Lund University GEM thesis series nr 25

Development of a digitalization tool for linking thematic data to a background map

Haiqi Xu

2017

Department of Physical Geography and Ecosystem Science

Lund University

Sölvegatan 12

S-223 62 Lund

LUND UNIVERSITY

UNIVERSITY OF TWENTE.

FACULTY OF GEO-INFORMATION SCIENCE AND EARTH OBSERVATION

Development of a digitalization tool for linking thematic data to a background map

by

Haiqi Xu

Thesis submitted to the department of Physical Geography and Ecosystem Science, Lund University, in partial fulfilment of the requirements for the degree of Master of Science in Geo-information Science and Earth Observation for Environmental Modelling and Management

Thesis assessment Board

First Supervisor: *Senior lecturer, Lars Harrie* (Lund University)
Co-supervisors: *Dr, Ehsan Abdolmajidi* (Lund University)
Co-supervisors: *Dr, Weiming Huang* (Lund University)

Exam committee:
Dr. Ali Mansourian (Lund University)
Dr, Per-Ola Olsson (Lund University)

Disclaimer

This document describes work undertaken as part of a program of study at the University of Lund. All views and opinions expressed therein remain the sole responsibility of the author, and do not necessarily represent those of the institute.

Course title: Geo-information Science and Earth Observation for Environmental Modelling and Management (GEM)

Level: Master of Science (MSc)

Course duration: January 2017 until June 2017

Consortium partners:

The GEM master program is a cooperation of departments at 5 different universities:
University of Twente, ITC (The Netherlands)
University of Lund (Sweden)
University of Southampton (UK)
University of Warsaw (Poland)
University of Iceland (Iceland)

Abstract

Cartographic mashups present geographic data from various sources. A typical example is adding thematic data on top of a multi-scales background map. Currently, thematic data and multi-scales background data are not linked together in cartographic mashups which may have potential problems such as inconsistency of scale ranges. However, in some cases, thematic data and background data have some identical geometries. By creating linkage via these common parts, thematic data can be linked to background map.

The Sematic Web is a Web 3.0 technology through standards by the World Wide Web Consortium (W3C). It provides techniques that enable people to create and link data in standard data formats by semantically describing interrelations of datasets. The linked data can be shared and reused across application, enterprise, and community boundaries.

This study develops a digitalization tool to create and link thematic data to background data in a cartographic mashup. The tool provides two ways to create geometries of thematic data: (1) using common geometries from background map and (2) generating new geometries. Then, thematic data are linked to background data by applying the Semantic Web technologies in the tool. After implementation of the digitalization tool, it is used to create and link thematic data to background map in a use case. Finally, the tool is evaluated against some predefined requirement specifications. It turns out that the digitalization tool manages to create and link thematic data with a background map.

With the digitalization tool developed in this study, thematic data are linked to background map which contributes to synchronize scale ranges between two datasets. It also benefits the real-time data integration of multi-scale data in cartographic mashups.

ACKNOWLEDGEMENTS

I would like to thank Lars Harrie, my first supervisor, for offering the interesting idea that make this study possible and weekly feedback throughout the thesis writing period. His guidance and comments really improved my writing. I appreciate the valuable advises and technical support of my second supervisor, Ehsan Abdolmajid. Plenty of thanks also go to Weiming, my co-supervisor, who not only provided the background data and ontologies in this study, but also inspired and helped me to solve problems in the tool development phase.

To my dearest friends, thank you all for the constant encouragement and accompany.

Finally, I would like to express my gratitude to my parents for the unconditional love and support.

2017-05-23
Haiqi Xu

Contents

1. Introduction 1
   1.1 Background 1
   1.2 Problem statement 1
   1.3 Aim 2
   1.4 Method 3
   1.5 Disposition 3
2. Literature review 4
   2.1 Cartographic mashups 4
   2.2 Multiple representation of geographic data 5
   2.3 Ontology and linked data 6
   2.4 Ontologies for geographic applications 9
   2.5 GeoSPARQL 10
   2.6 Geographic data published as linked data 11
3. Technical development of the digitalization tool 13
   3.1 Background 13
   3.2 Requirement specification of the digitalization tool 15
   3.3 Method 16
   3.4 Implementation 18
4. A use case of the digitalization tool 28
   4.1 Background 28
   4.2 Study area and input data 28
   4.3 User interaction 29
   4.4 Output of the digitalization tool 32
   4.5 Evaluation 34
5. Discussion 36
   5.1 Cartographic mashups 36
   5.2 Multiple representation databases 36
   5.3 Applications of the Sematic Web technologies for geographic data 37

5.4 Prerequisites for the method 37
5.5 The digitalization tool 38
6. Conclusions 39
Reference 40

List of Abbreviations

W3C	World Wide Web Consortium
MRDB	Multiple representation databases
RDF	Resource Description Framework
RDFS	RDF Schema
OWL	Web Ontology Language
URI	Uniform Resource Identifier
HTTP	Hypertext Transfer Protocol
TURTLE	Terse RDF Triple Language
OSM	OpenStreetMap
SQL	Structured Query Language
OGC	Open Geospatial Consortium
WKT	Well Known Text
SDI	Spatial Data Infrastructure
CSW	Catalogue Service for the Web
WFS	Web Feature Service
USGS	U.S. Geological Survey
GML	Geography Markup Language
CRS	Coordinate Reference System

