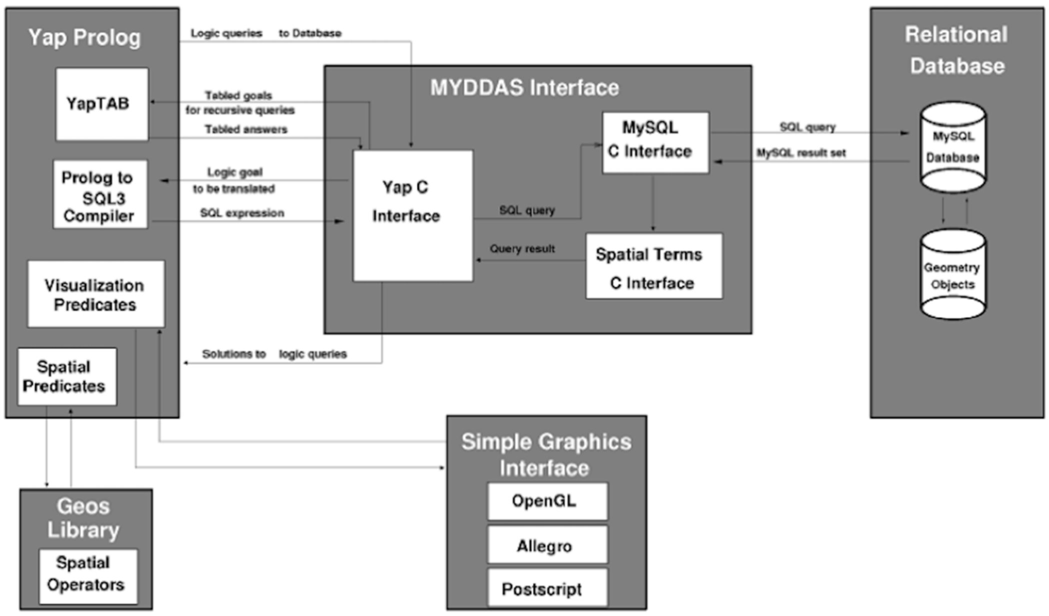


Figure 24. Spatial YAP Blueprint (Vaz et. al, 2007).

Spatial predicates are organized into three main groups and explained with their syntax in Tables 13-15 (Modified from Vaz et al. , 2007).

[11] http://liballeg.org/
[12] http://pslib.sourceforge.net/

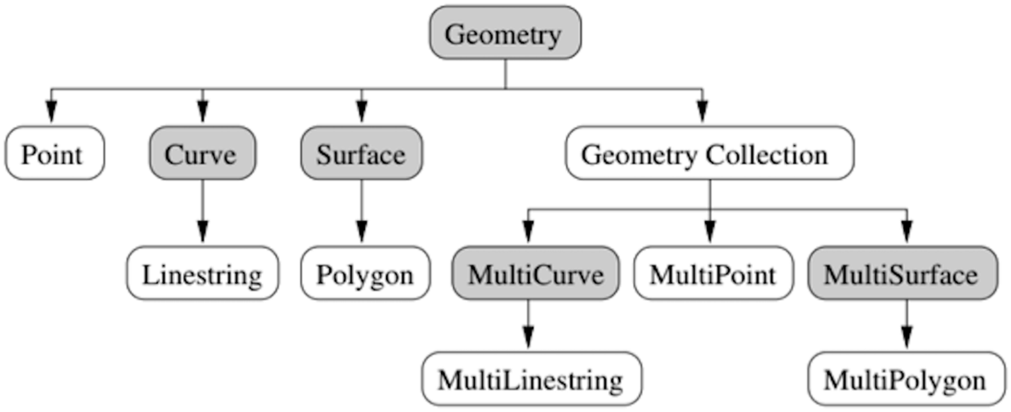


Figure 25. Geometry types, grey types are abstract geometries.

Predicates for spatial analysis, constructs new spatial terms as result of input terms. It also has metric predicates in Table 13.

Table 13. Predicates that support Spatial Analysis in Spatial-YAP library (Modified from Vaz 2007) .

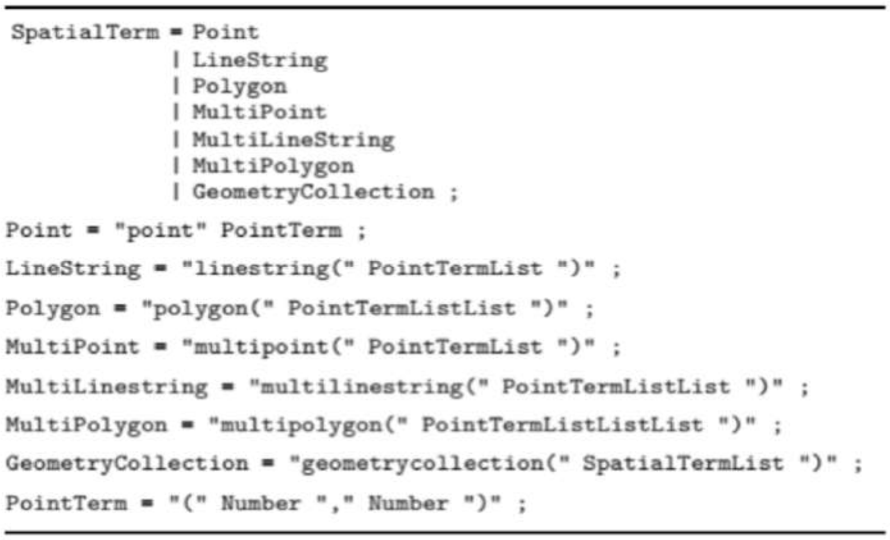
Predicate	**Explanation**
ogc_distance(+Geometry1,+Geometry2, Distance)	Returns the shortest Distance between any two points in the two geometries.
ogc_envelope(+Geometry,-Polygon)	Returns the rectangle bounding the Geometry as a Polygon. The polygon is defined by the corner points of the bounding box.
ogc_boundary(+Geometry,-Boundary)	Returns the Geometry Boundary.
ogc buffer(+Geometry,+Distance,-Buffer)	Returns a geometry (Buffer) that represents all points whose distance from this Geometry is less than or equal to Distance.
ogc_convex hull(+Geometry,-ConvexHull)	Returns a geometry (ConvexHull) that is the convex hull of this Geometry.
ogc_intersection(+Geometry1,+Geometry2,-Geometry)	Returns the Geometry that represents the point set intersection of Geometry1 with Geometry2.
ogc union(+Geometry1,+Geometry2,-Geometry)	Returns the Geometry that represents the point set union of Geometry1 and Geometry2.
ogc difference(+Geometry1,+Geometry2,-Geometry)	Returns the Geometry that represents the point set difference of Geometry1and Geometry2.
ogc symmetric difference(+Geometry1,+Geometry2,-Geometry)	Returns the Geometry that represents the point set symmetric difference of Geometry1 and Geometry2.

Predicates on specific spatial terms. These especially for specific geometry type of spatial terms in Table 14.

Table 14. Predicates on specific spatial terms, especially geometry type spatial terms. (Modified from Vaz 2007)

Geometry Type	Predicate	Explanation
Linestring/ Multilinestring	ogc_is_closed(+Geometry)	Succeeds if the Geometry is closed.
	ogc_is_ring(+Geometry)	Succeeds if the Geometry is ringed.
Polygon/ Multipolygon	ogc_area(+Geometry, ?Area)	Returns the area of the geometry.
	ogc_centroid(+Geometry, ?GeomPoint)	Returns the centroid of the geometry.

Spatial properties, including spatial relations between spatial terms Table 15. Spatial terms should conform to the grammar in Figure 26 (Vaz et al., 2007). The syntax is similar to the "Well Known Text" of OGC specification.



```
SpatialTerm = Point
            | LineString
            | Polygon
            | MultiPoint
            | MultiLineString
            | MultiPolygon
            | GeometryCollection ;
Point = "point" PointTerm ;
LineString = "linestring(" PointTermList ")" ;
Polygon = "polygon(" PointTermListList ")" ;
MultiPoint = "multipoint(" PointTermList ")" ;
MultiLinestring = "multilinestring(" PointTermListList ")" ;
MultiPolygon = "multipolygon(" PointTermListListList ")" ;
GeometryCollection = "geometrycollection(" SpatialTermList ")" ;
PointTerm = "(" Number "," Number ")" ;
```

Figure 26. Extended Backus-Naur form of Spatial Terms (Vaz et al., 2007).

Table 15. Predicates on spatial properties (Modified from Vaz 2007) .

Predicate	Explanation
ogc_is_empty(+Geometry)	Succeeds if the Geometry corresponds to the empty set.
ogc_is_simple(+Geometry)	Succeeds if the Geometry is simple.
ogc_equals(+Geometry1,+Geometry2)	Succeeds if Geometry1 and Geometry2 are spatially equal.
ogc_disjoint(+Geometry1,+Geometry2)	Succeeds if Geometry1 and Geometry2 are spatially disjoint, i.e., if the given geometry's do no intersect.
ogc_touches(+Geometry1,+Geometry2)	Succeeds if Geometry1 spatially touches Geometry2. Two geometries spatially touch if the interior of both geometries do not intersect, but the boundary of one of the geometries intersects either the boundary or the interior of the other.
ogc_within (+Geometry1,+Geometry2)	Succeeds if Geometry1 is completely within Geometry2.
ogc_overlaps(+Geometry1,+Geometry2)	Succeeds if Geometry1 spatially overlaps Geometry2. The term spatially overlaps is used if two geometries intersect and their intersection results in a geometry of the same dimension but not equal to either of the given geometries.
ogc_crosses(+Geometry1,+Geometry2)	Succeeds if Geometry1 spatially crosses Geometry2. The term spatially crosses denotes a spatial relation when two given geometries intersect, and their intersection results in a geometry that has a dimension that is one less than the maximum dimension of the two given geometries; the intersection is not equal to any of the two given geometries.
ogc_intersects(+Geometry1,+Geometry2)	Succeeds if Geometry1 spatially intersects Geometry2.
ogc_contains(+Geometry1,+Geometry2)	Succeeds if Geometry1 is completely contained in Geometry2.

2.1.1.1. User Defined Indexing

Prolog systems use unification for query answering with hash-based indexing that is good for answering the equation-based queries as follows.

```
?- p(A), q(B), A=B.
? p(A), q(A).
```

The way in which Prolog matches two terms is called unification. With unification, there are two terms and it is inspected whether they can be made to represent the same structure (Bramer, 2005).

It is not possible to answer queries based on ranges or including comparatives with hash-based indexing. Finding all values that are larger than some X can be given example to "comparative queries" and finding all values that are between two predefined boundaries can be given as example to "ranges queries".

Implementing indexing types has been an active study area for Relational Data Base Management Systems (RDBMS). An important operation is to compute whether two geographical objects *intersect*. An object's minimum bounding rectangle is used for R-Tree indexing in most popular spatial databases such as PostGIS. It helps quickly find all objects in a given area, e.g., "find all towns in Trabzon city". In Spatial Yap library, User Defined Indexing is applied with the help of MySQL R-Tree Indexing.

UDI is extensible indexing system that allows the programmer to define specialized indexing structures to take advantage of the semantics of the terms (Vaz et al., 2009). Then UDI can be applied to spatial terms with improvements because every spatial system relies heavily on indexing. For example, the following query can be given as an example to syntax of UDI within a Prolog query,

```
?- p(A), B && A, q(B), overlaps(A,B).
```

The binary constraint "&&" triggers a search on the call q(B), searching only for spatial terms B which are spatially close to A, saving precious time by avoiding evaluating unnecessary overlap calls. In the next section a case study implemented with Spatial Yap and UDI is used to creating queries for finding out inconsistent data within a domain.

2.1.2. Case Study 1: Prolog based Quality Assessment of Spatial Data

For the first case study, Prolog rules are defined for the base map data (cadastre, buildings, roads, planning zone) of Trabzon province in Turkey. To define the rules, the

Turkish Large-Scale Map and Map Production Regulation (LSMMPR) is used. The framework designed for Prolog based quality assessment is shown in Figure 27.

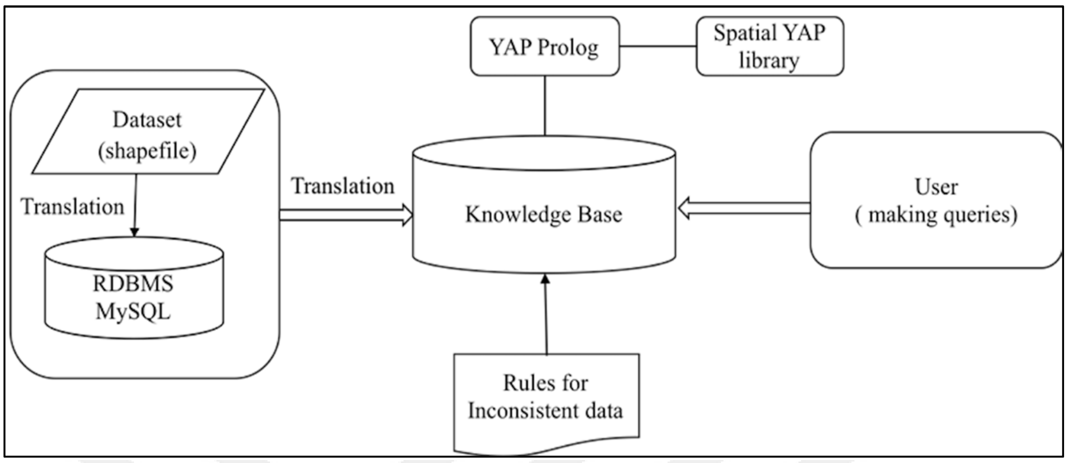


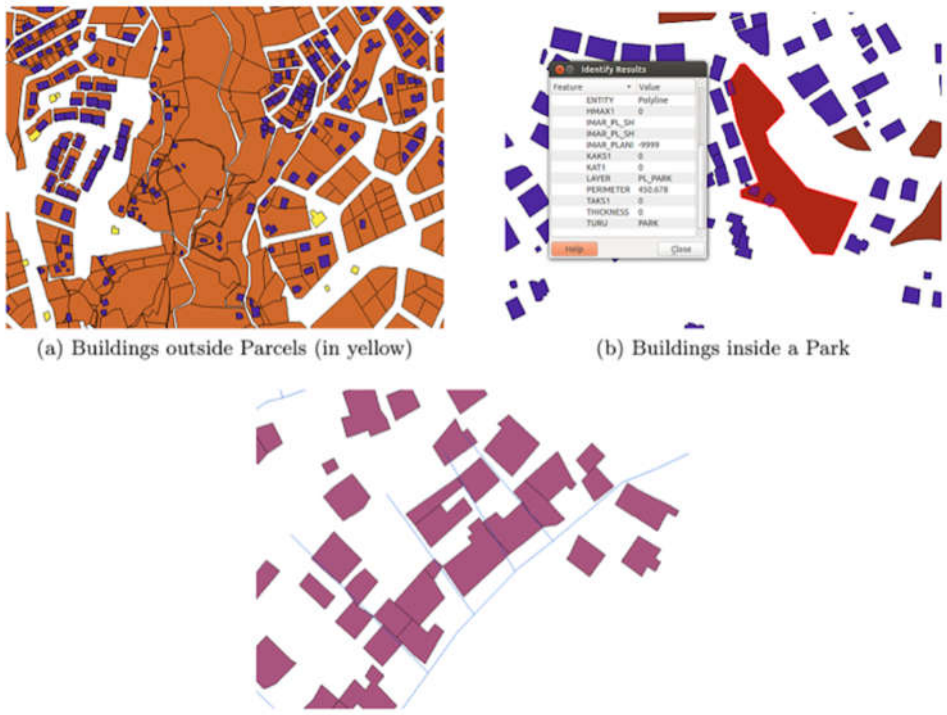
Figure 27. Prolog based quality assessment components.

This regulation is a basis for the institutions in Turkey responsible to produce large scaled base maps. According to this regulation there are no rules representing the spatial relations between spatial details. The lack of a thorough set of rules in the specification is an important issue, but not the focus of this study. Even though the regulations do not explicitly enforce relations between features, one can detect many errors in plain sight on the dataset.

Examples include:

- Buildings that are outside of the cadastral parcel;
- Roads that intersects with buildings;
- Planning zone area that has type Park contains buildings.

In Figure 28, these problems are depicted. There is a high number of buildings outside cadastral parcels.



Identify Results

Feature	Value
ENTITY	Polyline
HMAX1	0
IMAR_PL_SH	
IMAR_PL_SH	
IMAR_PLANI	-9999
KAKS1	0
KAT1	0
LAYER	PL_PARK
PERIMETER	450.678
TAKS1	0
THICKNESS	0
TURU	PARK

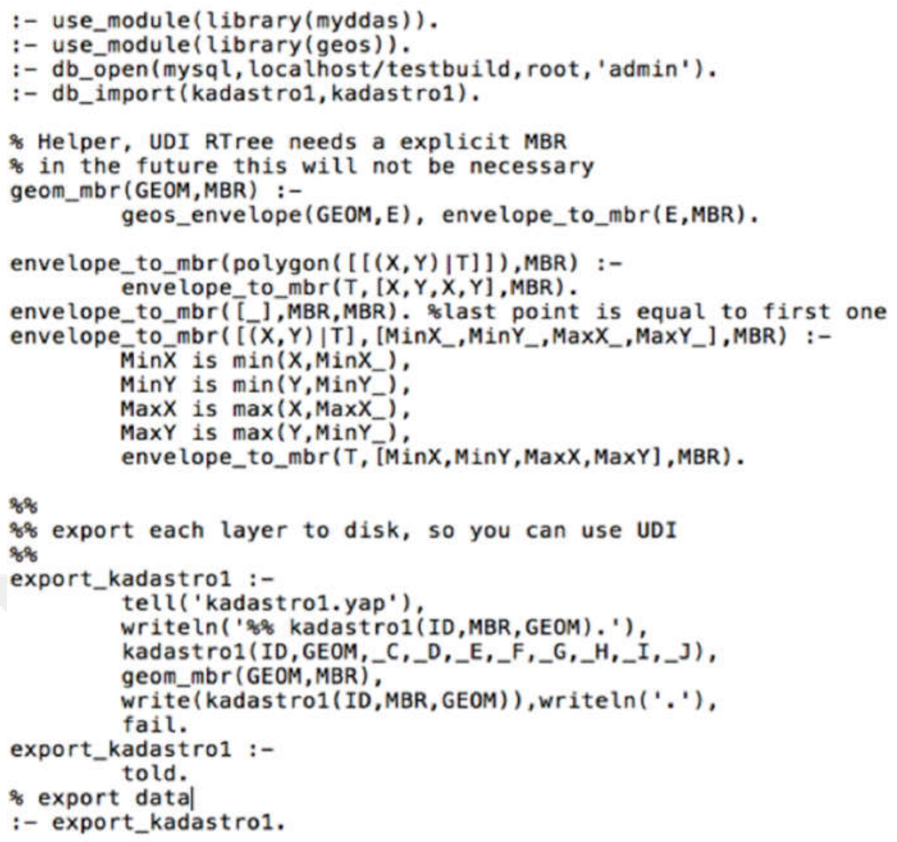
(a) Buildings outside Parcels (in yellow)

(b) Buildings inside a Park

(c) Roads intersecting with Buildings

Figure 28. Existing problems in the sample data, base map of Trabzon.

YAP Prolog supports two ways for data access. One option is to access data from databases, second is to transfer data from databases to Prolog. An example Prolog file for data transformation is demonstrated in Figure 29. Data exported from database act as Prolog facts and constitutes a knowledge base with the rules written for inconsistencies. MySQL database is supported by Yap Prolog, it is possible to manipulate data without exporting to Yap Prolog.



```
:- use_module(library(myddas)).
:- use_module(library(geos)).
:- db_open(mysql,localhost/testbuild,root,'admin').
:- db_import(kadastro1,kadastro1).

% Helper, UDI RTree needs a explicit MBR
% in the future this will not be necessary
geom_mbr(GEOM,MBR) :-
        geos_envelope(GEOM,E), envelope_to_mbr(E,MBR).

envelope_to_mbr(polygon([[(X,Y)|T]]),MBR) :-
        envelope_to_mbr(T,[X,Y,X,Y],MBR).
envelope_to_mbr([_],MBR,MBR). %last point is equal to first one
envelope_to_mbr([(X,Y)|T],[MinX_,MinY_,MaxX_,MaxY_],MBR) :-
        MinX is min(X,MinX_),
        MinY is min(Y,MinY_),
        MaxX is max(X,MaxX_),
        MaxY is max(Y,MinY_),
        envelope_to_mbr(T,[MinX,MinY,MaxX,MaxY],MBR).

%%
%% export each layer to disk, so you can use UDI
%%
export_kadastro1 :-
        tell('kadastro1.yap'),
        writeln('%% kadastro1(ID,MBR,GEOM).'),
        kadastro1(ID,GEOM,_C,_D,_E,_F,_G,_H,_I,_J),
        geom_mbr(GEOM,MBR),
        write(kadastro1(ID,MBR,GEOM)),writeln('.'),
        fail.
export_kadastro1 :-
        told.
% export data
:- export_kadastro1.
```

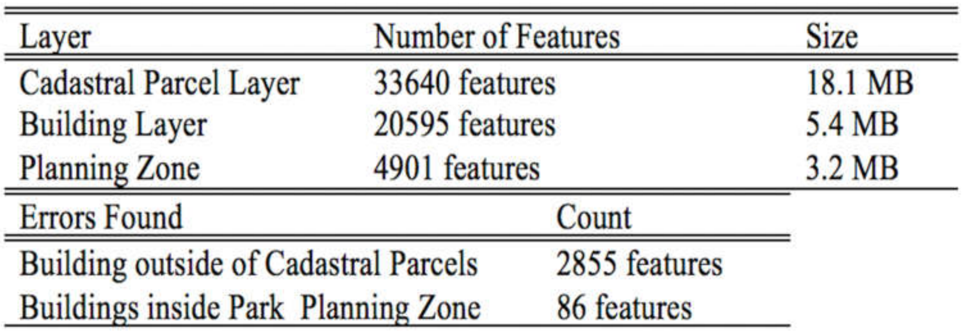
Figure 29. Prolog code for data transformation from database to Prolog internal database.

In the following lines the rules used to query inconsistent data in the sample dataset is represented in Prolog Programming Language. This is applied with YAP Prolog. The code is presented in Figure 30.

```
:- nogc.
:- op(700,xfx,'&&'). %% UDI search operator precedence
A '&&' B :- attributes:get_all_atts(A,C), attributes:put_att_term(A,overlap(C,B)).
:- use_module(library(geos)).
:- use_module(library(lists)).

% MBR is in 2 arg
:- udi(build(-,+,-,-)).
:- ['build'].
:- udi(kadastro1(-,+,-)).
:- ['kadastro1'].
:- udi(Imar (-,+,-,+)).
:- [' Imar '].


%Rule for buildings overlaps the cadastre parcels
related(A,B) :-   build(A,MBRA,GEOMA,_),   MBRB && MBRA, kadastro1(B,MBRB,GEOMB),
geos_overlaps(GEOMA,GEOMB).

%Find out the buildings which do not have any relation with any cadastre parcel,
not_related(X):-build(X,_,_,_),not related(X,Y).
:- findall(A,(build(A,_,_,_), related(A)), L), writeln(L), length(L,LL), writeln(LL).

%Rule for “Buildings which have distance smaller than 5 meters between the parcel which is inside”
bp_inside(X,Y):-build(X,MBRX,GEOMX,_),MBRY && MBRX,
                        kadastro1(Y,MBRY,GEOMY),
                        geos_within(GEOMX,GEOMY).

dist_pc(P,T,Q):- bp_inside(P,T),
                        build(P,_,GEOMA,_),
                        kadastro1(T,_,polygon([B|_])),
                        geos_distance(GEOMA,linestring(B),Q), Q>0.0.

%Find out the buildings which are not inside any parcel; overlapped and intersected ones
not_icinde(X):-build(X,_,_,_), not bpicinde(X,Y).
:- findall(A,(kadastro1(A,_,_), bpicinde(A)), L), writeln(L), length(L,LL), writeln(LL).

%Rule for area of the buildings from the attributes of the data,
build_area(A,AREA):-build(A,MBRA,GEOMA,AREA).

%Calculate the area of each parcel with SpatialYAP
cbuild_area(ID,CAREA):-build(ID,MBRA,GEOMA,_), geos_area(GEOMA,CAREA).

%Find out the parcels which have bigger area than the entered area value for data.
diff_area(ID,CAREA, EPSILON):-build(ID,MBRA,GEOMA,AREA), geos_area(GEOMA,CAREA),
                                            EPSILON is abs(AREA - CAREA), EPSILON < 1 / (10 ^ (2)).
dist_pc(X,Y,Z):-build(X,MBRA,GEOMA,_),MBRB &&
MBRA,kadastro1(Y,MBRB,GEOMB),geos_distance(GEOMA,GEOMB,Z),Z>0.0.

%Find out the 0.0 valued  "area" attribute in the building layer,
zero_area(A):-build_area(A,AREA),AREA = 0.0.

%Test Buildings Inside Planing Zone With Type Park
Within(A,B) :-   Imar(B,Mbrb,Geomb,'Park'),   Mbra && Mbrb, Build(A,Mbra,Geoma,_),
Geos_Within(Geoma,Geomb).
```

Figure 30. Prolog rules.

The result of validation applied in Prolog is shown Figure 31.



Layer	Number of Features	Size
Cadastral Parcel Layer	33640 features	18.1 MB
Building Layer	20595 features	5.4 MB
Planning Zone	4901 features	3.2 MB

Errors Found	Count
Building outside of Cadastral Parcels	2855 features
Buildings inside Park Planning Zone	86 features

Figure 31. Prolog Case Study results

2.2. Ontology Based Quality Assessment for Spatial Data

2.2.1. Ontology and Data Quality Evaluation

Ontologies are central to the Semantic Web. They are designed with aspects of reusability in mind. New users with eventually different datasets should be able to make use of the same ontologies with little change. In geospatial domain, “ontology deals with the totality of geospatial concepts, categories, relations and processes and with their interrelations at different resolutions” (Frank, 1997; Mostafavi et al., 2004; Spaccapietra et al., 2004). Spatial domain ontologies can be used to define all the “concepts”, “attributes”, and “interrelations” of concepts. Many quality frameworks as explained in the previous section design quality frameworks with ontologies. The main advantage of the ontology-based application is the reusability and interoperability of the frameworks. Quality management frameworks are designed for quality assessment and representation of the quality results with respect to the application domain rules. In the thesis study, it is researched how to apply constraints with ontology-based systems and there are two ways; queries and rules. As an initial study, it is defined what kinds of constraints are to be implemented. Specifications of some institutions and studies about formalizations of the spatial rules are examined (INSPIRE, 2014a; Servigne et al., 2000; Vallières et al., 2005; Watson, 2007). How to implement constraints for spatial domain is examined. Specifications have common types of rules for spatial objects such as topological rules including spatial relations, attribute rules for spatial data. Some rules are defined to implement with queries.

For Semantic Web, the basic W3C standard for querying is the SPARQL Query Language for RDF (SPARQL) (W3C, 2008b). OWL reasoning capabilities and expressiveness allows consistency checks for ontologies using reasoners such as PelletSpatial. Using the Region Connection Calculus (RCC) and its composition tables, PelletSpatial checks consistency. RCC has several variants depending on the number of relations. RCC8 is the best well-known, but there are also RCC5, RCC15, RCC23, RCC62, etc. (Cohn et al., 1997; Neng et al., 2016).

To implement a rule such as, “Building data must not intersect with road data” with SPARQL query, system should be designed as supporting spatial relations and geometry data types. For a geospatial query such as; “Find the buildings which intersect with road data”, relationship between the classes should be calculated for instances. For this purpose, GeoSPARQL is used.

GeoSPARQL is published as an OGC standard specification (OGC, 2012). The OGC GeoSPARQL standard supports representing and querying geospatial data on the Semantic Web. GeoSPARQL defines a vocabulary for representing geospatial data in RDF, and it defines an extension to the SPARQL query language for processing geospatial data (OGC, 2012). Its ontology has three basic classes; geo:SpatialObject, geo:Feature, geo: Geometry as illustrated Figure 32. Both geo: and ogc: prefixes are used for GeoSPARQL ontology.

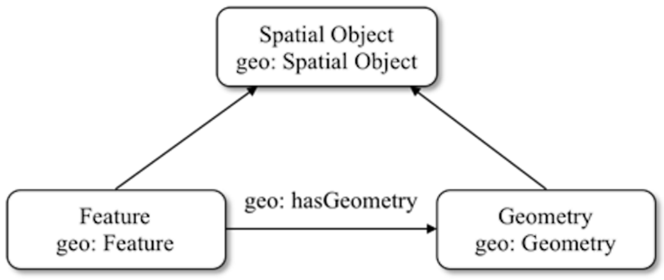
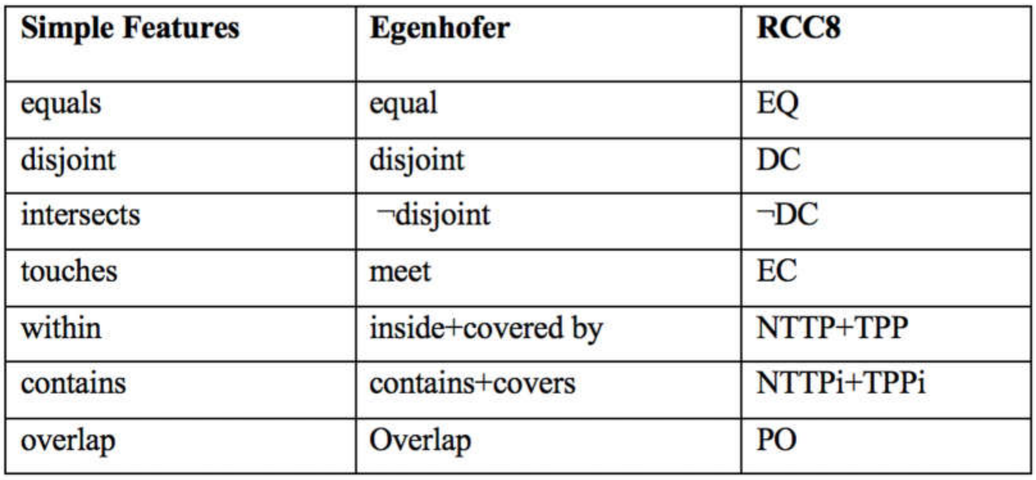


Figure 32. GeoSPARQL classes.

Text (geo:asWKT) for geospatial data. The geo:asWKT and geo:asGML properties link the geometry entities to the geometry literal representations. Values for these properties use the ogc:wktLiteral and gml:gmlLiteral data types respectively (OGC, 2012). Furthermore, Simple Features (OGC, 1999), DE-9IM (Egenhofer and Herring, 1990) and

Region Connection Calculus 8 (RCC8) (Randell et al., 1992) relations are used as topological relations as seen in Table 16. GeoSPARQL is used to take advantage of its spatial relations support for querying the data defined in RDF.

Table 16. Topological relations; Simple Features, Egenhofer DE-9IM, RCC8.



Simple Features	**Egenhofer**	**RCC8**
equals	equal	EQ
disjoint	disjoint	DC
intersects	¬disjoint	¬DC
touches	meet	EC
within	inside+covered by	NTTP+TPP
contains	contains+covers	NTTPi+TPPi
overlap	Overlap	PO

A comparison with Egenhofer and RCC8 relations is shown in Table 17 (OGC, 2012). The RCC8 relations are depicted in Figure 33.

Table 17. Egenhofer DE-9IM and RCC8 spatial relations (OGC, 2012)

RCC8	RCC8 Long Name	9 -IM
EQ (x,y)	x is identical with y	equal
DC (x,y)	x is disconnected from y	disjoint
¬DC (x,y)	x is not (disconnected) to y	intersects
EC (x,y)	x is externally connected to y	meet
NTTP (x,y)	x is a non-tangential proper part of y	inside
NTTPi (x,y)	x is non-tangential proper part inverse part inverse of y	contains
PO (x,y)	x partially overlaps y	overlap
TPP (x,y)	x is a tangential proper part of y	coveredBy
TPPi (x,y)	x is a tangential proper part inverse of y	covers

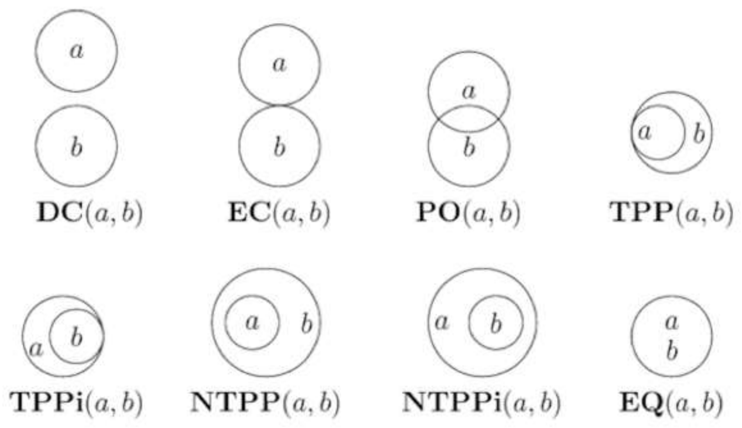


Figure 33. RCC8 relations.

It is seen that if x and y have EC relation and y and z have NTPPi relation, then x and Z must have DC relation. This way, one constructs the composition table (Bechhofer et al., 2003). OS[13], the mapping agency of Great Britain, has developed an ontology with some of the RCC8 relations for features in parts of Great Britain (Goodwin, 2005; Goodwin et al., 2008). PelletSpatial reasoner is a -now discontinued- reasoner for checking validity of RCC8 relations in a RCC8 ontology. Stocker and Sirin (2009) check and find errors in OS data using PelletSpatial.

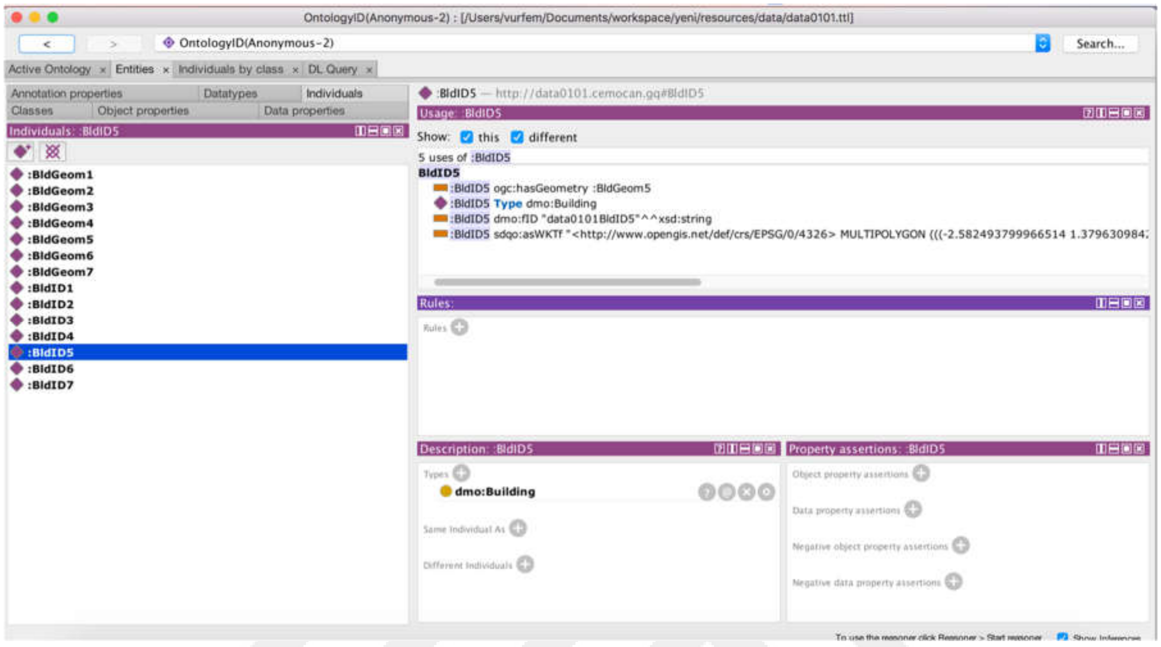
2.2.2. Case Study 2: Quality Assessment with GeoSPARQL Queries

For the case study, the base map of Trabzon is used as data, and several types of inconsistency related problems within data are identified, considering the specifications. Building, cadastral parcel and road data are used as sample data. The database is a MySQL relational database. A sub-collection of geometries conforming to OGC standards with QGIS is created and exported to MySQL. MySQL is open source, competitive, well-performing and popular. Open alternatives include PostGIS.

As the ontology editor, Protégé (versions 4.3 and 3.4.2) is used with its plugins. Protégé performs well and is free and user-friendly. A view from Protégé is shown in Figure

[13] https://www.ordnancesurvey.co.uk/

34. Alternatives include TopBraid Composer (Alatrish, 2013). An ontology associated with GeoSPARQL SpatialObject class and subclasses are created.


OntologyID(Anonymous-2) : [/Users/vurfem/Documents/workspace/yeni/resources/data/data0101.ttl]

Individuals: :BldID5

:BldGeom1
:BldGeom2
:BldGeom3
:BldGeom4
:BldGeom5
:BldGeom6
:BldGeom7
:BldID1
:BldID2
:BldID3
:BldID4
:BldID5
:BldID6
:BldID7

:BldID5 — http://data0101.cemocan.gq#BldID5

Usage: :BldID5

Show: ☑ this ☑ different

5 uses of :BldID5

```
BldID5
    :BldID5 ogc:hasGeometry :BldGeom5
    :BldID5 Type dmo:Building
    :BldID5 dmo:fID "data0101BldID5"^^xsd:string
    :BldID5 sdqo:asWKTF "<http://www.opengis.net/def/crs/EPSG/0/4326> MULTIPOLYGON (((-2.582493799966514 1.379630984
```

Description: :BldID5

Types
dmo:Building

Property assertions: :BldID5

Figure 34. Protégé ontology editor with the ontology.

To import spatial data into Protégé, DataMaster plug-in is used with version 3.4.2. as shown in in Figure 35. Each table in the database is imported as a subclass of "Feature" class which is subclass of "Spatial Object", associated with elements in Geometry subclass. Geometries are used as "asWKT" for each type of feature.

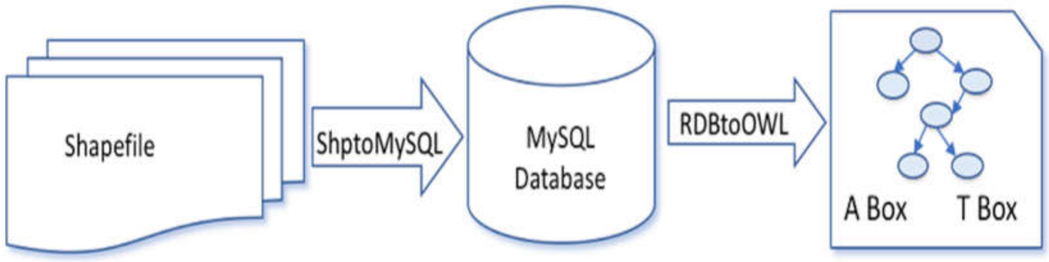


Figure 35. Conversion of spatial data to OWL with DataMaster, Protégé Plugin.

GeoSPARQL standard is implemented by Parliament, Oracle Spatial and Strabon. Finally, to find out inconsistencies, Parliament, an efficient triple store that implements

GeoSPARQL, is used for querying. Parliament creates spatial indices of input data for more efficient implementation. After converting the ontology and saving it as RDF, queries supported with OGC functions, e.g. sfIntersects, sfOverlaps, started to work.

Four types of queries are implemented taking into account classifications provided in literature (Michael F Goodchild, 1995; Servigne et al., 2000). In Figure 36, Road instance and its connection to its geometry is represented.

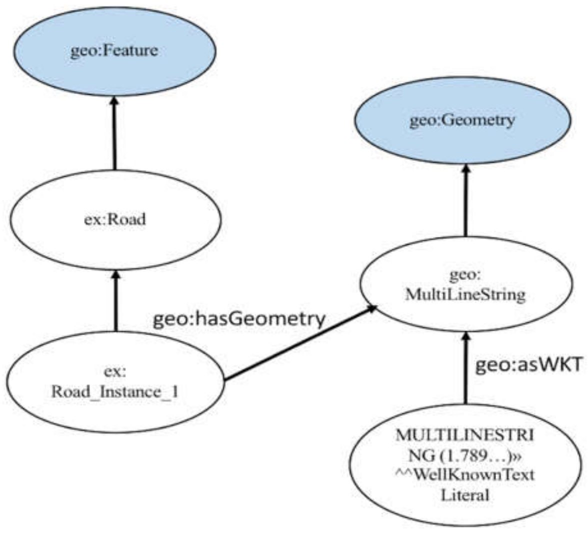
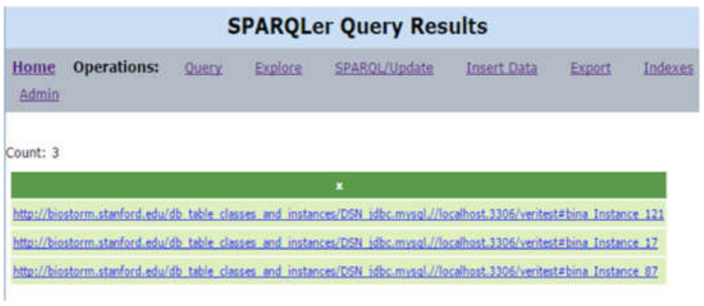


Figure 36. Relation of a "road instance" with its related geometry.

Query 1) "Find buildings that have more than seven floors." (where this is restricted).

```
SELECT ?x
WHERE {?x rdf:type ouront:bina .
?x ouront:bina.katadedi ?y .
FILTER( ?y > 7 ) .}
```

Only one instance satisfied this simple query as seen in Figure 37. (bina@tr = building@en, katadedi@tr = numfloors@en). The first query is an example for a general, non-spatial, given attribute-related situation. It features an attribute accuracy problem (Michael F Goodchild, 1995).



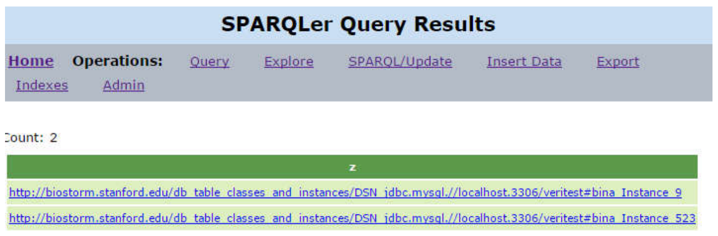
```
Count: 3

x
http://biostorm.stanford.edu/db_table_classes_and_instances/DSN_jdbc.mysql.//localhost.3306/veritest#bina_Instance_121
http://biostorm.stanford.edu/db_table_classes_and_instances/DSN_jdbc.mysql.//localhost.3306/veritest#bina_Instance_17
http://biostorm.stanford.edu/db_table_classes_and_instances/DSN_jdbc.mysql.//localhost.3306/veritest#bina_Instance_87
```

Figure 37. Result of the first query.

Query 2) "Find buildings that intersect with roads".

```
SELECT ?z
WHERE {
?x ouront:yol.asWKT ?y.
?z ouront:bina.asWKT ?b.
FILTER (geof:sfIntersects(?y,?b)) }
```

There are two buildings that intersect with roads as seen in Figure 38 (yol@tr = road@en). The second query is a simple query that has geospatial component, using OGC simple feature relations. It features a topo-semantic inconsistency, a type of logical inconsistency (Servigne et al., 2000).



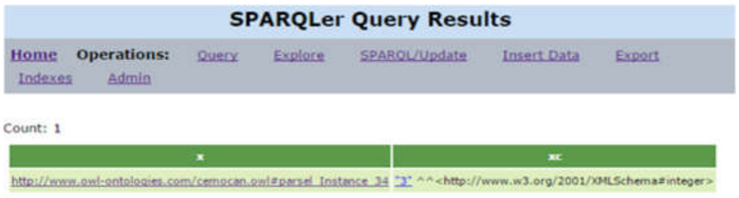
```
Count: 2

z
http://biostorm.stanford.edu/db_table_classes_and_instances/DSN_jdbc.mysql.//localhost.3306/veritest#bina_Instance_9
http://biostorm.stanford.edu/db_table_classes_and_instances/DSN_jdbc.mysql.//localhost.3306/veritest#bina_Instance_523
```

Figure 38. Result of the second query.

Query 3) "Find parcels that intersect with more than 2 buildings"

```
SELECT ?x (COUNT(?x) as ?xc)
WHERE {?x ouront2:parsel.asWKT ?p .
?z ouront:bina.asWKT ?b .
FILTER(geof:sfIntersects(?p, ?b))}
GROUP BY ?x
HAVING ( ?xc >2)
```

There is only one such parcel, and it intersects with three buildings (parsel@tr = parcel@en) as seen in Figure 39.



SPARQLer Query Results

Home Operations: Query Explore SPARQL/Update Insert Data Export Indexes Admin

Count: 1

x	xc
http://www.owl-ontologies.com/cemocan.owl#parsel_Instance_34	"3" ^^<http://www.w3.org/2001/XMLSchema#integer>

Figure 39. Result of the third query

The third query uses more advanced functions related to SPARQL (ARQ type aggregates, here), also present in GeoSPARQL. This also features a topo-semantic error (Servigne et al., 2000).

4) “Find roads whose endpoints are close to the boundary of some building smaller or equal to 0.5”.

```
SELECT ?y
WHERE {
?y ouront:yol.asWKT ?yw .
?b ouront:bina.asWKT ?bw .
BIND (geo:boundary(?yw) as ?n ) .
BIND (geo:boundary(?bw) as ?k ) .
BIND (geo:buffer(?k, 0.5) as ?kb ) .
FILTER (geof:sfIntersects(?n, ?kb) ) .
}
```

No road instance satisfied the query.

When the bound for distance is selected correctly, the result of fourth query helps us solve the following problem that occurs occasionally. The surveyors sometimes do not complete line detail when it is nearby a border of a polygon detail. This is a geometric error (Servigne et al., 2000), which is visually hard to detect. The data, sub collection of geometries in the thesis study do not have such a situation according to the query (that should return a superset), and if the bound is correct, the afore-mentioned situation did not occur in the thesis study. These queries worked slowly with the whole dataset on an i5 PC with 4GB memory. Restricting the dataset to a town, and a sub-collection of geometries is created.

2.2.3. Ontology Based Framework for Quality Assessment with SWRL Rules

The quality of the spatial data is important for both decision-making and transaction accuracy. According to the specifications, restrictions can be classified in mainly two categories; restrictions about attributes and restrictions involving spatial relations. As spatial data is used in many different domains, they all have different specifications for data to comply with. It is desired to make a framework that can be applicable to spatial data, independent from the domain. In literature, rule-based methods are proposed for quality assessment. Semantic Web components are used for Rule-based system implementation. This is still an active research theme with recommended, accepted and upcoming standards; SWRL, SPARQL Inferencing Notation (SPIN) (W3C, 2011d). Rule Interchange Format (RIF) (W3C, 2013) and a new standard for schema validation Shapes Constraint Language (SHACL). (W3C, 2017). SPIN is a W3C recommendation to define classes with SPARQL queries to formalize rules and constraints for related classes (W3C, 2011d).

Reasoning capability and re-usability of Semantic Web components are expected to promote efficient implementation. This part of study investigates the use of Web Ontology Language (OWL) and SWRL to design a data quality management framework for geospatial domain. In literature, researchers have used SWRL for quality management studies and spatial applications (Cherfi et al., 2017; Keßler et al., 2009; Wang et al., 2005; Zhong Hu et al., 2013) . This section gives information about the use of ontologies and SWRL rules for quality assessment. Because of the need for spatial relations tests, implementation and technical components of such custom built-ins for SWRL is described. The framework of rule-based approach with SWRL rules is explained. Following sections describe the devised ontologies which are the main components of the framework.

2.2.3.1. Quality Assessment with SWRL

In geospatial domain, there are many kinds of constraints for spatial quality assessment. Some of them exceeds the expressivity of OWL, cannot be expressed with OWL assertions. “Road’s width cannot be more than eight meters” or “Roads must not cross Buildings” are some examples for that kind of constraints. These constraints need mathematical and spatial functions applied with rules.

Semantic Web Rule Language (SWRL), a W3C submission, is a language for specifying rules to be applied on ontologies. Syntactically, SWRL is an extension to OWL with a new sort of conditional (i.e., if-then statements). SWRL incorporates an existing rule language (RuleML) with OWL (W3C, 2004d). Figure 40 represents a knowledge base with rules.

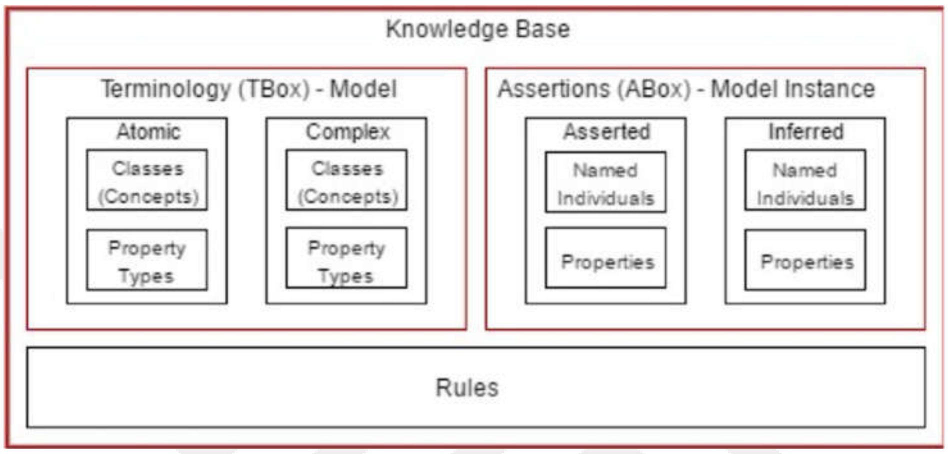


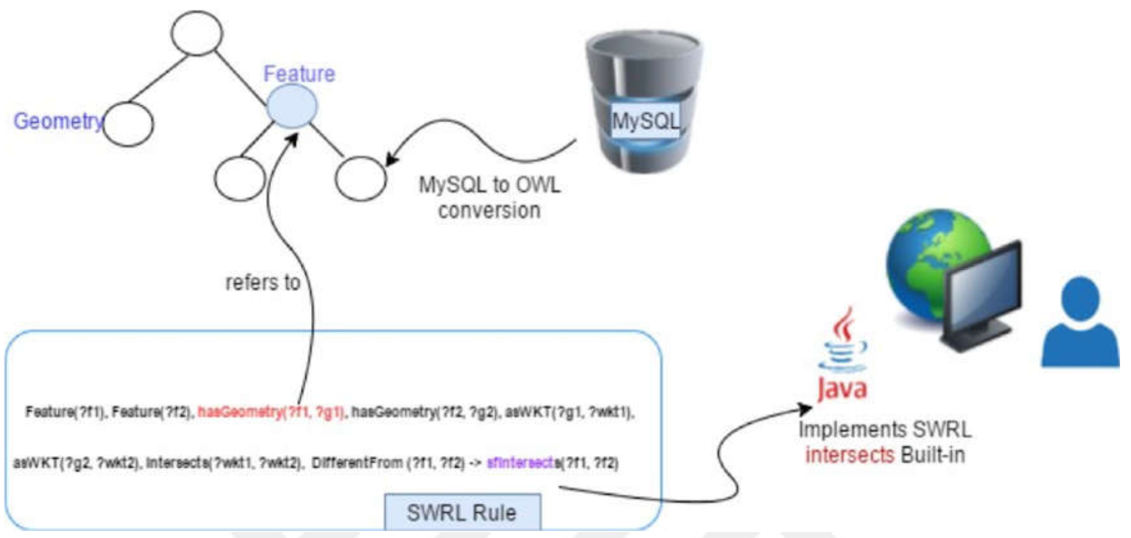
Figure 40. Knowledge base components; ABox, TBox and rules.

SWRL uses built-ins for mathematical operations and comparisons. A spatial rule to infer such as “Roads must not cross Buildings” needs a spatial built-in. SWRL needs to be extended with spatial built-ins. It is possible to create custom SWRL built-ins with the help of Semantic Web tools. Next section explains the implementation of such built-ins with current components.

2.2.3.2. Spatial Built-ins for SWRL and Environment

There is an online open source library for spatial SWRL built-ins (URL-7, 2016). A workflow for one of the built-ins is shown in Figure 41. This is the only study that does so, to the best of the author’s knowledge. In this thesis, similar built-ins for OGC simple relations are created as well, but more importantly one additional built-in is created, for intersection matrices. This increases the efficiency. Once the intersection matrix is found, the relation can be found from the intersection matrix and the dimensions. In the other study, one needs to check the spatial relation for all the relations, which is a time-consuming task.

For applying the SWRL with custom built-ins in Java, a new SWRL built-in registry needs to be created. This is the same with Openllet and Pellet. The custom built-ins are registered with the new built-in registry. The reasoning involves these custom built-ins.



```
Feature(?f1), Feature(?f2), hasGeometry(?f1, ?g1), hasGeometry(?f2, ?g2), asWKT(?g1, ?wkt1),
asWKT(?g2, ?wkt2), Intersects(?wkt1, ?wkt2), DifferentFrom (?f1, ?f2) -> sfIntersects(?f1, ?f2)
```

Figure 41. Workflow of the intersects SWRL built-in.

In the thesis study, the DE-9IM masks for OGC simple feature relations are used with the intersection matrix built-in for finding the spatial relation. The masks are different for other spatial relations, but “T*T***T**” can both be ogc:sfCrosses and ogc:sfOverlaps, depending on the dimensions. For this, a new object property is created encompassing ogc:sfCrosses and ogc:sfOverlaps, which is used in the other rules to identify whether it is a “crosses” relation or “overlaps” relation. “T*T***F**” false is also possible for ogc:sfCrosses . In the case study, the zero-dimension features are not checked.

First issue is to decide which reasoner to be used for implementation. Since, there are many reasoners for OWL but some of them support SWRL rules. Dentler et al. (2011) compares reasoners that can work with large EL Ontologies and considers eight reasoners. It compares them in terms of supported expressivity, soundness, completeness, rule support and a few other features. For this implementation, SWRL rule support is the main criterion. The study compares HermiT, Pellet, and RacerPro, all SWRL supporting reasoners, and concludes that, Pellet is the most expressive one which were. After Pellet became proprietary, an open source fork named Openllet is developed (Rattanasawad et al., 2018).

For OWL ontology manipulation, OWLAPI is the primary choice (Bechhofer et al., 2003; Horridge and Bechhofer, 2011). The alternative, Jena is aimed towards RDF (McBride, 2001). They have bindings for Pellet (Parsia and Sirin, 2004). As the main components (Pellet and OWLAPI) of implementation are in Java, for spatial and spatial data support open source Java libraries are needed. Java 8 is the version of Java implementation with its Stream capability and lambda expressions (Schildt, 2014). The libraries used have not yet completely ported to later Java versions such as Java 10.

Java Topology Suite, initially by Vivid Solutions, currently by LocationTech is used for efficient spatial calculations (URL-5, 2015). LocationTech is a working group of Eclipse foundation whose flagship product Eclipse (Luna, Mars, Neon, then Oxygen version) is the choice of Java integrated development environment. The primary alternative is IntelliJ Idea by JetBrains (Saunders et al., 2006; Yang and Jiang, 2007). Both of them are close choices. Eclipse was preferred slightly because of its set of plugins and familiarity. The interfaces are done with JavaFX graphics (Dea, 2012).

2.2.4. Ontology Based Quality Framework

In this section, a framework for assessing spatial data quality, with the goal of being interoperable, reusable, extensible, customizable and domain-independent is presented and the implementation of the framework is explained.

2.2.4.1. Framework Essentials

- Types of data quality problems to detect (define scope)
- Ontology and framework (devise)
- Rules and processes for extracting erroneous data (implement)
- Quality report format and publishing (report)
- Updating the ontology and rule base structure (update)

2.2.4.2. Scope

In the scope of this part of the thesis, the compliance of spatial data with specifications and the relation with Open World reasoning is assessed. Sub-elements of logical consistency, domain consistency and topo-semantic consistency are chosen. According to specifications, rules are designed related with the features and their relations. Attributes of features and value restrictions for attributes are among the main components of a spatial data producing specification (Sandgren, 2009). Data quality elements tested in the study are shown in Table 18. Besides these, commission as part of completeness, is also checked.

Main components of the framework are based on ontologies. Ontology based data conversion software/tools are used in data access part of the framework. RDB2RDF Mapping Language (R2RML) mappings are created and used for data conversion with the help of GeoTriples tool as can be seen in Figure 42. Alternatively, a SPARQL endpoint can be used for accessing data.

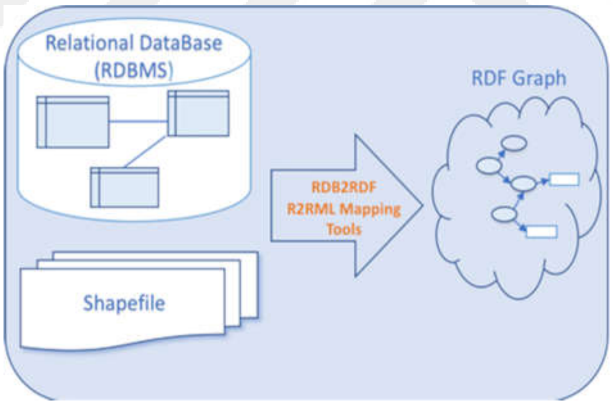
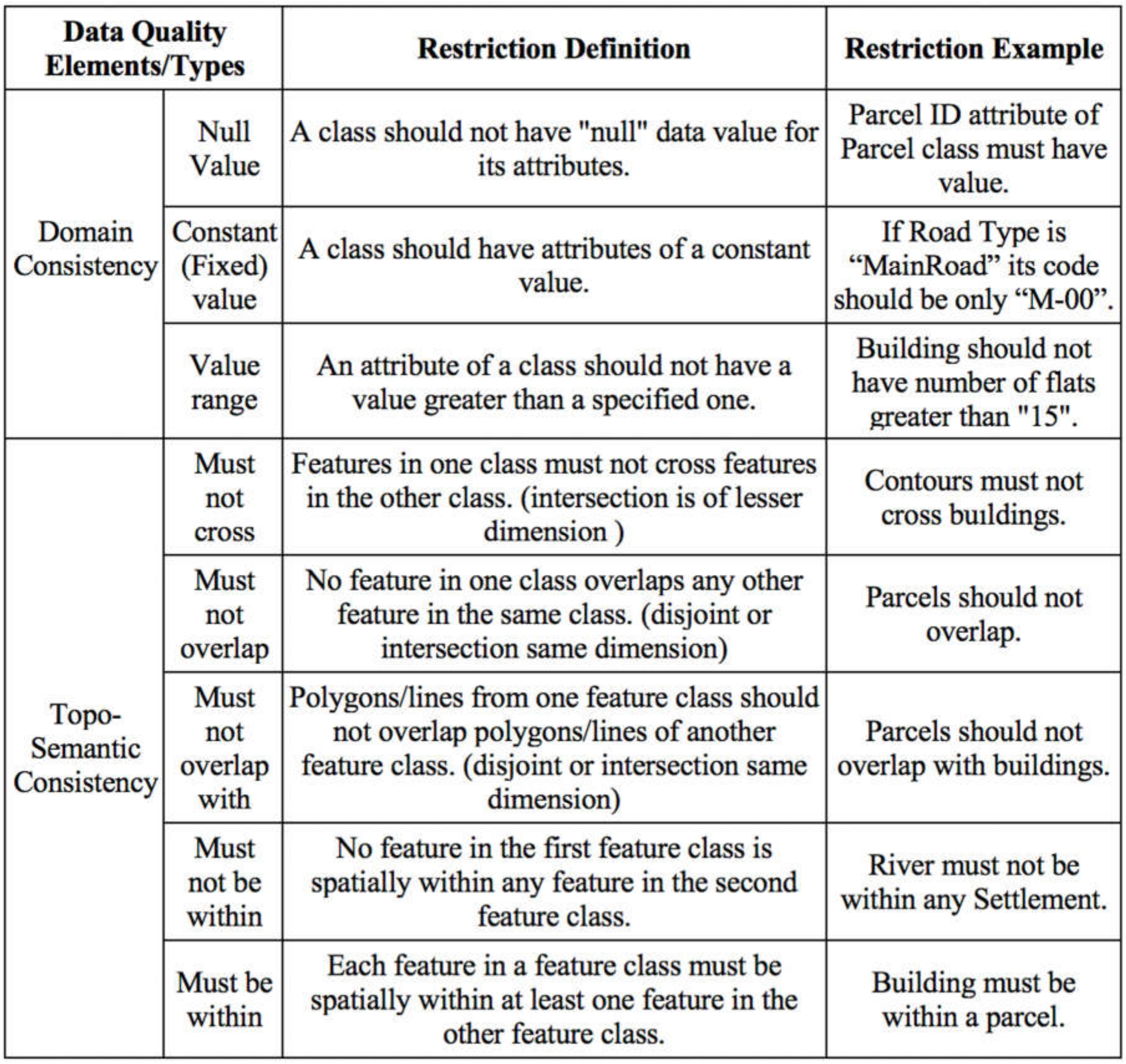


Figure 42. Data conversion from RDB to RDF graph.

Table 18. Logical DQ elements and metrics tested in the study.



Data Quality Elements/Types		Restriction Definition	Restriction Example
Domain Consistency	Null Value	A class should not have "null" data value for its attributes.	Parcel ID attribute of Parcel class must have value.
	Constant (Fixed) value	A class should have attributes of a constant value.	If Road Type is "MainRoad" its code should be only "M-00".
	Value range	An attribute of a class should not have a value greater than a specified one.	Building should not have number of flats greater than "15".
Topo-Semantic Consistency	Must not cross	Features in one class must not cross features in the other class. (intersection is of lesser dimension)	Contours must not cross buildings.
	Must not overlap	No feature in one class overlaps any other feature in the same class. (disjoint or intersection same dimension)	Parcels should not overlap.
	Must not overlap with	Polygons/lines from one feature class should not overlap polygons/lines of another feature class. (disjoint or intersection same dimension)	Parcels should not overlap with buildings.
	Must not be within	No feature in the first feature class is spatially within any feature in the second feature class.	River must not be within any Settlement.
	Must be within	Each feature in a feature class must be spatially within at least one feature in the other feature class.	Building must be within a parcel.

2.2.4.3. Ontology and the Rule Base

For the framework, two types of ontologies are developed; the SDQO and SfOs. SDQO defines concepts related with quality assessment process. The data quality elements are linked to individuals for processes and results. These individuals are used in SWRL rules and SPARQL queries. Classes for features with constraints are defined. SfOs are devised to define which restrictions will be applied to dataset according to specifications by a domain expert. They are mostly hierarchical. SfOs import SDQO and data ontologies. The data ontologies are likely to have some faults. The system should be robust and stay consistent. The system is designed to be robust and user-friendly, therefore usable. The common parts of the SfOs are moved to SDQO. The redundancy is reduced. Most of the rules are in the SDQO. Rules such as associations between SfOs exist within the related SfOs. The use of SDQO enables domain-independent, easy-to-update quality assessment with the SfOs. Especially with an ontology editor, it is expected that domain experts can quickly understand how to manipulate and update their SfOs, when necessary. The system is designed for several SfOs in mind. Interoperability, especially with the inter-SfO associations, follow. SfOs typically do not use the whole capability of the SDQO. When the rulesets change, even without needing to change the SDQO significantly, it is expected that the new SfO can be implemented. Thanks to the prevailing OWA, SDQO is easy-to-update, when necessary. With OWA, truth of statements does not change, only the unknowns become more and more known. Relations between the devised ontologies (SDQO, SfO), data ontologies and GeoSPARQL ontology for proposed framework is shown in Figure 43.

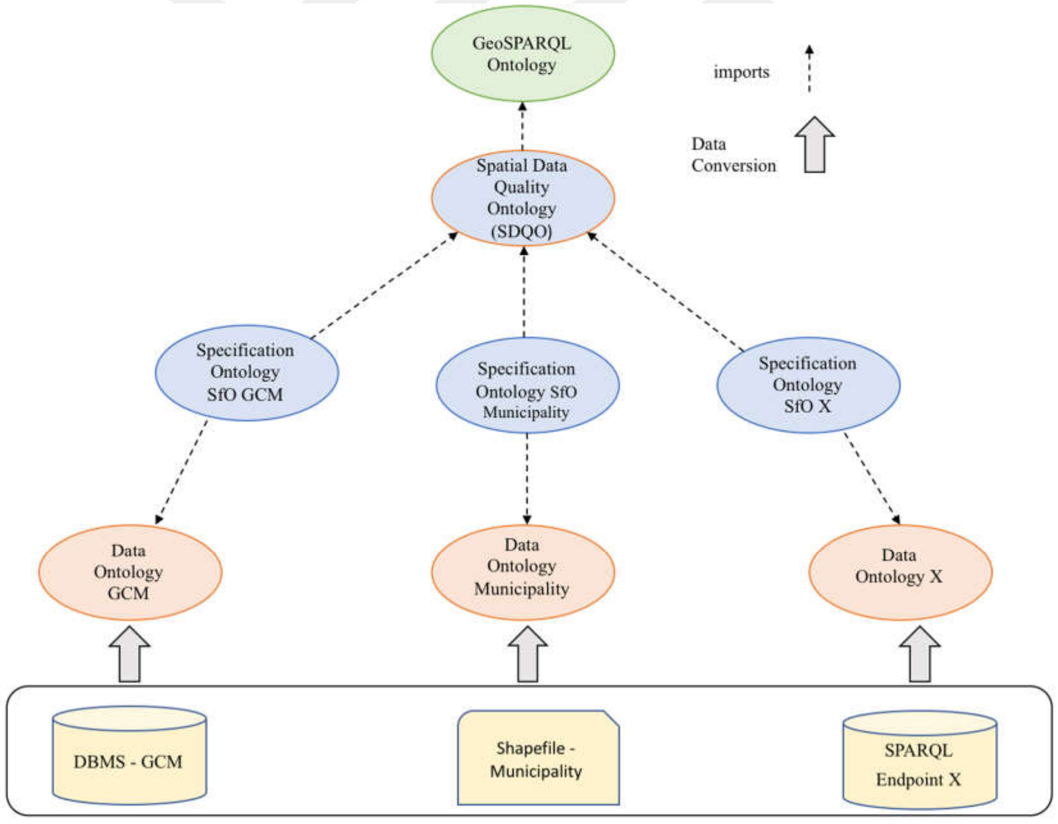

GeoSPARQL Ontology
imports
Data Conversion
Spatial Data Quality Ontology (SDQO)
Specification Ontology SfO GCM
Specification Ontology SfO Municipality
Specification Ontology SfO X
Data Ontology GCM
Data Ontology Municipality
Data Ontology X
DBMS - GCM
Shapefile - Municipality
SPARQL Endpoint X

Figure 43. Ontologies and their relations in the proposed framework.

2.2.4.4. Assessment and Inference

Once the related SfO is chosen, and input dataset is accessed as ontology documents, quality of a dataset is assessed according to the specified restrictions in the SfO. The Java implementation interface, that is also accessible using web services, follows through the process as instructed using SfO. Having imported the SDQO, Openllet, free fork of Pellet reasoner is used for reasoning which produces the result ontology. All the erroneous features are classified according to the sources of error. A dump of the result ontology is produced. A SPARQL endpoint is connected. Also, the results can be visualized.

If the SfO has an association with another SfO, knowledge from that SfO can be used. For instance, for two features, under the change of the scale, relationships such as being disjoint should not change. For instance, a new SfO can be created that uses an old SfO as a reference.

2.2.4.5. Framework and Implementation

Mostly used ontology editors (e.g. Protégé), libraries (e.g. OWL API, Jena) and software are based on Java. Hence, the run-time system is implemented with Java based components. System is responsible with the tasks such as spatial tests, inference of new relations for intended use, creating the quality report. Figure 44 represents the components of the proposed framework. Knowledge Platform part has the tools related with data quality assessment and rule definition. Tool for conversion of data, the result is the data ontology.

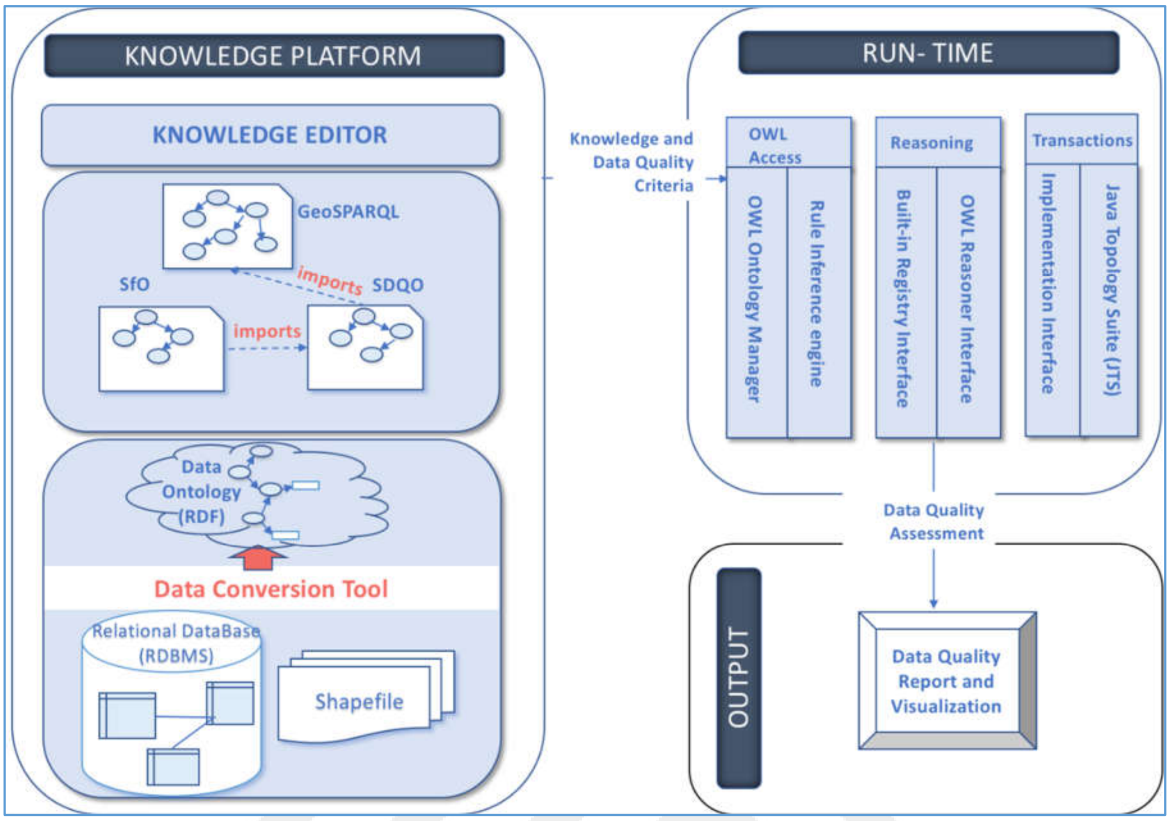


Figure 44. Data Quality Assessment Framework, components and interactions.

2.2.4.5.1. Data Conversion Component

Currently large amounts of spatial data are kept in relational databases. Since the proposed framework is mainly based on ontologies, ontology-based data access software or data conversion is needed for quality assessment. There exist several methods for data conversion. GeoTriples is used to convert relational data and shapefiles with the help of R2RML mappings. Tools such as Ontop-spatial directly access relational database. Other alternatives are TripleGeo and Data Master which is a plug-in for old versions of Protégé.

Data conversion tools support different formats and environment for input and output, such as ESRI shapefiles and relational databases for input and RDF or OWL files for output.

If the tool does not support the format of the input data, libraries such as ogr2ogr[14] can be used for necessary conversion.

TripleGeo, is used for integrating features from geospatial databases into RDF triples supporting GeoSPARQL standard. Besides, it can extract geometries into other vocabularies as well as Virtuoso's own vocabulary (URL-8, 1992) and W3C Basic Geo Vocabulary (Patroumpas et al., 2014). TripleGeo uses configuration file created by the user for translation process. Configuration file is used to declare the input data source and types, geometries to be used, serialization type of the translated data and creates an RDF file as output. In Figure 45, the process flow of TripleGeo and its components are represented. It supports, output serializations through Jena-API[15]. These are, RDF/XML (default), RDF/XML-ABBREV, N-TRIPLES, N3, and TURTLE (TTL). TripleGeo is based on URL-9 (2015) and has limited support for different vocabularies, this affects translation of attribute data. Users should modify TripleGeo for supporting data conversion to different vocabularies.

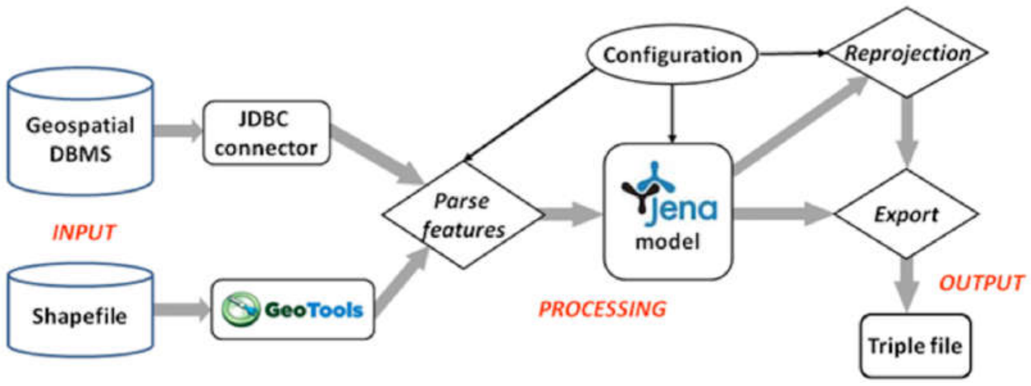


Figure 45. Processing flow diagram for extract-transform-load (ETL) utility TripleGeo (Patroumpas et al., 2014).

GeoTriples transforms spatial data from relational databases, shapefiles and KML into RDF graphs (Kyzirakos et al., 2014). It uses R2RML for data conversion. R2RML is a W3C recommendation for creating customized mappings from relational databases to RDF graphs

[14] http://www.gdal.org/ogr2ogr.html
[15] https://jena.apache.org/

for data translation purpose (W3C, 2012b). With the help of R2RML, attribute transferring becomes more convenient. GeoTriples also generates R2RML files from the schema of the database. These can be further edited. GeoTriples processes these mappings to produce RDF graph (Kyzirakos et al., 2014). The architecture of GeoTriples is represented in Figure 46.

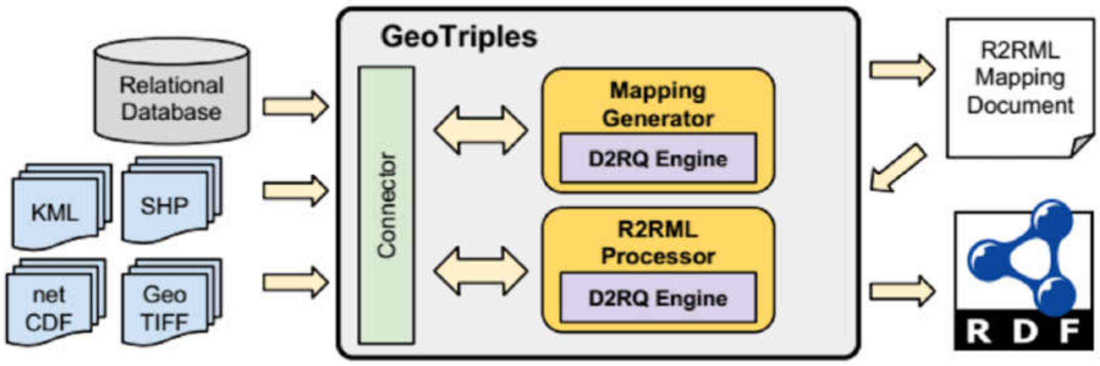
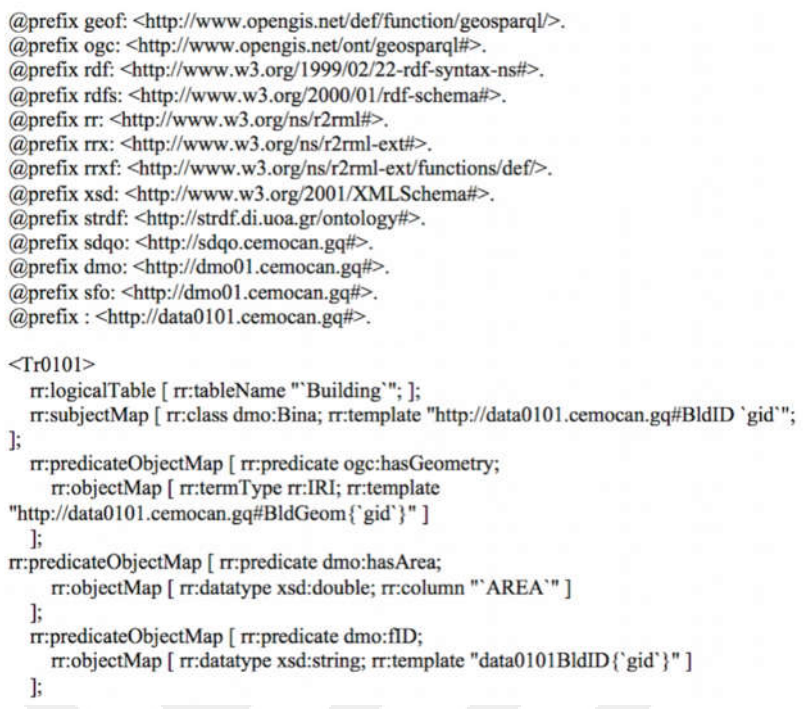


Figure 46. GeoTriples architecture (Kyzirakos et al., 2014)..

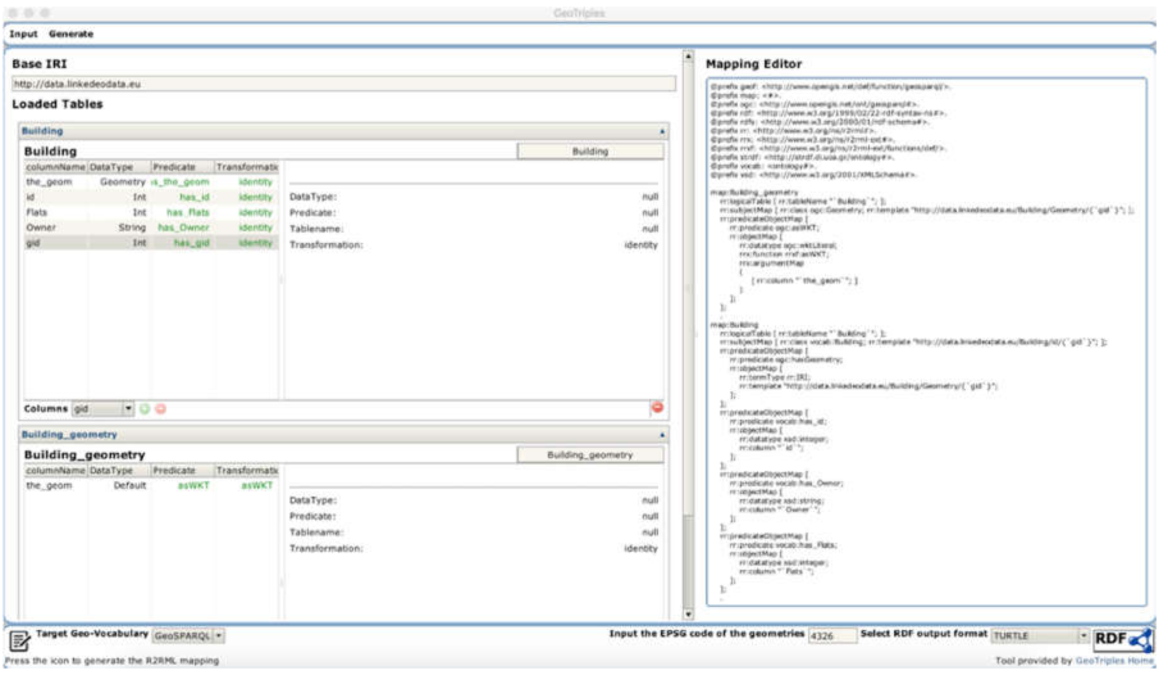
R2RML uses RDF triples for defining the mappings. A sample R2RML Turtle file is shown in Figure 47. The mappings such as <Tr0101> are the subject of these triples. The predicates can be, among others, rr:logicalTable, rr:subjectMap, rr:objectMap, rr:predicateMap and rr:predicateObjectMap. One uses rr:logicalTable for identifying the database/shapefile table. Each row of the table is processed according to the remaining R2RML triples generating RDF triples for the output file. rr:subjectMap, rr:objectMap and rr:predicateMap are for the output file RDF triple subject, object, predicate, respectively. rr:predicateObjectMap is for simultaneous processing of the predicate and the object.



```
@prefix geof: <http://www.opengis.net/def/function/geosparql/>.
@prefix ogc: <http://www.opengis.net/ont/geosparql#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix rrx: <http://www.w3.org/ns/r2rml-ext#>.
@prefix rrxf: <http://www.w3.org/ns/r2rml-ext/functions/def/>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix strdf: <http://strdf.di.uoa.gr/ontology#>.
@prefix sdqo: <http://sdqo.cemocan.gq#>.
@prefix dmo: <http://dmo01.cemocan.gq#>.
@prefix sfo: <http://dmo01.cemocan.gq#>.
@prefix : <http://data0101.cemocan.gq#>.

<Tr0101>
   rr:logicalTable [ rr:tableName "`Building`"; ];
   rr:subjectMap [ rr:class dmo:Bina; rr:template "http://data0101.cemocan.gq#BldID `gid`";
];
   rr:predicateObjectMap [ rr:predicate ogc:hasGeometry;
      rr:objectMap [ rr:termType rr:IRI; rr:template
"http://data0101.cemocan.gq#BldGeom{`gid`}" ]
   ];
rr:predicateObjectMap [ rr:predicate dmo:hasArea;
      rr:objectMap [ rr:datatype xsd:double; rr:column "`AREA`" ]
   ];
   rr:predicateObjectMap [ rr:predicate dmo:fID;
      rr:objectMap [ rr:datatype xsd:string; rr:template "data0101BldID{`gid`}" ]
   ];
```

Figure 47. Mappings created with the settings for Building table.

In Figure 48, GeoTriples GUI is shown which is used in this thesis for data conversion.



Base IRI

http://data.linkedeodata.eu

Loaded Tables

Building

columnName	DataType	Predicate	Transformatic
the_geom	Geometry	is_the_geom	identity
id	Int	has_id	identity
Flats	Int	has_flats	identity
Owner	String	has_Owner	identity
gid	Int	has_gid	identity

DataType: null
Predicate: null
Tablename: null
Transformation: identity

Columns gid

Building_geometry

columnName	DataType	Predicate	Transformatic
the_geom	Default	asWKT	asWKT

DataType: null
Predicate: null
Tablename: null
Transformation: identity

```
@prefix geof: <http://www.opengis.net/def/function/geosparql/>.
@prefix map: <#>.
@prefix ogc: <http://www.opengis.net/ont/geosparql#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix rrx: <http://www.w3.org/ns/r2rml-ext#>.
@prefix rrxf: <http://www.w3.org/ns/r2rml-ext/functions/def/>.
@prefix strdf: <http://strdf.di.uoa.gr/ontology#>.
@prefix vocab: <ontology#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.

map:Building_geometry
   rr:logicalTable [ rr:tableName "`Building`"; ];
   rr:subjectMap [ rr:class ogc:Geometry; rr:template "http://data.linkedeodata.eu/Building/Geometry/{`gid`}"; ];
   rr:predicateObjectMap [
      rr:predicate ogc:asWKT;
      rr:objectMap [
         rr:datatype ogc:wktLiteral;
         rrx:function rrxf:asWKT;
         rrx:argumentMap
         (
            [ rr:column "`the_geom`"; ]
         )
      ];
   ];
.
map:Building
   rr:logicalTable [ rr:tableName "`Building`"; ];
   rr:subjectMap [ rr:class vocab:Building; rr:template "http://data.linkedeodata.eu/Building/id/{`gid`}"; ];
   rr:predicateObjectMap [
      rr:predicate ogc:hasGeometry;
      rr:objectMap [
         rr:termType rr:IRI;
         rr:template "http://data.linkedeodata.eu/Building/Geometry/{`gid`}";
      ];
   ];
   rr:predicateObjectMap [
      rr:predicate vocab:has_id;
      rr:objectMap [
         rr:datatype xsd:integer;
         rr:column "`id`";
      ];
   ];
   rr:predicateObjectMap [
      rr:predicate vocab:has_Owner;
      rr:objectMap [
         rr:datatype xsd:string;
         rr:column "`Owner`";
      ];
   ];
   rr:predicateObjectMap [
      rr:predicate vocab:has_Flats;
      rr:objectMap [
         rr:datatype xsd:integer;
         rr:column "`Flats`";
      ];
   ];
.
```

Target Geo-Vocabulary: GeoSPARQL
Press the icon to generate the R2RML mapping
Input the EPSG code of the geometries: 4326
Select RDF output format: TURTLE

Figure 48. GeoTriples GUI.

Ontop-spatial creates virtual RDF graphs for database access. These graphs are read-only, they cannot bound or updated (Bereta et al. 2016). Recent Protégé editions have Ontop-spatial plugin. Relational spatial databases can be queried directly from Protégé with this plugin. It can be used with OWLAPI. The architecture of Ontop-spatial is shown in Figure 49.

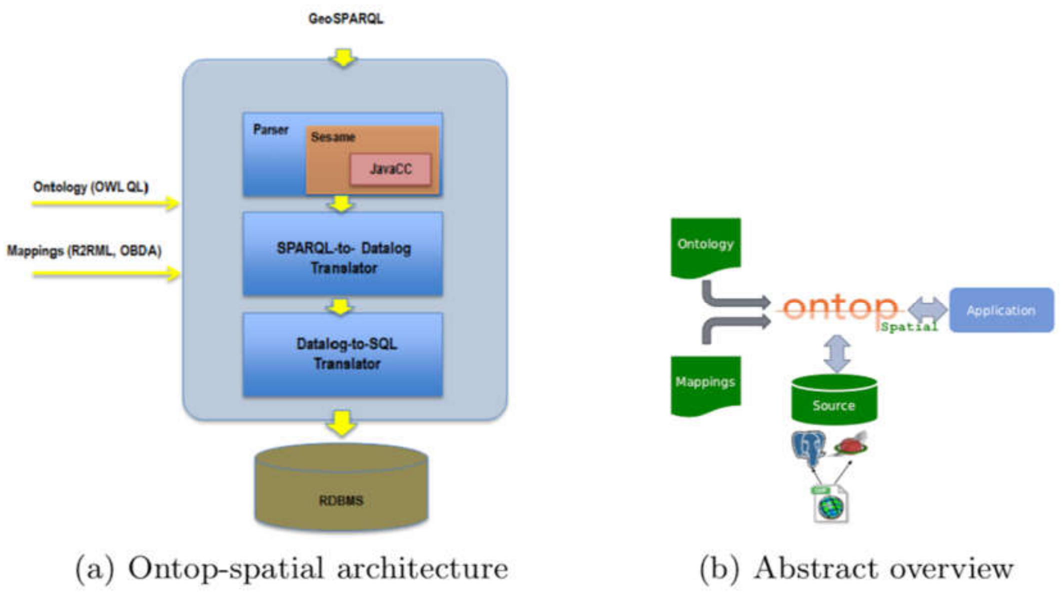


Figure 49. (a) Ontop-spatial architecture (b) Ontop-spatial abstract (Bereta et al., 2016).

DataMaster is a Protégé plug-in that supports the importing of schema and data from relational databases (Nyulas et al., 2007). It can be used with Protégé 3.4.2 version released in 2009. Geometry serialization is created with “asWKT” predicate having WKT Literal values for each type of feature. DataMaster is not supported by the new versions of Protégé.

There are other conversion tools such as D2RQ[16], DataGenie and morph-RDB[17].

[16] http://d2rq.org/
[17] http://mayor2.dia.fi.upm.es/oeg-upm/index.php/en/technologies/315-morph-rdb/index.html

2.2.4.5.2. Ontology Component

Initial step to create ontology for the quality management is defining the concepts in that domain and research for if there is any ontology fit for the intended usage. For this purpose, studies which implemented with ontologies related to this domain are researched. (Fürber and Hepp 2011; Debattista et. al., 2016; Degbelo 2012; Geisler et al. 2016) are the most related studies. All of the mentioned studies are applied with Semantic Web components including OWL.

Fürber and Hepp (2011) introduce Data Quality Management (dqm) ontology. Basic purposes of this ontology are; representation of data quality requirements in a machine-readable way and annotation of quality-relevant meta-information to data elements. Some quality dimensions including, e.g. unique value test, legal value test. This ontology classifies the rules for evaluating data quality metrics takes the user requirements. User requirements are defined for quality evaluation and these are classified as rules in dqm ontology. The main class dqm: DataRequirement is designed as the superclass of all other requirements. This ontology has concepts about data quality such as, QualityElement and QualityScore. It does not proper for the ontologies in the thesis study. For example, it is required to conceptualize classes to represent spatial relations, and rules to define the inconsistent data as a result of spatial quality dimensions such as; Domain Consistency, Topo-Semantic Consistency.

Debattista et al. (2016) developed daQ (Dataset Quality Ontology) for data quality purposes especially for linked data, shown in Figure 50. They created a linked data quality framework called Luzzu. One of the aims of the daQ study is to make quality results available for datasets for making comparisons among existing datasets. It provides ontology-driven back-end for representing quality metadata. Main classes of daQ ontology are, daQ: Category, daQ: Dimension and daQ: Metric. These are abstract classes created to represent specific data quality aspects for assessment. This ontology is modelled as an architecture design pattern for a general purpose; modelling dataset quality results with a common model.

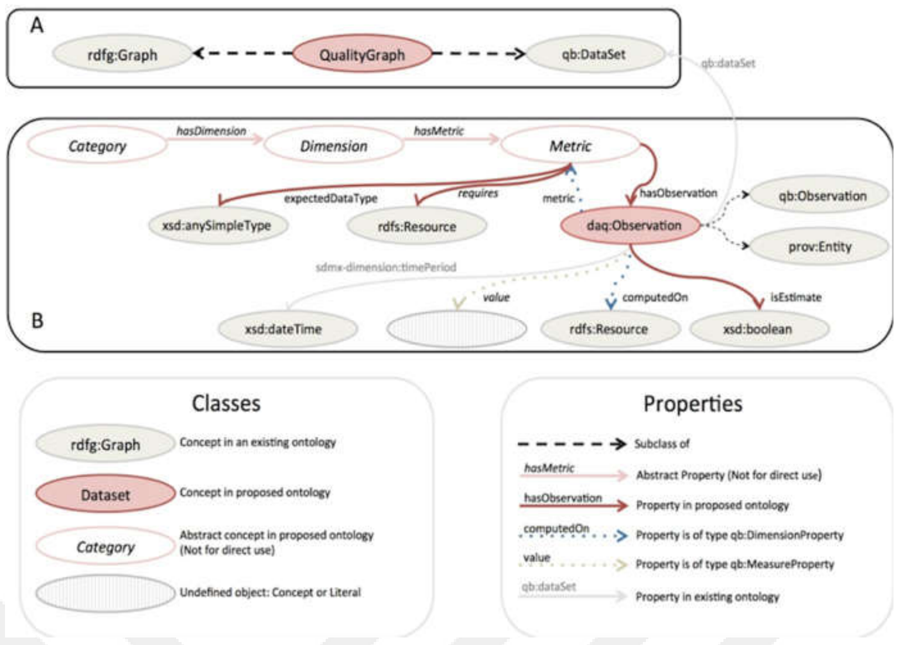


Figure 50. The Dataset Quality Ontology (daQ) created for Luzzu (Debattista et al. 2016).

Degbelo (2012) create an ontology design pattern (ODP) to model quality aspects of Semantic Sensor Network in Figure 51. ODP proposed in this study is built upon ontologies for sensors and observations such as Stimulus Sensor Observation Ontology. For quality component ‘resolution’, is chosen for sample implementation. ODP implementation for quality assessment with SWRL rules made it possible to find the value of the spatial data quality component for any sensor observation and the quality criterion used for assessing the sensor observation’s quality with inference. There is a similar ontology design in this thesis with Degbelo (2012). This ontology has a specific model to solve a recurrent problem in the Semantic Sensor area. The rules in the ontology is not proper for the requirements of the thesis study. The design pattern is shown in Figure 51.

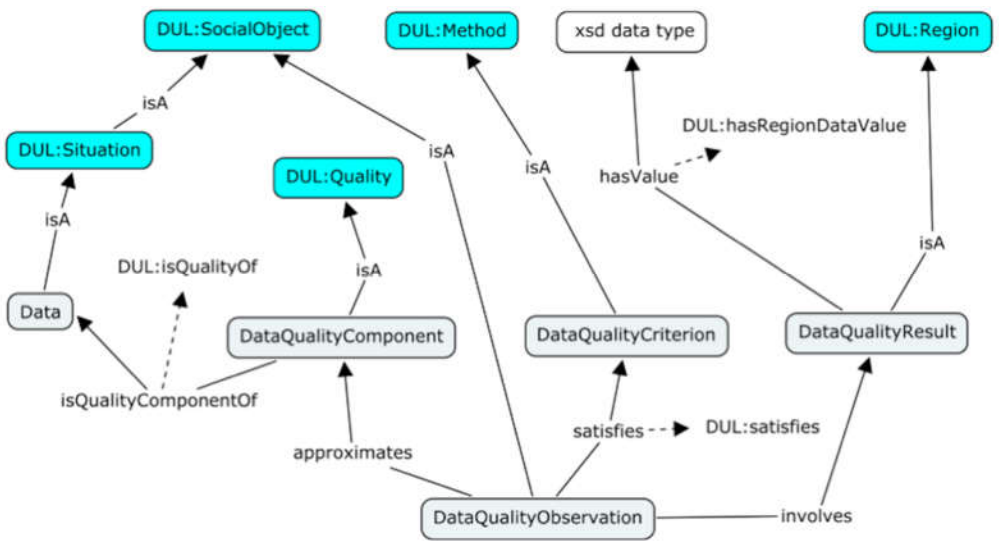


Figure 51. Ontology Design Pattern for Semantic Sensor Web (Degbelo, 2012).

Geisler et al. (2016) propose an ontology-based approach to test data streams. Their ontology defines information about quality assessment which is basically composed of data quality metrics, dimensions, functions and their relations. Three types of quality metrics are defined according to the requirements of application. These are content-based metrics, query-based metrics and application-based metrics. See Figure 52 for visualization.

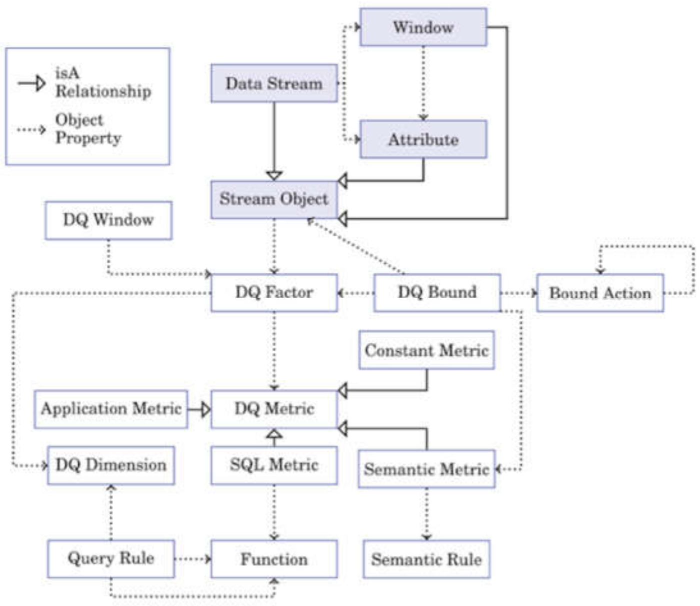


Figure 52. Visualization of DQ Ontology (Geisler et al. 2016).

Ontology component of the system comprises two types of ontologies. The SDQO ontology, contains the necessary rules and the concepts related with the quality assessment. The SfOs are simple ontologies, developed keeping in mind the reusability of rules with different kinds of datasets. There can be one or more SfOs (eg. for different scales) for each institution such as Turkish General Command of Mapping (GCM) and municipalities.

2.2.4.5.2.1. SDQO

While creating the SDQO, terminologies which are used in above are considered. Although, all ontologies are created with the same intention, quality management, they are specific to different application domains. They have some common concepts such as “Quality Dimension”, “Quality Metric” or “Quality Result”. None of the mentioned ontologies are conform to the study purpose. Therefore, a new ontology is needed.

For SDQO, the ontologies for geospatial domain is researched. Mostly used ontologies for geospatial domain are; GeoNames, W3C Geo, GeoOWL and OGC GeoSPARQL. The GeoNames Ontology and W3C Geo support only point type geometries (URL-10, 2012; URL-11, 2007). GeoOWL, the updated model of W3C Geo, is created compatible with GeoRSS Feature Model (URL-11, 2007). Only GeoSPARQL supports other geometry types such as polygon and lines and basic spatial relations. The GeoSPARQL ontology has been selected for the implementation.

In GeoSPARQL, any feature or geometry is a spatial object. In ontological terms, ogc:SpatialObject has two subclasses, ogc:Feature and ogc:Geometry. These the two subclasses are disjoint from each other (owl:disjoint), and they are connected with the object property ogc:hasGeometry. The domain of this object property is ogc:Feature and the range is ogc:Geometry.

Any feature can be associated with a geometry using ogc:hasGeometry. The owl:disjoint relation implies that any element in one class is different from any element in the other class (owl:differentFrom). The owl:disjoint relation is inherited by subclasses. As features are OWL individuals of type ogc:Feature or a subclass (of any level), and geometries are OWL individuals of type ogc:Geometry or a subclass (of any level), no feature can be a same as (owl:sameAs) a geometry and vice versa. ogc:asWKT datatype property associates a geometry to a WKT literal ogc:WKTLiteral, such as “POLYGON ((0 0, 0 1, 1 1, 1 0, 0

0))"^^ogc:WKTLiteral. GeoSPARQL supports GML as well. GeoSPARQL has other datatype and object properties, as well. Most of the object properties are spatial relationships.

According to the set of rules depending on appropriate and chosen elements for data quality, SDQO is also responsible of processing and integrating data quality elements with associated procedures and implementing the procedures in accordance with the geospatial data and the quality elements. SDQO prepares the resulting spatial data quality ontology which relates data quality results with tested data and prepares it for queries or publishing.

In SDQO, there are three top classes directly below owl:Thing , one for data, one for data quality elements, and one for data quality results and processes. These are ogc:SpatialObject, sdqo:DataQualityElement, sdqo:DataQualityProcessResult, respectively Figure 53.

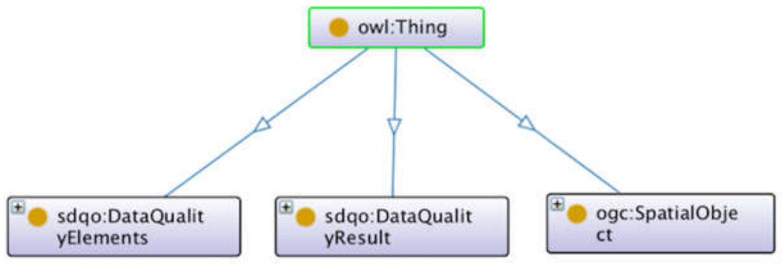


Figure 53. SDQO ontology, top classes.

ISO 19157 spatial data quality standard classifies the basic quality elements and their assessment process. "Data Quality Element" is defined as "a component describing a certain aspect of the quality of geographic data" (ISO 2013). As defined in the ISO 19157, in the SDQO, "Data Quality Element" is used as abstract class to represent the data quality elements. "Data Quality Result" is defined in the ISO 19157 as the report of assessed data with respect to "Data Quality Element". In the ontology SDQO, no additional subclasses of ogc:Geometry are created; all subclasses are from indirectly imported ontologies such as OGC simple features ontology.

In SDQO, ogc:Feature has three direct subclasses, other than the ones from imported ontologies. These are sdqo:GeomClassifiedFeature, sdqo:FixedRefFeature and sdqo:RestrictedFeature. There is no owl:disjoint relation declared between them. These classes are represented in Figure 54.

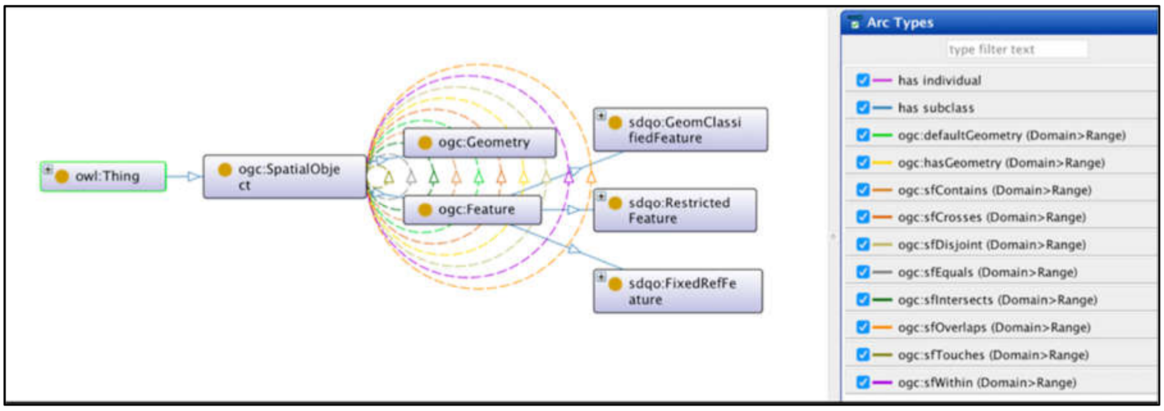

owl:Thing
ogc:SpatialObject
ogc:Geometry
ogc:Feature
sdqo:GeomClassifiedFeature
sdqo:RestrictedFeature
sdqo:FixedRefFeature
Arc Types
type filter text
has individual
has subclass
ogc:defaultGeometry (Domain>Range)
ogc:hasGeometry (Domain>Range)
ogc:sfContains (Domain>Range)
ogc:sfCrosses (Domain>Range)
ogc:sfDisjoint (Domain>Range)
ogc:sfEquals (Domain>Range)
ogc:sfIntersects (Domain>Range)
ogc:sfOverlaps (Domain>Range)
ogc:sfTouches (Domain>Range)
ogc:sfWithin (Domain>Range)

Figure 54. OGC Feature Class with subclasses.

sdqo:GeomClassifiedFeature has three subclasses, sdqo:CalcPoly, sdqo:CalcLine and sdqo:CalcPoint, declared to be owl:disjoint. Any feature that is associated with a geometry that has a valid ogc:asWKT value, is automatically put in the correct one according to the ogc:asWKT value, using a SWRL rule. This information is to be used in other rules. sdqo:GeomClassifiedFeature has label “Features classified according to their geometries”.

sdqo:FixedRefFeature has OWL named individuals to be used in rules as reference markings. For instance, attribute tests can make use of reference individuals.

sdqo:RestrictedFeature has four direct subclasses, sdqo:InterObjectPrRF, sdqo:IntraDataPrRF, sdqo:IntraObjectPrRF and sdqo:InterDataPrRF. Rules and given subclass relations determine which features these classes have. The labels and the descriptions are listed in Table 19. The subclass relations are represented in Figure 55. “Intra” classes are for restrictions within a single class and “Inter” classes are for restrictions in class pairs. sdqo:RestrictedFeature has the label “Features according to the property restrictions”.

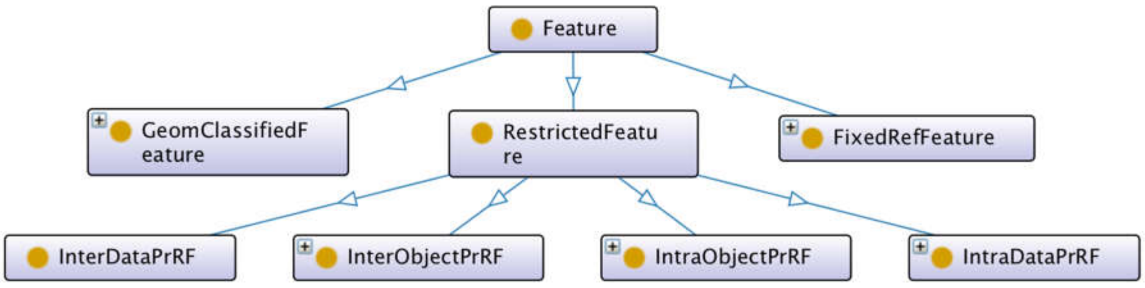


Figure 55. RestrictedFeature class and its subclasses.

Table 19. Labels for the direct subclasses of sdqo: RestrictedFeature

Class	**rdfs:label**
sdqo:InterObjectPrRF	Feature restricted wrt relations between classes
sdqo:InterDataPrRF	Feature restricted by attributes between classes
sdqo:IntraDataPrRF	Feature restricted by attributes within class
sdqo:IntraObjectPrRF	Feature restricted wrt relations within class

sdqo:InterObjectPrRF has classes for spatial object properties, and also general object properties.

One of the subclass pairs of sdqo:InterObjectPrRF is pair the classes sdqo:ClassOverlap1 and sdqo:ClassOverlap2. These classes have labels "Must Not Overlap With members of second class" and "Must Not Overlap With members of first class", respectively.

SfOs importing SDQO can establish further subclass relations with this pair. In the case study, one of the SfOs, the one devised from the specifications of the Turkish General Command of Mapping, contains classes such as sfo:Contour, sfo:Road, sfo:Building, sfo:Parcel, sfo:Lake, etc. If sfo:Contour is a subclass of sfo:Class0v1_01 and the others are subclasses of sfo:ClassOv2_01, the associated rules infer features of type sfo:Contour overlapping with features of type any of the other classes above. Furthermore, these rules infer more descriptive results about these erroneous occurrences. This corresponds to the specification that contours cannot intersect with roads, buildings, parcels and lakes.

sdqo:IntraDataPrRF is the class used for establishing single class constraint using datatype properties.

The sdqo:ClassSameDataProp subclass finds relevant features from the imported data ontology that fail to have the same value for a particular data property with the other relevant features. For instance, one can find out if all the relevant buildings have the same residential type. One can also find out if any of the shores fail to have same elevation with the seas. This is done by subclass relations as above and by equivalency relations with the related datatype property. The datatype property used in an associated rule is set equivalent to a datatype property such as sfo:elevation or sfo: residentialType. sdqo:ClassDistinctDataProp is for finding features that have same data property when they are forbidden to do so.

In SDQO, the direct subclasses of the classes sdqo:ClassSameWithFixed each contain an individual that is also of type sdqo:FixedRefFeature. There are associated rules in SDQO, for each direct subclass. These rules make use datatype properties within SDQO. During application, the datatype property used in an associated rule is set equivalent to a datatype property in the data ontology, also the fixed reference individual is set to be equivalent to a feature in the data ontology (alternatively, a value can be given). For instance, one can find out if all the relevant buildings have five floors, one can also find out whether any relevant building fails to have less than seven floors.

sdqo:IntraObjectPrRF is the class used for establishing single class restrictions using object properties. “Parcels should not overlap” rule can be given as an example to such restrictions.

sdqo:Completeness class has subclasses sdqo:Commision and sdqo:Ommission. Commissions can be done with SDQO using rules. This also gives quantitative info about omissions.

sdqo:GeometricAccuracy has sdqo:GeometryValidity as a subclass.

sdqo:TopoSemanticConsistency is a subclass to sdqo:SemanticAccuracy and sdqo:LogicalConsistency. sdqo:LogicalConsistency also has sdqo:DomainConsistency.

sdqo:DataQualityResult is defined as abstract class to classify the results of data quality assessment.

Class hierarchy of SDQO ontology is shown in Figure 56.

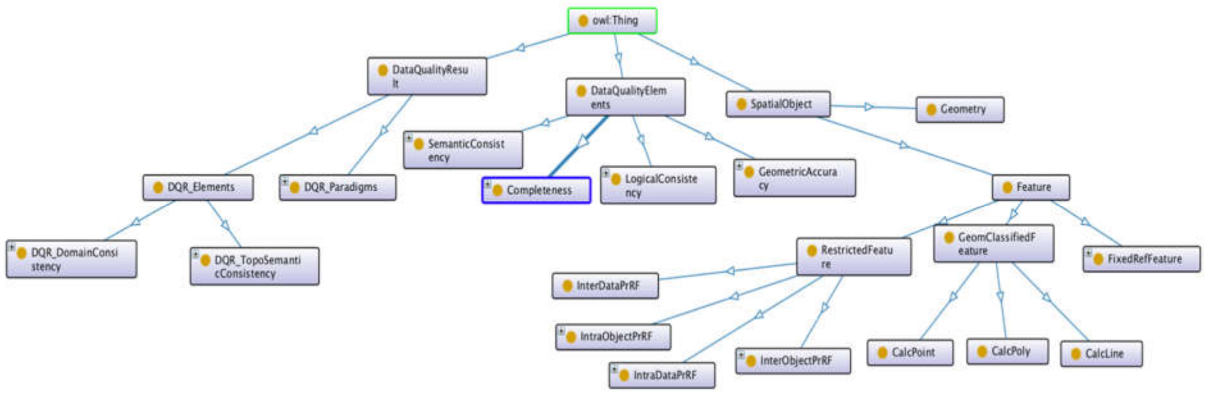


Figure 56. SDQO ontology.

SDQO has the following own datatype properties. Several of them are to be used as a super property to data properties in SfOs. Camel case naming is used as it is usually done with ontology properties. Other datatype properties are schema:startTime and OGC datatype properties such as ogc:asWKT.

sdqo:errorCode datatype property is defined to give a code to results to identify the problem of data.

sdqo:hasErrorCode datatype property is defined to relate data with the resultant sdqo:errorCode.

sdqo:dataHasErrorWithCode datatype property connects erroneous data to error codes.

Every feature must have sdqo:featureID datatype property which is to be unique to every feature. SfOs should either use sdqo:featureID or define a data property as a subproperty to sdqo:featureID, uniquely identifying features for that SfO. If this is not done, erroneous features will still be found but error messages will fail to be complete.

sdqo:hasMessage datatype property connects results and processes to error messages.

sdqo:elementHasMessage datatype property is a shortcut from elements to error messages (sdqo:hasResult and sdqo:hasMessage).

There are SDQO datatype properties for attribute tests such as sdqo:dataProp01.

sdqo:subnr property is the main data property for assigning the classes to be tested with the subclass number. It has subproperties such as sdqo:subnrOverlap and sdqo:subnrMustWithin. These subproperties are used in defining top classes in SfO.

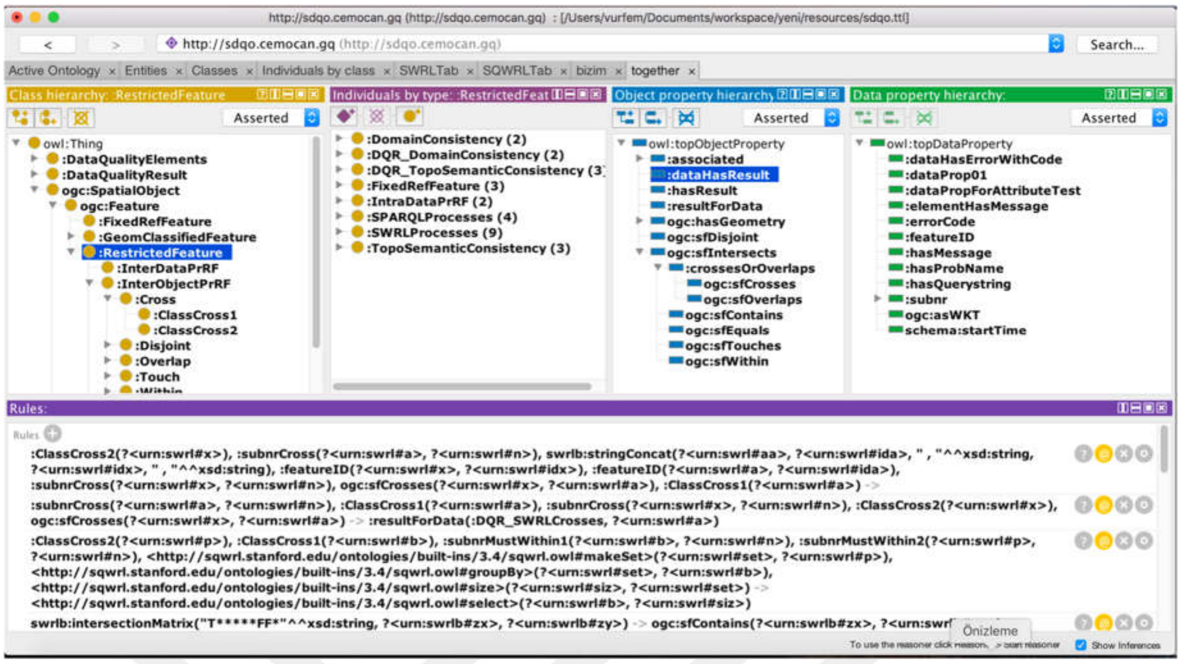
sdqo:hasQueryString datatype property is for SPARQL query strings.

sdqo:associated is to be used as a super property for association object properties between SfOs. These association properties are SfO-specific except two common ones; sdqo:scalewiseAssociated is used for scale-wise association and sdqo:scalewiseAssociatedSameDimension is for scale-wise association with no change in geometry types. Object relations such as disjointness are expected to be preserved. sdqo:crossesOrOverlaps is a super property connecting ogc:sfOverlaps and ogc:sfCrosses. Intersection matrix masking is used for geospatial relations.

While sdqo:resultForData object property create a relationship between quality results and data, sdqo:hasResult relates data quality elements to results. sdqo:dataHasResult is inverse property to sdqo:resultForData.

Furthermore, GeoSPARQL object properties such as ogc:sfOverlaps are used for relations between spatial classes. ogc:sfIntersects is declared to be a super property to intersection type properties such as ogc:sfOverlaps.

SDQO ontology view in Protégé ontology editor is shown in Figure 57.



```
:ClassCross2(?<urn:swrl#x>), :subnrCross(?<urn:swrl#a>, ?<urn:swrl#n>), swrlb:stringConcat(?<urn:swrl#aa>, ?<urn:swrl#ida>, " , "^^xsd:string,
?<urn:swrl#idx>, " , "^^xsd:string), :featureID(?<urn:swrl#x>, ?<urn:swrl#idx>), :featureID(?<urn:swrl#a>, ?<urn:swrl#ida>),
:subnrCross(?<urn:swrl#x>, ?<urn:swrl#n>), ogc:sfCrosses(?<urn:swrl#x>, ?<urn:swrl#a>), :ClassCross1(?<urn:swrl#a>) ->
:subnrCross(?<urn:swrl#a>, ?<urn:swrl#n>), :ClassCross1(?<urn:swrl#a>), :subnrCross(?<urn:swrl#x>, ?<urn:swrl#n>), :ClassCross2(?<urn:swrl#x>),
ogc:sfCrosses(?<urn:swrl#x>, ?<urn:swrl#a>) -> :resultForData(:DQR_SWRLCrosses, ?<urn:swrl#a>)
:ClassCross2(?<urn:swrl#p>), :ClassCross1(?<urn:swrl#b>), :subnrMustWithin1(?<urn:swrl#b>, ?<urn:swrl#n>), :subnrMustWithin2(?<urn:swrl#p>,
?<urn:swrl#n>), <http://sqwrl.stanford.edu/ontologies/built-ins/3.4/sqwrl.owl#makeSet>(?<urn:swrl#set>, ?<urn:swrl#p>),
<http://sqwrl.stanford.edu/ontologies/built-ins/3.4/sqwrl.owl#groupBy>(?<urn:swrl#set>, ?<urn:swrl#b>),
<http://sqwrl.stanford.edu/ontologies/built-ins/3.4/sqwrl.owl#size>(?<urn:swrl#siz>, ?<urn:swrl#set>) ->
<http://sqwrl.stanford.edu/ontologies/built-ins/3.4/sqwrl.owl#select>(?<urn:swrl#b>, ?<urn:swrl#siz>)
swrlb:intersectionMatrix("T*****FF*"^^xsd:string, ?<urn:swrlb#zx>, ?<urn:swrlb#zy>) -> ogc:sfContains(?<urn:swrlb#zx>, ?<urn:swrl
```

Figure 57. SDQO in Protege ontology editor.

2.2.4.5.2.2. SfO

SfOs directly import SDQO and the directly or indirectly import data ontologies associated with that SfO or other SfOs. One can have one data ontology for each database/shapefile table, or they can be combined. The data ontologies are created with R2RML, keeping in mind that they will be imported by its SfO. Even if the individuals are in the data ontologies, the classes have SfO IRIs.

A SfO has the class hierarchy reflecting the associated specifications. Furthermore, these specification hierarchy classes have superclasses, still in SfO. These classes are created to de-anonymize the restrictions. They are the top classes with SfO's IRI. They are subclasses to SDQO classes and establish the connection to SDQO. Furthermore, these top classes should have associated geometries under the appropriate simple features class, such as sf:Polygon or sf:Linestring.

In general, SfOs do not have many rules; SDQO has them. With SfO, more general relations are translated into is-a relations in the scope of SfO, if possible. The feature pairs causing the errors are identified. Below, a sample translation is given.

Specification rule	Implementation
Features in class A must not overlap with features in class B	A is a subclass of C, an SfO class; B is a subclass D, an SfO class; these are subclasses of some SDQO classes that take part in SWRL rules establishing the relevant error properties involving faulty features in A, B and forbidden overlaps relation.

According to the inspected data quality metrics, with the help of SDQO and the related processes implemented within the (reasoning) component, new data properties inferred from the applied quality rules. The resulting inferred data properties are added to the data ontology containing the information about quality error.

SfOs define specification classes and generic classes for data to be tested.

Specification classes are SfO classes such as “sfo:Road” and “sfo:Building”, that exist in the specifications. They can have subclasses; sfo:PermanentLake can be a subclass to sfo:Lake.

SfO generic classes are defined according to the spatial relations related to data. Spatial relations should be defined between feature classes. In specifications, a feature class might have topological relations with; itself, a feature class, more than one feature class.

Examples include sfo:ClassOvf01, sfo:ClassOvf02, etc. and sfo:ClassOvs01, sdqo:ClassOvs02, etc. . Here Ov is for Overlaps, following f is for ‘first’, ‘s’ is for ‘second’. The numbers represent the subclass numbers, the order in pairs of specification classes. .

Following lines define sfo:ClassOvfO1 in Turtle format. Here, one uses OWL restrictions which are anonymous classes

```
sfo:ClassOvf01 rdf:type owl:Class ;
            owl:equivalentClass [ owl:intersectionOf ( sdqo:ClassOverlap1
                                              [ rdf:type owl:Restriction ;
                                                owl:onProperty sdqo:subnrOverlap ;
                                                owl:hasValue 1
                                              ]
                                            ) ;
                         rdf:type owl:Class
                       ] .
```

The class hierarchy above for sfo:PermanentLake is to be tested with mustNotOverlapWith rule is shown in Figure 58. Other SfO classes have similar class hierarchies.

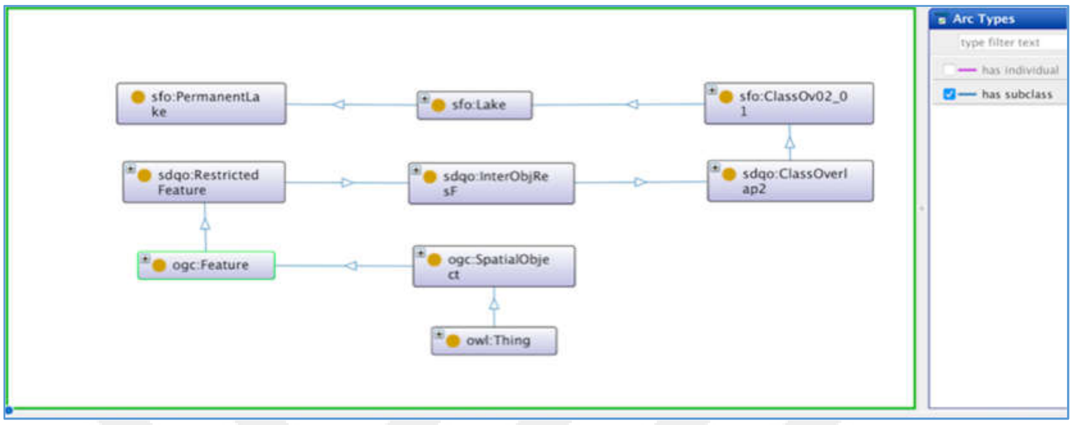


Figure 58. A path of subclass relations in a SfO.

The system is designed to be usable. It is expected that, when necessary, even the domain experts can update their SfOs, especially using ontology editors such as Protégé. This can be done by manipulating class hierarchy. As an example, assume that sfo:DamLake is a new class for illegal "Crosses" (ogc:sfCrosses) relationships with the sfo:Road. This can be done by creating a class sfo:DamLake under the generic class containing the other classes with that relationship. Moving a class out of that generic class invalidates the relationship.

2.2.4.5.2.3. Rules for Quality Assessment

The ontologies are designed for purposes such as usability, reusability, extensibility, robustness. This is reflected in the design of the rules. For instance, it is possible to apply a rule to subclasses of a class with the help of a SWRL supporting reasoner. This helps reducing the redundancy.

Almost all the rules are part of the SDQO. Very few rules are left to the SfOs. SDQO is designed so that it will be imported by some SfOs. As in all other ontologies, SDQO is ignorant of the ontologies importing it. Importing an ontology that imports back makes both ontologies and any ontology in between equivalent (similar to the following inequality implication: $a \leq b \leq a \rightarrow a = b$). This is not to be allowed.

The top classes specific to the SfOs are subclasses to certain classes in SDQO. The data properties sdqo:subnr and sdqo:associated are to be used as super properties. sdqo:subnr and subproperties are used to handle and make use of these subclass relations. The rules in SDQO make use of them.

Some SDQO rules are for management within SDQO, such as to connect properties. Basic ones are shown in Table 20. Some rules also make use custom SWRL built-in swrlg:intersectionMatrix for calculating spatial relations if they do not exist yet. Spatial constraint rules are based on the spatial relations defined for simple features.

Table 20. Rules for relations within SDQO.

No.	SWRL Expression
1	swrlb:stringConcat(?aap, ?p, " , "^^xsd:string, ?aa)^ hasMessage(?r, ?aa)^ hasProbName(?t, ?p)^ hasResult(?t, ?r) -> elementHasMessage(?t, ?aap)
2	errorCode(?r, ?n)^ resultForData(?r, ?a) -> dataHasErrorWithCode(?a, ?n)
3	crossesOrOverlaps(?x, ?y)^ CalcPoly(?x)^ CalcPoly(?y) -> sfOverlaps(?x, ?y)
4	crossesOrOverlaps (?y, ?x)^ CalcLine(?x)^ CalcPoly(?y) -> sfCrosses(?x, ?y)
5	crossesOrOverlaps(?x, ?y)^ CalcLine(?x)^ CalcPoly(?y) -> sfCrosses(?x, ?y)
6	RestrictedFeature(?x) ^ ogc:hasGeometry(?x, ?g) ^ ogc:asWKT(?g, ?w) ^ swrlb:contains(?w, "POLYGON") -> CalcPoly(?x)
7	RestrictedFeature(?x) ^ ogc:hasGeometry(?x, ?g) ^ ogc:asWKT(?g, ?w) ^ swrlb:contains(?w, "LINESTRING") -> CalcLine(?x)
8	RestrictedFeature(?x) ^ ogc:hasGeometry(?x, ?g) ^ ogc:asWKT(?g, ?w) ^ swrlb:contains(?w, "POINT") -> CalcPoint(?x)

Spatial rules are generated according to mostly used constraints in specifications. Following implementation of these rules is demonstrated using a "Must not cross" rule.

"Must not cross" rule constraints the cross relation of first feature with the second feature. "Any Road must not cross with a Building or Sea" rule can be given as example.

For this, in the SfO, several classes exist. sfo:Road is a subclass of sfo:ClassCrf03, sfo:Building and sfo:Sea are subclasses of sfo:ClassCrs03. These specification classes can have subclasses, as well. It is assumed that, the SfO is optimized as explained in the SfO

Creation section below. In particular, sfo:Road is not in sfo:ClassCrf01 or sfo:ClassCrf02. All these ‘first side’ generic classes have only one direct subclass.

sfo:ClassCrf03, therefore its direct or indirect subclasses (including sfo:Road) are subclasses of both sdqo:ClassCross1 and also Restriction for having sdqo:subnrCross equal to 3. Therefore, any feature in sfo:Road satisfies sdqo:ClassCross1(?a) ^ sdqo:subnrCross(?a, ?n). for n=3.

sfo:ClassCrs03, therefore its direct or indirect subclasses (including sfo:Building and sfo:Sea) are subclasses of both sdqo:ClassCross2 and also Restriction for having sdqo:subnrCross equal to 3. Therefore, any feature in sfo:Building or sfo:Sea satisfies sdqo:ClassCross2(?x) ^ sdqo:subnrCross(?x, ?n). for n=3.

Two related rules are listed below. The first rule is for getting roads with unallowed crossing relation. The second rule is for the error message in Comma-Separated Values (CSV) format. Due to Open World reasoning, the order of clauses or rules do not matter. The rules are numbered for convenience. This process is associated with the individual sdqo:DQR_SWRLCrosses. This individual is within classes sdqo:DQR_TopoSemanticConsistency and sdqo:SWRLProcesses which are indirect subclasses of sdqo:DataQualityResult.

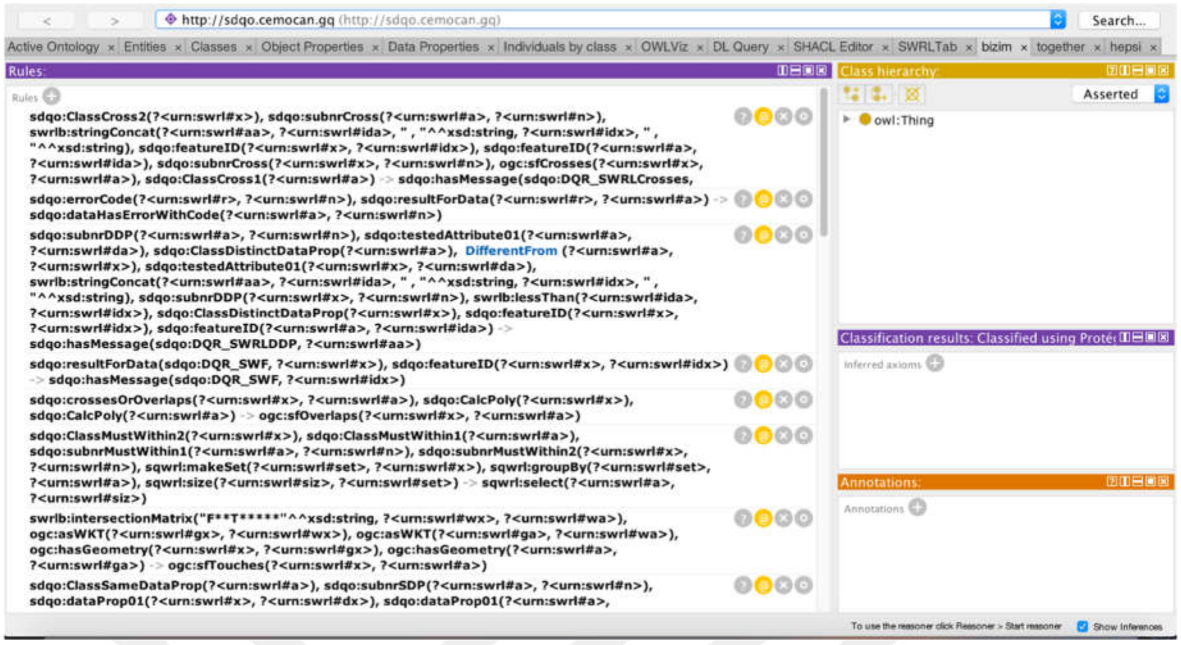
First rule:

subnrCross(?a, ?n) ^ ClassCross1(?a) ^ subnrCross(?x, ?n) ^ ClassCross2(?x) ^ ogc:sfCrosses(?x, ?a) → resultForData(DQR_SWRLCrosses, ?a)

Second rule:

sdqo:ClassCross2(?x) ^ sdqo:subnrCross(?a, ?n) ^ swrlb:stringConcat(?aa, ?ida, " , ", ?idx, " , ") ^ sdqo:featureID(?x, ?idx) ^ sdqo:featureID(?a, ?ida) ^ sdqo:subnrCross(?x, ?n) ^ ogc:sfCrosses(?x, ?a) ^ sdqo:ClassCross1(?a) sdqo:hasMessage(DQR_SWRLCrosses, ?aa)

The second rule uses sdqo:featureID which should be a super property to an feature identifying data property in SfO. It can be labelled sfo:fID. Reasoning process follows, to complete the ontology. The SWRL rules are shown in Figure 59.



```
sdqo:ClassCross2(?<urn:swrl#x>), sdqo:subnrCross(?<urn:swrl#a>, ?<urn:swrl#n>),
swrlb:stringConcat(?<urn:swrl#aa>, ?<urn:swrl#ida>, " , "^^xsd:string, ?<urn:swrl#idx>, " ,
"^^xsd:string), sdqo:featureID(?<urn:swrl#x>, ?<urn:swrl#idx>), sdqo:featureID(?<urn:swrl#a>,
?<urn:swrl#ida>), sdqo:subnrCross(?<urn:swrl#x>, ?<urn:swrl#n>), ogc:sfCrosses(?<urn:swrl#x>,
?<urn:swrl#a>), sdqo:ClassCross1(?<urn:swrl#a>) -> sdqo:hasMessage(sdqo:DQR_SWRLCrosses,
sdqo:errorCode(?<urn:swrl#r>, ?<urn:swrl#n>), sdqo:resultForData(?<urn:swrl#r>, ?<urn:swrl#a>) ->
sdqo:dataHasErrorWithCode(?<urn:swrl#a>, ?<urn:swrl#n>)
sdqo:subnrDDP(?<urn:swrl#a>, ?<urn:swrl#n>), sdqo:testedAttribute01(?<urn:swrl#a>,
?<urn:swrl#da>), sdqo:ClassDistinctDataProp(?<urn:swrl#a>), DifferentFrom (?<urn:swrl#a>,
?<urn:swrl#x>), sdqo:testedAttribute01(?<urn:swrl#x>, ?<urn:swrl#da>),
swrlb:stringConcat(?<urn:swrl#aa>, ?<urn:swrl#ida>, " , "^^xsd:string, ?<urn:swrl#idx>, " ,
"^^xsd:string), sdqo:subnrDDP(?<urn:swrl#x>, ?<urn:swrl#n>), swrlb:lessThan(?<urn:swrl#ida>,
?<urn:swrl#idx>), sdqo:ClassDistinctDataProp(?<urn:swrl#x>), sdqo:featureID(?<urn:swrl#x>,
?<urn:swrl#idx>), sdqo:featureID(?<urn:swrl#a>, ?<urn:swrl#ida>) ->
sdqo:hasMessage(sdqo:DQR_SWRLDDP, ?<urn:swrl#aa>)
sdqo:resultForData(sdqo:DQR_SWF, ?<urn:swrl#x>), sdqo:featureID(?<urn:swrl#x>, ?<urn:swrl#idx>)
-> sdqo:hasMessage(sdqo:DQR_SWF, ?<urn:swrl#idx>)
sdqo:crossesOrOverlaps(?<urn:swrl#x>, ?<urn:swrl#a>), sdqo:CalcPoly(?<urn:swrl#x>),
sdqo:CalcPoly(?<urn:swrl#a>) -> ogc:sfOverlaps(?<urn:swrl#x>, ?<urn:swrl#a>)
sdqo:ClassMustWithin2(?<urn:swrl#x>), sdqo:ClassMustWithin1(?<urn:swrl#a>),
sdqo:subnrMustWithin1(?<urn:swrl#a>, ?<urn:swrl#n>), sdqo:subnrMustWithin2(?<urn:swrl#x>,
?<urn:swrl#n>), sqwrl:makeSet(?<urn:swrl#set>, ?<urn:swrl#x>), sqwrl:groupBy(?<urn:swrl#set>,
?<urn:swrl#a>), sqwrl:size(?<urn:swrl#siz>, ?<urn:swrl#set>) -> sqwrl:select(?<urn:swrl#a>,
?<urn:swrl#siz>)
swrlb:intersectionMatrix("F**T*****"^^xsd:string, ?<urn:swrl#wx>, ?<urn:swrl#wa>),
ogc:asWKT(?<urn:swrl#gx>, ?<urn:swrl#wx>), ogc:asWKT(?<urn:swrl#ga>, ?<urn:swrl#wa>),
ogc:hasGeometry(?<urn:swrl#x>, ?<urn:swrl#gx>), ogc:hasGeometry(?<urn:swrl#a>,
?<urn:swrl#ga>) -> ogc:sfTouches(?<urn:swrl#x>, ?<urn:swrl#a>)
sdqo:ClassSameDataProp(?<urn:swrl#a>), sdqo:subnrSDP(?<urn:swrl#a>, ?<urn:swrl#n>),
sdqo:dataProp01(?<urn:swrl#x>, ?<urn:swrl#dx>), sdqo:dataProp01(?<urn:swrl#a>,
```

Figure 59. SWRL rules in SDQO.

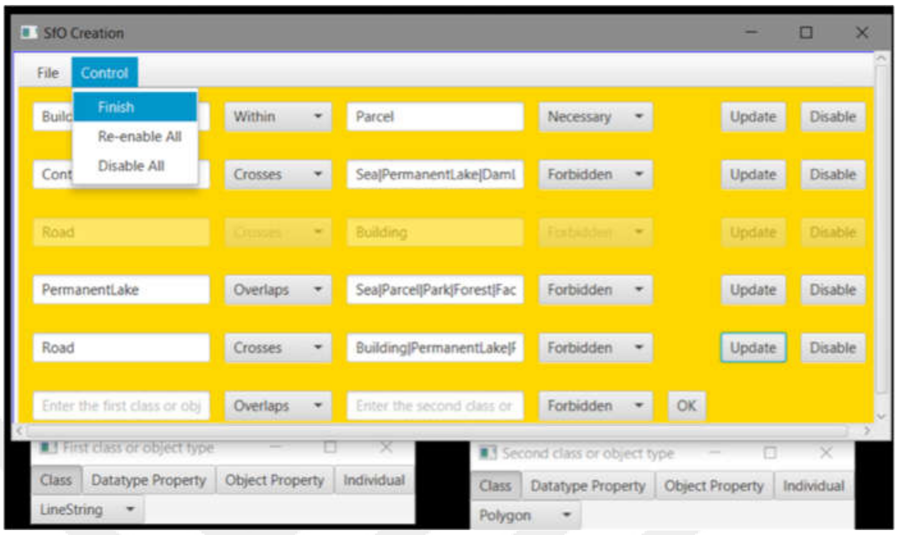
2.2.4.6. Case Study 3

The third case study in the thesis involves spatial data quality assessment with ontologies. There are two datasets. For the first and larger one, the basemap of Trabzon province is clipped and applied to regulations based on Turkish LSMMPR and General Command Mapping legislation. This dataset has thousands of features; including buildings, parcels and roads. The second dataset is created by us and is smaller, with intentional quality errors.

SfO is selected/created according to the following steps.

1. If the user knows the appropriate SfO, the user can use this SfO.
2. The user may compare database schema against existing SfO. The SfOs with a similarity above a given threshold are listed, sorted by the sum of the distances. The user can choose one of the SfOs. If no SfO in the system is similar enough, the user is prompted to create a new SfO as in step 3. The column names of schema are compared against best matching specification classes.
3. Using the SfO creation interface as in Figure 60 and a sample database, SfO can be created. The SfO creation needs to be done by an expert of the domain and according to

its specifications. For each SfO, a Turtle file of the ontology, an R2RML mapping file and a post-reasoning complete CSV file summarizing the ontology are created.

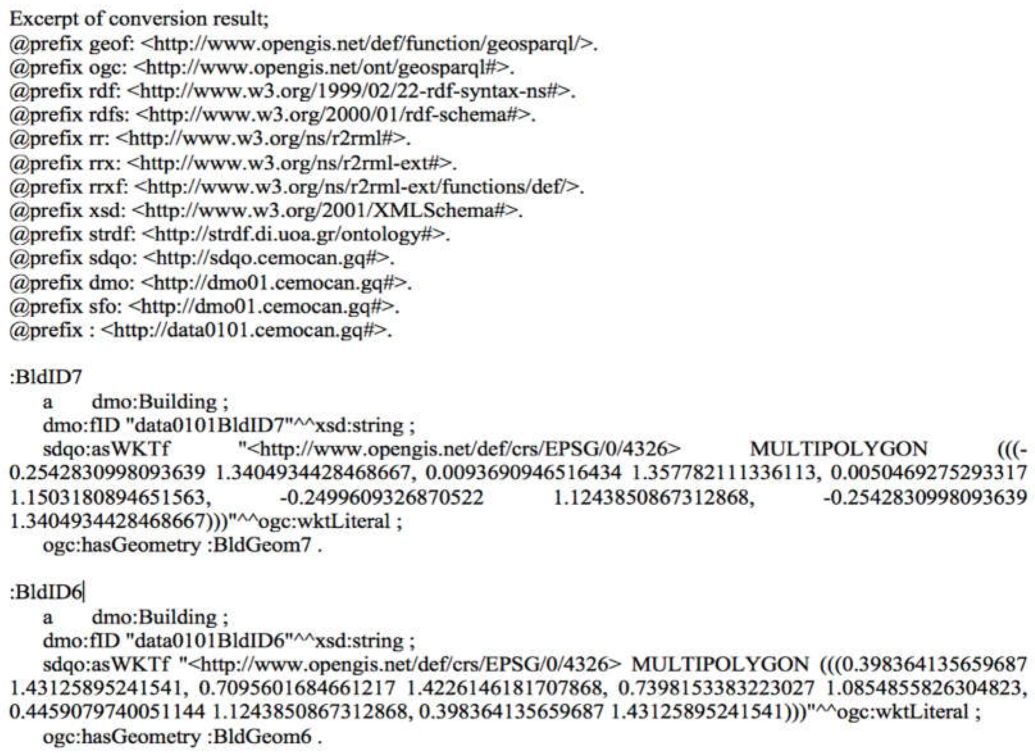


Build	Within	Parcel	Necessary
Cont	Crosses	Sea|PermanentLake|DamL	Forbidden
Road	Crosses	Building	Forbidden
PermanentLake	Overlaps	Sea|Parcel|Park|Forest|Fac	Forbidden
Road	Crosses	Building|PermanentLake|F	Forbidden
Enter the first class or obj	Overlaps	Enter the second class or	Forbidden

Figure 60. Graphical User Interface for SfO creation.

If features in a class A of line geometry must not overlap features in classes B and C of polygon geometry according to the specification, this is represented by "c1, c2, Forbidden, Overlaps, A, B | C, timestamp_value". SfO creation and modification is further explained in the next subsection. The timestamp value should be common. If they differ by at most a hundredth of a day, the minimum one is taken and used as annotation for the SfO.

Data conversion to ontology: R2RML mappings are automatically created for each relevant database table column towards classes in SfO. (database table column) to be tested. Once mappings are created, they can be stored for further quality assessment of the same institution. The excerpt of data after conversion is can be seen on Figure 61.



```
Excerpt of conversion result;
@prefix geof: <http://www.opengis.net/def/function/geosparql/>.
@prefix ogc: <http://www.opengis.net/ont/geosparql#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix rrx: <http://www.w3.org/ns/r2rml-ext#>.
@prefix rrxf: <http://www.w3.org/ns/r2rml-ext/functions/def/>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix strdf: <http://strdf.di.uoa.gr/ontology#>.
@prefix sdqo: <http://sdqo.cemocan.gq#>.
@prefix dmo: <http://dmo01.cemocan.gq#>.
@prefix sfo: <http://dmo01.cemocan.gq#>.
@prefix : <http://data0101.cemocan.gq#>.

:BldID7
    a      dmo:Building ;
    dmo:fID "data0101BldID7"^^xsd:string ;
    sdqo:asWKTf          "<http://www.opengis.net/def/crs/EPSG/0/4326>          MULTIPOLYGON          (((-
0.2542830998093639 1.3404934428468667, 0.0093690946516434 1.357782111336113, 0.0050469275293317
1.1503180894651563,         -0.2499609326870522          1.1243850867312868,          -0.2542830998093639
1.3404934428468667)))"^^ogc:wktLiteral ;
    ogc:hasGeometry :BldGeom7 .

:BldID6|
    a      dmo:Building ;
    dmo:fID "data0101BldID6"^^xsd:string ;
    sdqo:asWKTf "<http://www.opengis.net/def/crs/EPSG/0/4326> MULTIPOLYGON (((0.398364135659687
1.43125895241541, 0.7095601684661217 1.4226146181707868, 0.7398153383223027 1.0854855826304823,
0.4459079740051144 1.1243850867312868, 0.398364135659687 1.43125895241541)))"^^ogc:wktLiteral ;
    ogc:hasGeometry :BldGeom6 .
```

Figure 61. Excerpt of conversion result.

Integrating data ontology with SfO compatible with rules: SfO is created according to the rules to be tested. Data ontologies are associated with the related classes. In this case, SfO created with generic classes.

The R2RML mapping can also employ a datatype property mapping for lineage using the timestamp values. With the latest implementation, the erroneous data are timestamped, solely. One can add an annotation property to the SfO.

SfOs import the data ontologies and SDQO. Each SfO has a graphical interface, using the lastly used R2RML file initially. If necessary, the R2RML file is updated. Running the assessment service with the SfO interface, the erroneous data is extracted. The ontology with error messages is obtained and the result is passed to the endpoint and the graphical backend with Sextant.

2.2.4.6.1. SfO Creation

A graphical user interface is employed as shown in Figure 60 to create a SfO. For each restriction, the domain expert fills a row. New blank rows are added after each row. In each row on the graphical user interface, there are two text fields and two combo-boxes. When the text fields are filled, popup menus appear. The domain expert makes a choice between Class, Datatype Property, Object Property and Individuals, for the name entered in the text field. For geometric classes, the domain expert also asserts the geometry type with a popup window. With the first combo-box, one selects the relations, for instance “Overlaps” for geometric classes. Second combo-box gives the types of restriction, such as “Forbidden”, “Necessary” or “Equivalent”. When these are selected, and the OK button is pressed, a CSV row is produced for the entered data and a new row is shown on the graphical user interface. It is expected that the domain expert will quickly be able to grasp the CSV and to some extent, the SfO hierarchy mechanism.

2.2.4.6.1.1. GUI to CSV

The first entry on the CSV row is the type of the first entity, and the second entry is the type of the second entity. “d”, “o”, “i” stand for “Datatype Property”, “Object Property”, “Individual”, respectively. More commonly, “c2”, “c1”, “c0” or “cc” are there, which correspond to polygonal class, line-type geometric class, point-type geometric class, general class, respectively. The third and fourth entries on the CSV row are the restriction type for and the name of the relation, for instance “Forbidden, Overlaps”. For geometric classes, the OGC spatial relations are listed. In other cases, relations such as “Same”, “LessThan”, “LessThanEQ” and “OneOf” are used. The fifth and the sixth entries are the names of the first and second entities. The last entry gives the time. In case, the timestamps are close, the SfO is annotated using that timestamp with schema:startTime.

If the second entity is empty, it is a “within Class” -type relation. One can have several classes for an entity, separating them with the symbol “|”. This serves as a separator between distinct classes; occasionally can also be considered as “OR”.

If one wants to declare that features in the line-type class “Contour” may not intersect features in the polygonal type classes “Building”, “Lake”, “Sea”, this can be done using the CSV row “c1, c2, Forbidden, Intersects, Contour, Building|Lake|Sea, timestamp_value”.

For the timestamp, System. currentTimeMillis() is used. The starting date is January 1, 2000 00:00; it is subtracted to get the number of miliseconds. The result is divided by 8640000, to get the number of days. The timestamp value is then rounded upwards to the

1/100th of a day (14 minutes 24 seconds). 1/100th of a day should also be enough for the execution of at least the initial phase.

Before passing to the next stage, where new triples are added to the new SfO for each CSV, the CSV rows are reorganized. First, all piped class sets are split. One row exists for each class etc. pair. The symmetricity and other properties of the relations are used to switch the class names when necessary, so that the first one is either of smaller geometry type (point vs line or polygon; line vs polygon) or "smaller" name if they are of the same type (encyclopaedic ordering). They are sorted by the first, second, third, fourth and fifth entry (geometry types of the two entities, relation restriction type, relation name, name of the first entity). If the first five entries are the same for two rows, they are collected together with the "|". Now, for the geometry classes case, first entity is a single class, but the second entity can consist of several classes, each of the same geometry type.

The Finish item in the Menu Bar finishes the process except passing any additional notes to the control and ontology expert. The GUI is not restricted to the geometric class relations. The domain expert still may choose to add more restriction using the additional notes screen. For instance, associations to the other SfOs can be mentioned there ("This new SfO is the 1/1000 scale version of this other, already well-established SfO"). Next, the SfO is created in Turtle format. Once SfO is created, through an interface or manually using an ontology editor, the SfO can be updated. Removing generic classes corresponds to removing specification rules. Subclass relations can be changed to reflect changes in the specification.

2.2.4.6.1.2. CSV to Turtle to CSV and R2RML

The SfO creation service adds, using the OWLAPI, several triples to the newly created SfO, for each CSV row. CSV reorganization in the previous section should be done before this step. At the end of this step, the newly created SfO is processed with Openllet/Pellet reasoner, to complete the SfO. The complete SfO is summarized into a CSV file, that would produce the SfO. This complete CSV is shown to the domain expert for confirmation. An R2RML file is generated using the class and property names, etc. If desired, timestamp addition can also be included in the R2RML file. The remaining part of this subsection explains in more detail the transformation from CSV rows to Turtle SfO, in various cases.

Case 1: Geometric classes with Forbidden relation (Inter-object Must Not).

Example: “c1, c2, Forbidden, Crosses, class1, class2|class3, 6210.12”

Meaning: “class1 is of line-type, class2 and class3 are of polygon-type, features in class1 cannot have the Crosses relation with the features in the classes class2 and class3” 6210.12 is the timestamp, which is the same throughout the process (the minimal one).

Two generic classes are created in the SfO; they are set as subclasses of the relevant classes in SDQO and subclasses of the restriction for the appropriate subnr sub-property having value equal the row index. If there are already six “c1,c2, Forbidden,Crosses” pairs, the created classes are sfo:ClassCrf07 and sfo:ClassCrs07. Here “Cr” stands for “Crosses”, “f” stands for “first” and “s” stands for “second”. Both of these classes are set to be subclasses of the restriction [sdqo:subnrCross = 7]. Restrictions are among anonymous classes in OWL. sfo:ClassCrf07 is set as a subclass of SDQO class sdqo:ClassCross1 and sfo:ClassCrs07 is set as a subclass of sdqo:ClassCross2.

Each specification class is created if they are not already present. The subclass relations are established. sfo:class1 is created as a subclass of sfo:ClassCrf07. sfo:class2 and sfo:class3 are created as subclasses of sfo:ClassCrs07. Furthermore, they are subclasses of SDQO classes for their asserted geometry types. “c2”, “c1” and “c0” stand for polygon-type, line-type and point-type features, respectively. A sample for relevant class hierarchies are given below with the associated SWRL rule. “<” denotes the subclass relation. Case 1 type cases (“Inter-object Must Not”) are easily updated by updating the subclass relations of the specification classes or creating new classes with the appropriate subclass relations.

Sample class hierarchy paths:

- sfo:Road < sfo:ClassCrf07 < sdqo:ClassCross1 < sdqo:Cross < sdqo:InterObjectPrRF < sdqo:RestrictedFeature < ogc:Feature < ogc:SpatialObject < owl:Thing
- sfo:Road < sfo:ClassCrf07 < [sdqo:subnrCross = 7]
- sfo:Building < sfo:ClassCrs07 < sdqo:ClassCross2 < sdqo:Cross < sdqo:InterObjectPrRF < sdqo:RestrictedFeature < ogc:Feature < ogc:SpatialObject < owl:Thing
- sfo:Building < sfo:ClassCrs07 < [sdqo:subnrCross = 7]

Associated SWRL rules (need to be free from SfO, order does not matter):

- sdqo:ClassCross1(?a) ^ sdqo:ClassCross2(?x) ^ogc:sfCrosses(?a,?x) ^ sdqo:subnrCross(?a,?n) ^ sdqo:subnrCross(?x,?n) → resultForData(DQR_SWRLCross, ?a)

- sdqo:ClassCross2(?x) ^ sdqo:subnrCross(?a, ?n) ^ swrlb:stringConcat(?aa, ?ida, " , ", ?idx, " , ") ^ sdqo:featureID(?x, ?idx) ^ sdqo:featureID(?a, ?ida) ^ sdqo:subnrCross(?x, ?n) ^ ogc:sfCrosses(?x, ?a) ^ sdqo:ClassCross1(?a) sdqo:hasMessage(DQR_SWRLCross, ?aa)

"If a and x, despite being in sdqo:ClassCross1 and sdqo:ClassCross2, respectively, still cross with each other, then there is an error."

Case 2: One geometric class with Forbidden relation (Intra-object Must Not)::

Example: "c2, , Forbidden, Overlaps, class1, , 6210.12" (Second class is empty)

Meaning: "class1 is of polynomial type, and features in class1 are either the same or not overlapping".

If there are already two "c2,,Forbidden,Overlaps" pairs, the generic class sfo:ClassOv03 is created; "Ov" stands for "Overlaps" and there is no "f" or "s", unlike in first case above. sfo:ClassOv03 Is set as a subclass to the related SDQO base class sdqo:ClassOverlaps and to the restriction [sdqo:subnrOverlaps = 3]. The specification class sfo:class1 is created, if it does not already exist. It is set to be a subclass of the generic class sfo:ClassOv03.

A sample for relevant class hierarchies are given below with the associated SWRL rule. Case 2 type cases ("Intra-object Must Not") are easily updated as in case 1 above.

Sample class hierarchy paths:

- sfo:Building < sfo:ClassOv03 < sdqo:ClassOverlaps < sdqo:IntraObjectPrRF < sdqo:RestrictedFeature < ogc:Feature < ogc:SpatialObject < owl:Thing
- sfo:Building < sfo:ClassOv03 < [sdqo:subnrOverlaps = 7]

Associated SWRL rule:

- sdqo:ClassOverlaps(?a) ^ sdqo:featureID(?x, ?idx) ^ ogc:sfOverlaps(?x, ?a) ^ sdqo:featureID(?a, ?ida) ^ sdqo:ClassOverlaps(?x) ^ swrlb:stringConcat(?aa, ?ida, " , ", ?idx, " , ") ^ swrlb:lessThan(?ida,?idx) sdqo:subnrOverlaps(?a,?n) ^ sdqo:subnrOverlaps(?x,?n) → sdqo:hasMessage(DQR_SWRLOverlaps, ?aa)

"If a and x with different sdqo:featureID values, despite being in the class sdqo:ClassOverlaps with the same sdqo:subnrOverlaps data property value, still overlap with each other, then there is an error."

Case 3: Geometric classes with Necessary relation (Inter-object Must).

Example: “c2, c2, Necessary, Within, class1, class2, 6210.12”

Meaning: “class1 and class2 are of polygon-type, any feature in class1 must be within at least one feature in class2.”

As above, two generic classes are created. The superclass that is a base class in SDQO is associated with a SQWRL rule; SWRL does not suffice. Here negation is needed.

Sample class hierarchy paths:

- sfo:Building < sfo:ClassMustWithinf01 < sdqo:ClassMustWithin1 < sdqo:Within < sdqo:InterObjectPrRF < sdqo:RestrictedFeature < ogc:Feature < ogc:SpatialObject < owl:Thing
- sfo:Building < sfo:ClassMustWithinf01 < [sdqo:subnrMustWithin1 = 1]
- sfo:Parcel < sfo:ClassMustWithins01 < sdqo:ClassMustWithin2 < sdqo:Within < sdqo:InterObjectPrRF < sdqo:RestrictedFeature < ogc:Feature < ogc:SpatialObject < owl:Thing
- sfo:Parcel < sfo:ClassMustWithins01 < [sdqo:subnrMustWithin2 = 1]

According to the size of containing features in sfo:Parcel, features in sfo:Building are selected.

Case 4: One class with Forbidden type attribute restriction (Intra-data Must not).

Example: “c1, d, Forbidden, Same, Road, RoadID,6210.12”

Meaning: “linestring-type class Road or subclasses must have different values for the hasRoadID attribute”.

sfo:hasRoadID is created as a datatype property equivalent to sdqo:testedAttribute01. sfo:Road is created as a subclass of a generic SfO class sfo:ClassDDP01 which is a subclass to sdqo:ClassDistinctDataProp and restriction [sdqo:subnrDDP = 1].

Sample class hierarchy paths:

- sfo:Road < sfo:ClassDDP01 < sdqo:ClassDistinctDataProp < sdqo:IntraDatatPrRF < sdqo:RestrictedFeature < ogc:Feature < ogc:SpatialObject < owl:Thing
- sfo:Road < sfo:ClassDDP01 < [sdqo:subnrDDP = 1]

Associated SWRL rules:

- owl:differentFrom(?a, ?x) ^ sdqo:testedAttribute01(?a, ?d) ^ sdqo:testedAttribute01(?x, ?d) ^ sdqo:subnrDDP(?a, ?n) ^ sdqo:subnrDDP(?x, ?n) ^ sdqo:ClassDistinctDataProp(?a) ^ sdqo:ClassDistinctDataProp(?x) → resultForData(DQR_SWRLDDP, ?a)

- owl:differentFrom(?a, ?x) ^ sdqo:testedAttribute01(?a, ?d) ^ sdqo:testedAttribute01(?x, ?d) ^ sdqo:subnrDDP(?a, ?n) ^ sdqo:subnrDDP(?x, ?n) ^ sdqo:ClassDistinctDataProp(?a) ^ sdqo:ClassDistinctDataProp(?x) ^ sdqo:featureID(?a, ?ida) ^ sdqo:featureID(?x, ?idx) ^ swrlb:lessThan(?ida, ?idx) ^ swrlb:stringConcat(?aa, ?ida, " , ", ?idx, " , ") → sdqo:hasMessage(DQR_SWRLDDP, ?aa)

"If a and x, asserted to be different, despite being in the class sdqo:ClassDistinctDataProp with the same sdqo:subnrDDP data property value, still have the same value for the data property sdqo:testedAttribute01 (and therefore any equivalent data property), then there is an error."

Case 5: One class with Necessary type attribute restriction (Intra-Data Must), single value.

Example: "c2, d, Necessary, Same, Sea, Elevation,6210.12"

Meaning: "polygon-type class Sea or subclasses must have same value for the hasElevation attribute".

sfo:hasElevation is created as a datatype property equivalent to sdqo:dataProp01. sfo:Sea is created as a subclass of a generic SfO class sfo:ClassSDP01 which is a subclass to sdqo:ClassSameDataProp and restriction [sdqo:subnrSDP = 1].

Sample class hierarchy paths:

- sfo:Sea < sfo:ClassSDP01 < sdqo:ClassSameDataProp < sdqo:IntraDatatPrRF < sdqo:RestrictedFeature < ogc:Feature < ogc:SpatialObject < owl:Thing
- sfo:Sea < sfo:ClassSDP01 < [sdqo:subnrSDP = 1]

Associated SWRL rules:

- sdqo:ClassSameDataProp(?a) ^ sdqo:subnrSDP(?a, ?n) ^ sdqo:dataProp01(?x, ?dx) ^ sdqo:dataProp01(?a, ?da) ^ swrlb:lessThan(?dx, ?da) ^ sdqo:subnrSDP(?x, ?n) ^ sdqo:ClassSameDataProp(?x) → sdqo:resultForData(DQR_SWRLSDP, ?a)
- sdqo:ClassSameDataProp(?x) ^ sdqo:featureID(?x, ?idx) ^ sdqo:featureID(?a, ?ida) ^ sdqo:ClassSameDataProp(?a) ^ sdqo:subnrSDP(?a, ?n) ^ sdqo:dataProp01(?a, ?da) ^ sdqo:dataProp01(?x, ?dx) ^ swrlb:lessThan(?dx, ?da) ^ swrlb:stringConcat(?aa, ", ", ?ida, ", ", ?idx, ", ") ^ sdqo:subnrSDP(?x, ?n) → sdqo:hasMessage(DQR_SWRLSDP, ?aa)

"If a and x, despite being in the class sdqo:ClassSameDataProp with the same sdqo:subnrSDP data property value, still have the same value for the data property sdqo:dataAttribute01 (and therefore any equivalent data property), then there is an error."

Case 6: One class with Forbidden type attribute restriction, multiple given values (Intra-data Must not).

Example: “c2,d,Forbidden,OneOf,Building,NumberOfFloors|13|4,6210.12”

Meaning: “polygon-type class Building features may not have hasNumberOfFloors value in the set (4, 13)”

This rule is recommended to be within SfO. Fixed references for values 13 and 4 are created outside sfo:Building, within sdqo:FixedRefFeature.

- sdqo:FixedRefFeature(?fr) ^ sfo:hasNumberOfFloors(?fr,?n) ^sfo:Building(?x) ^sfo:hasNumberOfFloors(?x,?n) ^ sdqo:featureID(?x,?idx) ^swrlb:stringConcat(?aa,” erroneous number of floors (“, ?n,”),”,?idx, “ ,”) → sdqo:hasMessage(DQR_MGV,?aa)

Here, instead of “OneOf" relation, relations “LessThan”, “LessThanEQ”, “GreaterThan” and “GreaterThanEQ” can also be used. “Same” relation can be used for setting equal to a fixed value, which is equivalent to “OneOf" with only one value.

2.2.4.6.2. Assessment and Report

SDQO and relevant transactions are responsible for assessment. Once the SfO is created for a domain and integrated with SDQO, then reasoner in the system will infer the quality results for the tested features. As a result, a quality report is created including the error codes with inconsistent features. The report can be dumped into a SPARQL endpoint for further queries. The report can be visualized with the help of visualization tools that support endpoints such as SEXTANTE[18].

70 of 4395 roads (1.6 %) are erroneous, they are crossing with one or more buildings as shown in Figure 62 .

587 of parcels (1.7%) are overlapping with other parcels as shown in Figure 62. There are 33641 parcels.

[18] http://sextant.di.uoa.gr/

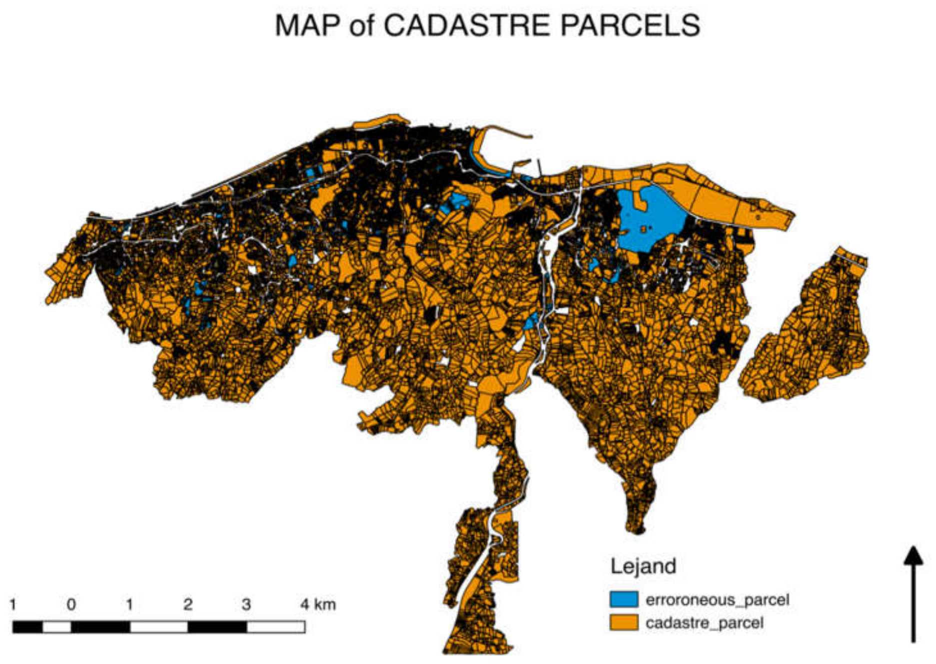


Figure 62. Cadastre parcels. Parcels with errors are marked blue.

1232 buildings are overlaping with other buildings. Totally 13815 of 20595 of buildings are erroneous (67.1%) as shown in Figure 63; two thirds of the buildings fail to be spatially within one parcel.

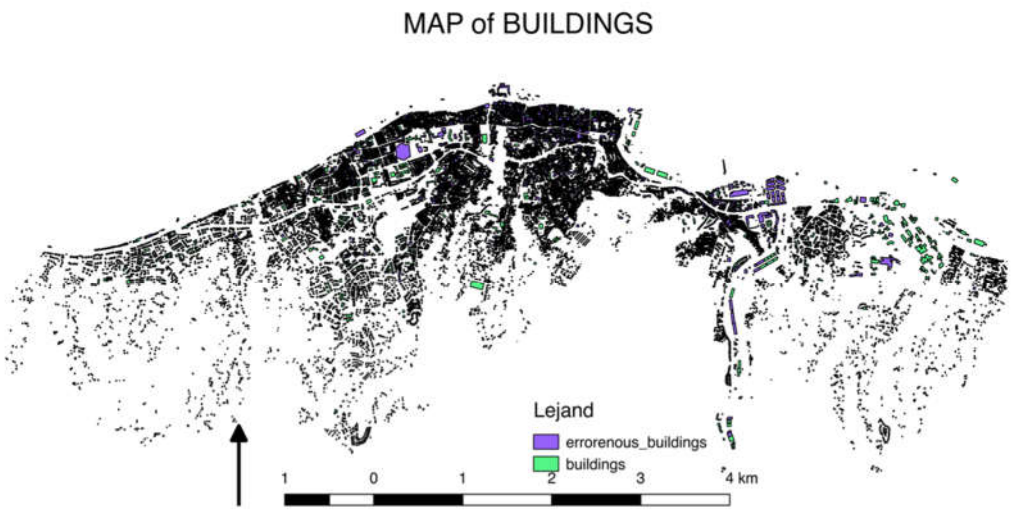


Figure 63. Buildings. Buildings with error are marked purple.

A zoomed map of erroneous features along with cadastre parcels are shown in Figure 64. There one can see that the erroneous road crosses the building and other errors.

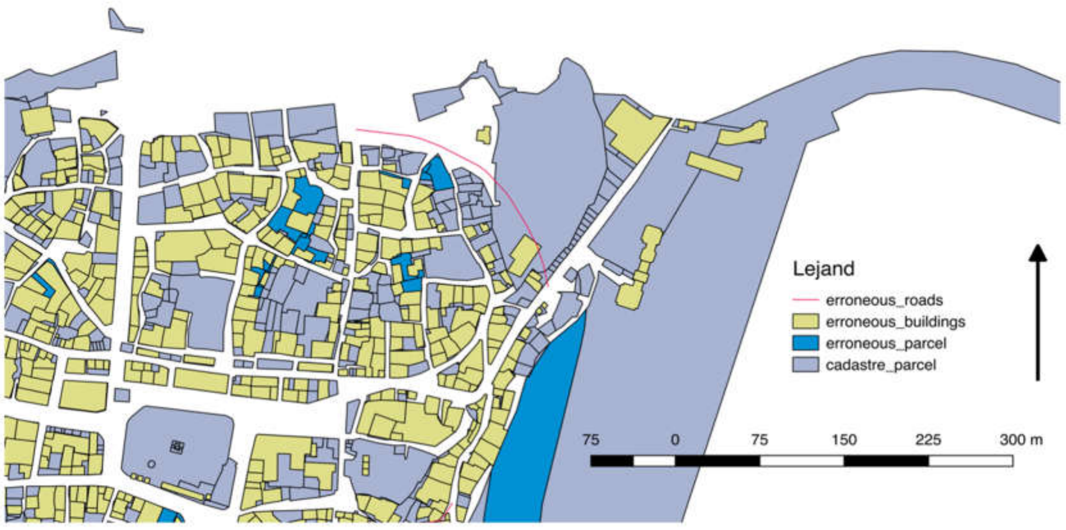


Figure 64. Erroneous features. Red road crosses with the yellow buildings.

3. DISCUSSION AND CONCLUSION

In the study, firstly data quality elements are discussed. There are several data quality assessment concepts and elements. Many of the concepts and elements have some similarity, sometimes with divergences especially in the meaning, also depending on the domain of application. As explained in the section 1.2.1, some elements appear frequently in the standards such as ISO 19157:2013 and other literature. Classification of the quality elements enhances assessment processes for shareability and usability; classification also helps with standardized quality reports. This increases efficiency in comparing the results of quality assessments. There are several categorizations seen in the standards and literature. One basic categorization is external (user-side) and internal (producer-side) data quality. The internal data quality is the consistency of data with the specifications. The internal data quality is further categorized into data quality elements and sub-elements. Associated with these elements and sub-elements, for assessment, data quality metrics and processes are introduced. Quality metrics are classified according to the associated quality elements.

The spatial data are at the central point of Geomatics Engineering and there are several solutions for assessment. Many professionals use the proprietary solutions such as ArcGIS Data Reviewer. The specifications are either not enough or not applied well enough in the field. Professionals, especially when they are from different background or institutions, introduce their own version of solutions on the field. These have been causing problems and inconsistency; a lot of resources are lost. Furthermore, the same data is over and over assessed by the professionals. In the age of social communications, it is even easier to interoperate, if the problems restricting this is resolved.

There is a need for a framework for domain-independent (spatial) data quality assessment that is reusable, robust, extensible and usable with a non-proprietary solution with web-based access. With this study, quality assessment processes are investigated in the paradigms of Open World and Closed World Assumptions, and a framework is designed for spatial quality assessment with a solution following the research objectives mentioned above.

The system is to be accessed by users with expertise in different domains. Domain independence and usability imply that users are not demanded what they are not expected to be capable of. The system does not only look for the existence of the errors; it also extracts the source of errors. The system needs to be robust against at least expected types of errors.

In the thesis, three case studies are implemented; the first one is for CWA (Prolog-based), the others are for OWA (OWL-based); with either queries or SWRL rules.

For the first case study, Prolog based quality assessment is implemented with the dataset including three classes. Some sample rules are implemented. In the implementation, it is seen that the system is not easily updatable. In Prolog, as it is common in Closed World system, the order of the rules is important and affects the result of the assessment. If there is a need to implement a new rule, some of the Prolog rules are likely to be rewritten. A new Closed World is in order and the system needs to be redesigned. In the thesis, several rules are created with Prolog. Two of the rules are; "Buildings that are outside of the cadastral parcel" and "Planning zone area that has type Park contains buildings". First rule is implemented with the use of Negation as Failure feature of Prolog programming. The second rule includes the attribute of the class for quality testing.

In Closed World, anything that cannot be proven true is set to be false. There are no unknowns; any change requires a whole redefining. Therefore, usually it can only be well suited for static structures. During assessment, Closed World (SPARQL) queries are also used.

The second implementation is based on ontologies with GeoSPARQL queries applied on them. This can be considered as, both Closed World and Open World implementation. The queries are done with closing the world while the data still are OWL based. SPARQL based querying became more feasible with the introduction of SPIN and SHACL. SPARQL has GeoSPARQL for spatial capabilities. SHACL will have JavaScript capabilities. Java has JTS, Python has GEOS for spatial operations. There are attempts for developing JTS-like libraries for JavaScript. When a JTS like library can be used by SHACL or a successor, the second implementation will become most feasible among three implementations.

The third implementation is OWL-based with SWRL rules, so Open World reasoning. According to Reiter (1981), Closed World is, in general problematic for assessment and evaluation, but the Horn-type cases are good; furthermore they suggest that any Closed World query can be translated into Open World. Therefore, the third implementation is possible, every time the first two implementations are possible.

For the third implementation, a framework is designed towards the research objectives. It is based on two types of ontologies. They are; SDQO, the main ontology, and SfOs, the specification ontologies. The separation helps domain-independence, reusability and extendibility. SfOs are created based on the specifications or users’ requirements and designed consistent with the SDQO. SfOs mostly comprise class hierarchies, and therefore a domain expert can easily learn correctly updating and manipulating them when necessary. A user interface is designed to ease the creation of the ontologies for the users. SDQO has general classes for the concepts that are needed for spatial quality assessment. In a specification, there may be spatial rules for a class against some of the classes and their subclasses. For example, a “building” must not overlap with, “sea”, “lake” and subclasses or other buildings. These are represented in the specification with different rules for each relation. In the framework, common rules with different classes are optimized, also according to the spatial relations.

The SfOs are hierarchical; there are the specification classes such as Building and Road and there are generically named superclasses such as ClassOvf01 and ClassOvs01. These superclasses in SfOs are subclasses of SDQO “ClassOverlap1” and “ClassOverlap2”, respectively. They are also subclasses of appropriate restrictions in SDQO. SDQO ontology rules are implemented using these subclass relations and quality assessment follows. Any building overlapping with the features in sea, lake classes are found without causing any inconsistency in the ontologies due to robustness. Also, using properties such as symmetricity of the spatial relation increases the efficiency. This way, forbidden type spatial relations and attribute relations are easily implemented. Necessary type spatial relations and attribute relations can also be implemented, but one needs to be especially careful, as the negation can be problematic in Open World reasoning. Rules can be implemented in SfOs for this kind of cases.

In the case studies, there are several options for input such as MySQL/PostGIS relational databases, ESRI shapefiles and RDF data. The studies are oriented towards institutional data and related specifications. For further studies, the intention is to allow employing SPARQL endpoints and make use of linked data, both for input and output. The specifications for linked data, data streams, VGI and institutional/stationary data have some significant difference. The ontology SDQO in the case study is not designed towards the nature of linked data.

SHACL is a promising W3C recommendation for RDF validation. In the future, SHACL can be incorporated within the framework. SHACL also has a version with similar design but for JavaScript instead of SPARQL (URL-13, 2017). This version can be used with JavaScript libraries alternative to Java’s JTS, when they are developed.

4. BIBLIOGRAPHY

Acciarri, A., Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Palmieri, M. and Rosati, R., 2005. Quonto: Querying Ontologies, The Twentieth National Conference on Artificial Intelligence, Pittsburgh, PA, USA, 5: 1670-1671.

Alatrish, E., 2013. Comparison Some of Ontology Editors, Journal of Management Information Systems, 8,2, 18-24.

Anellis, I. H., 2012. Peirce's Truth-Functional Analysis and the Origin of the Truth Table, History and Philosophy of Logic, 33,1, 87-97.

Antoniou, G. and Harmelen, F. V., 2008. A Semantic Web Primer, Choice Reviews Online, 46,03, 46-1523.

Baader, F., Calvanese, D., McGuinness, D., Patel-Schneider, P. and Nardi, D., 2003. The Description Logic Handbook: Theory, Implementation and Applications, Cambridge University Press, Cambridge.

Bagosi, T., Calvanese, D., Hardi, J., Komla-Ebri, S., Lanti, D., Rezk, M., Rodríguez-Muro, M., Slusnys, M. and Xiao, G., 2014. The Ontop Framework for Ontology Based Data Access, Eighth Chinese Semantic Web and Web Science Conference, Wuhan, China: 67-77.

Ballatore, A., Wilson, D. C. and Bertolotto, M., 2013. Quality Issues in the Management of Web Information, A Survey of Volunteered Open Geo-Knowledge Bases in the Semantic Web, Springer, 93-120.

Batini, C. and Scannapieca, M., 2006. Methodologies for Data Quality Measurement and Improvement, Data Quality: Concepts, Methodologies and Techniques, 161-200.

Bechhofer, S., Volz, R. and Lord, P., 2003. Cooking the Semantic Web with the OWL API, Second International Semantic Web Conference, Sanibel Island, FL, USA: 659-675.

Begam, M. F. and Ganapathy, G., 2011. Knowledge Engineering Approach for Constructing Ontology for E-Learning Services, Advanced Computer Science and Information System (ICACSIS), 2011 International Conference on: 125-132.

Berners-Lee, T., 2000. Semantic Web, XML 2000, Washington DC, USA.

Berners-Lee, T., 2006. Artificial Intelligence and the Semantic Web, The Twentyfirst National Conference on Artificial Intelligence, Boston, USA.

Berners-Lee, T., Fielding, R. T. and Masinter, L. 2005. Uniform Resource Identifier (Uri): Generic Syntax.

Berners-Lee, T. and Hendler, J., 2001. Publishing on the Semantic Web, Nature, 410,6832, 1023-1024.

Berners-Lee, T., Hendler, J. and Lassila, O., 2001. The Semantic Web, Scientific American, 284,5, 34-43.

Blackburn, P., Bos, J. and Striegnitz, K., 2006. Learn Prolog Now!, College Publications Londres.

Bramer, M., 2005. Logic Programming with Prolog, Springer.

Brassel, K., Bucher, F., Stephan, E.-M. and Vckovski, A., 1995. Elements of Spatial Data Quality, Completeness, Elsevier, 81-108.

Bravo, L. and Rodriguez, M. A., 2012. Formalization and Reasoning About Spatial Semantic Integrity Constraints, Data & Knowledge Engineering, 72, 63-82.

Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M. and Rosati, R., 2009. Ontologies and Databases: The DL-Lite Approach, Reasoning Web International Summer School, Brixen-Bressanone, Italy: 255-356.

Canada, G., 1997, Standards and Specifications of the National Topographic Data Base, Geomatics Canada, Canada.

CEN, 2013 CEN/TC 287, Geographic Information, European Committee for Standardization.

Ceri, S., Gottlob, G. and Tanca, L., 1989. What You Always Wanted to Know About Datalog (and Never Dared to Ask), IEEE Transactions on Knowledge and Data Engineering, 1,1, 146-166.

Cherfi, S., Hamdi, F., Rigaux, P., Thion, V. and Travers, N., 2017. Formalizing Quality Rules on Music Notation. An Ontology-Based Approach., International Conference on Technologies for Music Notation and Representation, University of A Coruña, A Coruña, Spain: 91-97.

Clarke, D. G. and Clark, D. M., 1995. Elements of Spatial Data Quality, Lineage, Elsevier, 13-30.

Clementini, E., Di Felice, P. and Van Oosterom, P., 1993. A Small Set of Formal Topological Relationships Suitable for End-User Interaction, The Fourth International Symposium on Spatial Databases, Portland, OR, USA: 277-295.

Clocksin, W. F. and Mellish, C. S., 2012. Programming in Prolog: Using the Iso Standard, Springer Science & Business Media.

Cohn, A. G., Bennett, B., Gooday, J. and Gotts, N. M., 1997. Qualitative Spatial Representation and Reasoning with the Region Connection Calculus, GeoInformatica, 1,3, 275-316.

Costa, V. S., Rocha, R. and Damas, L., 2012. The Yap Prolog System, Theory and Practice of Logic Programming, 12,1-2, 5-34.

Dea, C., 2012. Javafx 2.0: Introduction by Example, Apress.

Debattista, J., Auer, S. and Lange, C., 2016. Luzzu - a Framework for Linked Data Quality Assessment, IEEE Tenth International Conference on Semantic Computing, Laguna Hills, CA, USA: 124-131.

Debattista, J., Lange, C. and Auer, S., 2014. Representing Dataset Quality Metadata Using Multi-Dimensional Views, Proceedings of the 10th International Conference on Semantic Systems: 92-99.

Debattista, J., Lange, C. and Auer, S., 2015. Luzzu Quality Metric Language - a Dsl for Linked Data Quality Assessment, Computing Research Repository.

Degbelo, A., 2012. An Ontology Design Pattern for Spatial Data Quality Characterization in the Semantic Sensor Web, The Fifth International Conference on Semantic Sensor Networks, Boston, Massachusetts, USA, 904: 103-108.

Dentler, K., Cornet, R., Ten Teije, A. and De Keizer, N., 2011. Comparison of Reasoners for Large Ontologies in the OWL 2 El Profile, Semantic Web, 2,2, 71-87.

Devillers, R. and Jeansoulin, R., 2013. Fundamentals_of Spatial Data Quality.

Drummond, J., 1995. Elements of Spatial Data Quality, Positional Accuracy, Elsevier, 31-58.

Drummond, N. and Shearer, R., 2006. The Open World Assumption, eSI Workshop: The Closed World of Databases meets the Open World of the Semantic Web, Vienna, Austria, 15.

Dürst, M. and Suignard, M. 2004. Internationalized Resource Identifiers (IRIs).

Egenhofer, M. J. and Herring, J., 1990. A Mathematical Framework for the Definition of Topological Relationships, The Fourth International Symposium on Spatial Data Handling, Zurich, Switzerland: 803-813.

Enderton, H. B., 2015. The Stanford Encyclopedia of Philosophy, Second-Order and Higher-Order Logic, Metaphysics Research Lab, Stanford University, Stanford, CA, USA.

Fonte, C., Antoniou, V., Bastin, L., Estima, J., Arsanjani, J., Bayas, J., See, L. and Vatseva, R., 2017. Assessing Vgi Data Quality, Mapping and the Citizen Sensor, 137-163.

Frank, A. U., 1997. Spatial and Temporal Reasoning, Spatial Ontology: A Geographical Information Point of View, Springer, 135-153.

Fürber, C., 2015. Data Quality Management with Semantic Technologies, Springer.

Fürber, C. and Hepp, M., 2010. Using SPARQL and SPIN for Data Quality Management on the Semantic Web 1 Introduction, The Thirteenth International Conference on Business Information Systems, 47: 1--12.

Fürber, C. and Hepp, M., 2011. Towards a Vocabulary for Data Quality Management in Semantic Web Architectures, The First International Workshop on Linked Web Data Management, Bonn, Germany: 1-8.

Geisler, S., Quix, C., Weber, S. and Jarke, M., 2016. Ontology Based Data Quality Management for Data Streams, Journal of Data and Information Quality, 7,4, 18.

Goodchild, M. F., 1995. Elements of Spatial Data Quality, Attribute Accuracy, 59-79.

Goodchild, M. F., 2007. Citizens as Voluntary Sensors: Spatial Data Infrastructure in the World of Web 2.0, International Journal of Spatial Data Infrastructures Research, 2, 24-32.

Goodwin, J., 2005. Experiences of Using OWL at the Ordnance Survey, The First OWL:Experiences and Directions Workshop, Galway, Ireland, 188.

Goodwin, J., Dolbear, C. and Hart, G., 2008. Geographical Linked Data: The Administrative Geography of Great Britain on the Semantic Web, Transactions in GIS, 12,s1, 19-30.

Gouveia, C., Henriques, P., Nicolau, R., Rocha, J. and Santos, M., 2001. Moving from Cen TC 287 to Iso/TC 211-the Approach of the Portuguese National Geographic Information Infrastructure, Proceedings of the 4th AGILE Conference, 260: 269.

Gruber, T. R., 1995. Toward Principles for the Design of Ontologies Used for Knowledge Sharing?, International Journal of Human-Computer Studies, 43,5-6, 907-928.

Guha, R., McCool, R. and Miller, E., 2003. Semantic Search, The Twelfth International Conference on World Wide Web, Budapest, Hungary: 700-709.

Guptill, S. C. and Morrison, J. L., 2013. Elements of Spatial Data Quality, Elsevier.

Hamish, B., 2012. The Four V's of Big Data, Implementing Information Infrastructure Symposium, Singapore.

Hebeler, J., Fisher, M., Blace, R. and Perez-Lopez, A., 2011. Semantic Web Programming, John Wiley & Sons.

Henriksson, R. and Kauppinen, T., 2007. An Ontology Driven Approach for Spatial Data Quality Evaluation, The Fifth International Symposium on Spatial Data Quality, Enschede, NL.

Herzog, T. N., Scheuren, F. J. and Winkler, W. E., 2007. Data Quality and Record Linkage Techniques, Springer Science & Business Media.

Hilbert, D. and Ackermann, W., 1999. Principles of Mathematical Logic, American Mathematical Soc.

Hitzler, P., Krotzsch, M. and Rudolph, S., 2009. Foundations of Semantic Web Technologies, CRC Press.

Horn, A., 1951. On Sentences Which Are True of Direct Unions of Algebras, The Journal of Symbolic Logic, 16,1, 14-21.

Horridge, M. and Bechhofer, S., 2011. The OWL API: A Java API for OWL Ontologies, Semantic Web, 2,1, 11-21.

Horridge, M., Drummond, N., Goodwin, J., Rector, A. L., Stevens, R. and Wang, H., 2006. The Manchester OWL Syntax, OWLed, 216.

Horrocks, I., 2002. Description Logic: Axioms and Rules, European Conference on Artificial Intelligence.

Horrocks, I., Kutz, O. and Sattler, U., 2006. The Even More Irresistible SROIQ, The Tenth International Conference on the Principles of Knowledge Representation and Reasoning, The Lake District, UK, 6: 57-67.

Hustadt, U., 1994. Do We Need the Closed World Assumption in Knowledge Representation?, The First Knowledge Representation Meets Databases Workshop, Saarbrücken, Germany.

Hustadt, U., Motik, B. and Sattler, U., 2004. Reducing Shiq-Description Logic to Disjunctive Datalog Programs, KR, 2004, 152-162.

ICA, 1995. Elements of Spatial Data Quality, The Seventeenth International Cartographic Association Conference, Barcelona, Spain.

INSPIRE, 2014a D2.8.I.6, Data Specification on Cadastral Parcels - Technical Guidelines, European Commission Joint Research Centre.

INSPIRE, 2014b D2.8.I.8, Data Specification on Hydrography - Technical Guidelines, European Commission Joint Research Centre.

INSPIRE, 2014c D2.8.I.7, Data Specification on Transport Networks - Technical Guidelines, European Commission Joint Research Centre.

ISO, 1986 8402, Quality Management and Quality Assurance - Vocabulary, International Organization for Standards, USA.

ISO, 2002 19113, Geographic Information-Quality Principles, International Standardization Organization, Sweden.

ISO, 2003 19114, Geographic Information – Quality Evaluation Procedures, International Standardization Organization, Sweden.

ISO, 2005 19138, Geographic Information – Data Quality Measures, International Standardization Organization, Sweden.

ISO, 2013 19157, Geographic Information-Data Quality, International Standardization Organization, Sweden.

Janowicz, K. and Compton, M., 2010. The Stimulus-Sensor-Observation Ontology Design Pattern and Its Integration into the Semantic Sensor Network Ontology, The Third International Workshop on Semantic Sensor Networks, Shaghai, China.

Joos, G., 2006. Data Quality Standards, XXIII International FIG Congress, Munic, Germany.

Juran, J. M., 1974. Quality Control Handbook, McGraw-Hill, New York.

Kainz, W., 1995. Logical Consistency, Elements of Spatial Data Quality, 202, 109-137.

Keßler, C., Raubal, M. and Wosniok, C., 2009. Semantic Rules for Context-Aware Geographical Information Retrieval, The Fourth European Conference on Smart Sensing and Context, Guildford, UK, 5741: 77-92.

Knublauch, H., Hendler, J. and Idehen, K., 2009. Spin Sparql Inferencing Notation, Retrieved Feb, 8, 2013.

Krötzsch, M., Simancik, F. and Horrocks, I., 2012. A Description Logic Primer, Computing Research Repository.

Kyzirakos, K., Vlachopoulos, I., Savva, D., Manegold, S. and Koubarakis, M., 2014. Geotriples: A Tool for Publishing Geospatial Data as RDF Graphs Using R2RML Mappings, The Thirteenth International Semantic Web Conference, Riva Del Garda, Italy, 8796: 393-396.

Ling, T. W., 1990. The Prolog Not-Predicate and Negation as Failure Rule, New Generation Computing, 8,1, 5-31.

Mäs, S., Wang, F. and Reinhardt, W., 2005. Using Ontologies for Integrity Constraint Definition, The Fourth international symposium on spatial data quality, Beijing, China, 5: 25-26.

McBride, B., 2001. Jena: Implementing the Rdf Model and Syntax Specification, The Second International Conference on Semantic Web, Hong Kong, 40: 23-28.

Mostafavi, M.-A., Edwards, G. and Jeansoulin, R., 2004. An Ontology Based Method for Quality Assessment of Spatial Data Bases, The Third International Symposium on Spatial Data Quality, Bruck an der Leitha, Austria, 1: 49-66.

Nash, E., Nikkilä, R., Wiebensohn, J., Walter, K. and Bill, R., 2011. Interchange of Geospatial Rules–Towards Georules Interchange Format (Georif), GIS Science, 24,3, 82-94.

Nash, E., Wiebensohn, J., Nikkilä, R., Vatsanidou, A., Fountas, S. and Bill, R., 2011. Towards Automated Compliance Checking Based on a Formal Representation of Agricultural Production Standards, Computers and Electronics in Agriculture, 78,1, 28-37.

NCDCDS, 1987, A Draft Proposed Standard for Digital Cartographic Data, National Committee for Digital Cartographic Data Standards, Columbus, OH.

Neng, W., Zhongliang, D. and Guangyong, Y., 2016. Representation and Reasoning of Topological Relations between Enclave and Exclave Regions, International Conference on Computational Science and Its Applications: 301-311.

Nyulas, C., O'Connor, M. and Tu, S., 2007. Datamaster - a Plug-in for Importing Schemas and Data from Relational Databases into Protégé, The Tenth International Protégé Conference, Budapest, Hungary: 15-18.

O'Connor, M. J. and Das, A. K., 2009. SQWRL: A Query Language for OWL, The Sixth OWL:Experiences and Directions Workshop, Chantilly, Virginia, USA, 529.

OGC, 1999 99-049, Simple Features Specifications for Sql, Revision 1.1., Open GIS Consortium, Inc.

OGC, 2012 11-052r4, Geosparql-a Geographic Query Language for RDF Data, Open GIS Consortium, Inc.

Parsia, B. and Sirin, E., 2004. Pellet: An OWL DL Reasoner, The Third International Semantic Web Conference, Hiroshima, Japan, 18: 2.

Patroumpas, K., Alexakis, M., Giannopoulos, G. and Athanasiou, S., 2014. Triplegeo: An ETL Tool for Transforming Geospatial Data into RDF Triples, EDBT/ICDT 2014 Joint Conference, Athens, Greece: 275-278.

Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M. and Rosati, R., 2008. Journal on Data Semantics X, Linking Data to Ontologies, 4900, Springer, 133-173.

Randell, D. A., Cui, Z. and Cohn, A. G., 1992. A Spatial Logic Based on Regions and Connection.

Rattanasawad, T., Buranarach, M., Saikaew, K. R. and Supnithi, T., 2018. A Comparative Study of Rule-Based Inference Engines for the Semantic Web, IEICE TRANSACTIONS on Information and Systems, 101,1, 82-89.

Reiter, R., 1981. Readings in Artificial Intelligence, On Closed World Data Bases, Elsevier, 119-140.

Rosen, K. H., 2011. Discrete Mathematics and Its Applications, McGraw-Hill, New York.

Salgé, F., 1995. Elements of Spatial Data Quality, Semantic Accuracy, 139-151.

Sanderson, M., Ramage, S. and Van Linden, L., 2009. Sdi Communities: Data Quality and Knowledge Sharing, The Eleventh Global Spatial Data Infrastructure Association Conference, Rotterdam

Sandgren, U., 2009. Inspire Specification for Transport Networks, The Sixteenth Intelligent Transport Systems World Congress and Exhibition, Stockholm, Sweden.

Santos Costa, V., Damas, L., Reis, R. and Azevedo, R., 2002. Yap User's Manual, Universidade do Porto, Porto, Portugal.

Saunders, S., Fields, D. K. and Belayev, E., 2006. Intellij Idea in Action, Manning.

Schildt, H., 2014. Java: The Complete Reference, McGraw-Hill Education Group.

See, L., Estima, J., Pődör, A., Arsanjani, J. J., Bayas, J.-C. L. and Vatseva, R., 2017. Sources of Vgi for Mapping, Mapping and the Citizen Sensor, 13.

Segal, M., Akeley, K., Frazier, C., Leech, J. and Brown, P., 2010. The Opengl Graphics System: A Specification, Silicon Graphics, Inc., Mountain View, CA.

Servigne, S., Lesage, N. and Libourel, T., 2006. Fundamentals of Spatial Data Quality, Quality Components, Standards, and Metadata, 179-210.

Servigne, S., Ubeda, T., Puricelli, A. and Laurini, R., 2000. A Methodology for Spatial Consistency Improvement of Geographic Databases, GeoInformatica, 4,1, 7-34.

Sicilia, M.-A. and Lytras, M. D., 2008. Metadata and Semantics, Springer Science & Business Media.

Sirin, E., 2010. Data Validation with OWL Integrity Constraints, International Conference on Web Reasoning and Rule Systems: 18-22.

Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A. and Katz, Y., 2007. Pellet: A Practical OWL-DL Reasoner, Web Semantics, 5,2, 51-53.

Sirin, E., Smith, M. and Wallace, E., 2008. Opening, Closing Worlds-on Integrity Constraints, The Fifth OWL:Experiences and Directions Workshop, Karlsruhe, Germany.

Sirin, E. and Tao, J., 2009. Towards Integrity Constraints in OWL, The Sixth International Conference on OWL: Experiences and Directions, Chantilly, Virginia, USA: 79-88.

Skillen, K.-L., Chen, L., Nugent, C., Donnelly, M., Burns, W. and Solheim, I., 2013. Using SWRL and Ontological Reasoning for the Personalization of Context-Aware Assistive Services, The Sixth International Conference on PErvasive Technologies Related to Assistive Environments, Rhodes, Greece: 48.

Spaccapietra, S., Cullot, N., Parent, C. and Vangenot, C., 2004. On Spatial Ontologies, *GeoInfo 2004*.

Srinivasan, A. and Richards, J. A., 1993. Analysis of GIS Spatial Data Using Knowledge-Based Methods, International Journal of Geographical Information Science, 7,6, 479-500.

Sterling, L. and Shapiro, E. Y., 1994. The Art of Prolog: Advanced Programming Techniques, MIT press.

Stocker, M. and Sirin, E., 2009. Pelletspatial: A Hybrid RCC-8 and RDF/OWL Reasoning and Query Engine, The Sixth OWL:Experiences and Directions Workshop, Chantilly, Virginia, USA, 529.

Strong, D. M., Lee, Y. W. and Wang, R. Y., 1997. Data Quality in Context, Communications of the ACM, 40,5, 103-110.

Tao, J., Sirin, E., Bao, J. and McGuinness, D. L., 2010. Integrity Constraints in OWL, The Twentyfourth National Conference on Artificial Intelligence, Atlanta, GA, USA.

Tomaszuk, D., 2017. RDF Validation: A Brief Survey, The Thirteenth International Conference: Beyond Databases, Architectures and Structures, Ustroń, Poland: 344-355.

Tóth, K., Tomas, R., de Lima, V. N. and Cetl, V. 2013. Data Quality in INSPIRE: Balancing Legal Obligations with Technical Aspects.

URL-1, http://desktop.arcgis.com/en/arcmap/latest/extensions/data-reviewer/checks-in-data-reviewer.htm Checks in Data Reviewer. 01/05/2018.

URL-2, https://www.w3.org/2000/10/swap/Primer.html Primer: Getting into RDF & Semantic Web Using N3. 01/05/2018.

URL-3, https://www.w3.org/TR/owl-features/ OWL Web Ontology Language. 01/05/2018.

URL-4, https://www.w3.org/DesignIssues/RDB-RDF.html Relational Databases on the Semantic Web. 01/05/2018.

URL-5, https://locationtech.github.io/jts/ Java Topology Suite. 01/05/2018.

URL-6, https://trac.osgeo.org/geos/ Geos - Geometry Engine, Open Source. 01/05/2018.

URL-7, https://bitbucket.org/rcantonialli/geoswrl/ Geoswrl Builtins. 01/05/2018.

URL-8, http://virtuoso.openlinksw.com/ Openlink Virtuoso. 01/05/2018.

URL-9, http://mayor2.dia.fi.upm.es/oeg-upm/index.php/en/technologies/151-geometry2rdf/index.html Geometry2rdf. 01/05/2018.

URL-10, http://www.geonames.org/ontology/documentation.html Geonames Ontology. 01/05/2018.

URL-11, https://www.w3.org/2005/Incubator/geo/XGR-geo-20071023/#owl W3C Geospatial Vocabulary, W3C Incubator Group Report. May.

URL-12, http://spinrdf.org/spin-shacl.html From SPIN to SHACL. 01/05/2018.

URL-13, https://www.w3.org/TR/shacl-js/ SHACL Javascript Extensions. 01/05/2018.

Vallières, S., Brodeur, J. and Pilon, D., 2005. Fundamentals of Spatial Data Quality, Spatial Integrity Constraints: A Tool for Improving the Internal Quality of Spatial Data, 161-178.

Van Emden, M. H. and Kowalski, R. A., 1976. The Semantics of Predicate Logic as a Programming Language, Journal of the ACM, 23,4, 733-742.

Vaz, D., Costa, V. S. and Ferreira, M., 2009. User Defined Indexing, The Twentyfifth International Conference on Logic Programming, Pasadena, CA, USA: 372-386.

Vaz, D., Costa, V. S. and Ferreira, M., 2010. Fire! Firing Inductive Rules from Economic Geography for Fire Risk Detection, The Twentieth International Conference on Inductive Logic Programming, Florance, Italy: 238-252.

Vaz, D., Ferreira, M. and Lopes, R., 2007. Spatial-Yap: A Logic-Based Geographic Information System, The Twentythird International Conference on Logic Programming, Porto, Portugal: 195-208.

Veregin, H., 1999. Data Quality Parameters, Geographical Information Systems, 1, 177-189.

W3C, 2004a, RDF Primer, World Wide Web Consortium.

W3C, 2004b, RDF/XML Syntax Specification, World Wide Web Consortium.

W3C, 2004c, Resource Description Framework (RDF): Concepts and Abstract Syntax, World Wide Web Consortium.

W3C, 2004d, SWRL: A Semantic Web Rule Language Combining OWL and Ruleml, World Wide Web Consortium.

W3C, 2008a, RDFa in Xhtml: Syntax and Processing, World Wide Web Consortium.

W3C, 2008b, SPARQL Query Language for RDF, World Wide Web Consortium.

W3C, 2011a, Notation3 (N3): A Readable RDF Syntax, World Wide Web Consortium.

W3C, 2011b, Spin-Modeling Vocabulary, World Wide Web Consortium.

W3C, 2011c, Spin-SPARQL Syntax, World Wide Web Consortium.

W3C, 2011d, SPIN - Overview and Motivation, World Wide Web Consortium.

W3C, 2012a, OWL 2 Web Ontology Language, World Wide Web Consortium.

W3C, 2012b, R2RML: Rdb to RDF Mapping Language, World Wide Web Consortium.

W3C, 2013, RIF Basic Logic Dialect, World Wide Web Consortium.

W3C, 2014a, RDF 1.1 N-Triples: A Line-Based Syntax for an RDF Graph, World Wide Web Consortium.

W3C, 2014b, RDF 1.1 Turtle- Terse RDF Triple Language, World Wide Web Consortium.

W3C, 2014c, RDF Schema 1.1, World Wide Web Consortium.

W3C, 2015, RDFa Lite 1.1, World Wide Web Consortium.

W3C, 2017, Shapes Constraint Language (SHACL), World Wide Web Consortium.

Wand, Y. and Wang, R. Y., 1996. Anchoring Data Quality Dimensions in Ontological Foundations, Communications of the ACM, 39,11, 86-95.

Wang, F., Mäs, S., Reinhardt, W. and Kandawasvika, A., 2005. Ontology Based Quality Assurance for Mobile Data Acquisition, 19th International Conference on Informatics for Environmental Protection, Brno, Czech Republic.

Watson, P., 2007. Formal Languages for Expressing Spatial Data Constraints and Implications for Reporting of Quality Metadata, The Fifth International Symposium on Spatial Data Quality, Enschede, NL, 7.

Woodsford, P., 2007. Spatial Database Update and Validation — Current Advances in Theory and Technology, ISPRS Workshop on Updating Geo-spatial Databases with Imagery & The 5th ISPRS Workshop on Dynamic and Multi-dimensional GIS, Urumchi, China, 4: W54.

Yang, Z. and Jiang, M., 2007. Using Eclipse as a Tool-Integration Platform for Software Development, IEEE Software,2, 87-89.

Zaveri, A., Rula, A., Maurino, A., Pietrobon, R., Lehmann, J. and Auer, S., 2016. Quality Assessment for Linked Data: A Survey, Semantic Web, 7,1, 63-93.

Zhong Hu, Y., Zhong, B., Luo, H. and Meng Hu, H., 2013. The Regulation Constraint Modeling and Semantic Inferring in Construction Quality Checking, Structural Survey, 31,5, 387-400.

Zhu, L., 2014. SemDQ: A Semantic Framework for Data Quality Assessment, University of Waterloo.

CURRICULUM VITAE

She is born in Trabzon in 1984. After Yunus Emre Anatolian High School, she joined Karadeniz Technical University, Department of Geomatics Engineering, where she graduated in 2006 and started her post-graduate study. She completed her Master's degree in 2009 in Cartography division. She started PhD in the same division in 2010. She knows English.