1. Introduction

1.1 Background

A mashup combines widely available tools and data from two or more external sources to create new content *(Merrill 2006)*. A cartographic mashup presents geographic data from multiple datasets, an example is adding thematic data on top of a background map. There is an example of Lonely Planet[1], which gives the public a cartographic representation of tourist sites, restaurants, transport and other information on top of the OpenStreetMap. Other examples of cartographic mashups are the Natural hazards viewer[2] which shows tsunami events and earthquakes above a national geographic map, and city crime map above of a background map[3] and so on.

The current way to create a cartographic mashup is simply adding thematic data on top of the background map. It implies that there is no linkage between the thematic data and the background map which might be from different providers. In some cases, there might be displacements between thematic map and background map. And commonly, thematic data only have one or few scales while background maps have more. And thematic data are unlikely to be produced in several scales which are as same as background maps *(Huang et al. 2016)*. As a result, it could cause inconsistencies when adding thematic data above a background map.

In some applications, there is a possibility to link thematic data to a background map in a cartographic mashup, because a portion of thematic data and background data have identical geometries. By creating linkage via these common parts, thematic data may have same scales as background map, and thus scale ranges between thematic and background data could be synchronized.

The Sematic Web technologies can be used to linking data among different datasets. The Semantic Web is a Web 3.0 technology through standards by the World Wide Web Consortium (W3C). It provides a common framework that created linked data to be shared and reused across application, enterprise, and community boundaries *(W3C Semantic Web Activity Homepage 2017)*.

1.2 Problem statement

A project called Linked Thematic Data at Lund University is studying techniques for improving cartographic mashups. In the project, thematic data are linked to the background map to synchronize scale ranges between them. The background data in multiple scales are already organized in a multiple representation database and interlinked using the Semantic Web

technologies. If the thematic data were connected to the background data, they would have consistent scales.

Therefore, the Linked Thematic Data project requires a digitalization tool to create and link thematic features together with a background map. Figure 1.1 shows an example of the thematic and background data in this project. The green polygon is a nature reserve area (thematic data) presented above a topographic map (background map) which contains road layer (red line), cadastral line layer (black dashed line), lake layer (blue area), etc. Parts of the boundary of the nature reserve can be defined from the geometries of the road and cadastral line, while other parts cannot.

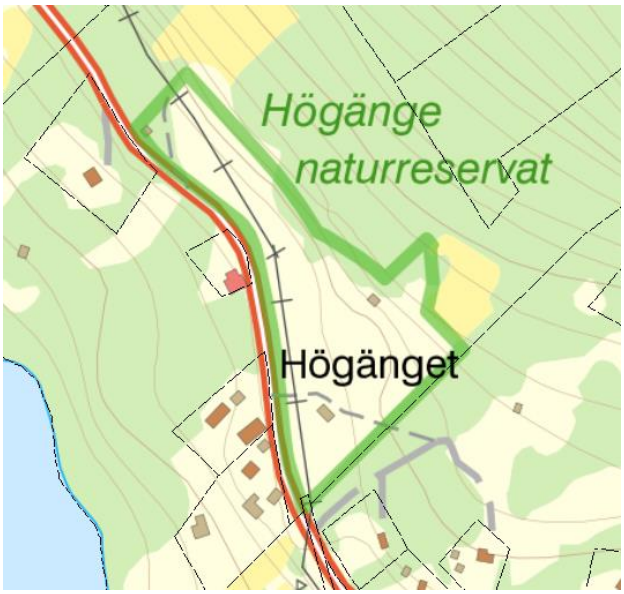


Figure 1.1 Example of data in the Linked Thematic Data Project.

This leads to the first problem of developing the digitalization tool – how to create geometries of thematic data in the digitalization tool. For those thematic data that can be defined by the background features, a cartographer could select a background feature and determine the correct part as a component of a thematic feature. Other thematic data could be created by drawing lines or polygons in the tool.

For the thematic components sharing the background geometries, the second problem is how to link them to their corresponding background features. By semantically describing and saving their relationships with the background features using the Semantic Web technologies, the linkage could be created.

1.3 Aim

The overall aim of this project is to develop a digitalization tool of thematic data that is linked to the background map. The specific aims are:

- The tool should provide two ways to generate the geometries of thematic data: select geometries from a background map and create new geometries.
- Publish the thematic data according to a certain ontology which enables them to be linked to the background map.

1.4 Method

The study starts with a theoretical study of the structures, principles, and application of sematic web technologies. The theoretic part also includes studies related to cartographic mashups and multiple representation of geographic data.

In the practical work, a current method is proposed by the Linked Thematic Data project. The classes and relationships between thematic data and a background map are summarized and defined through the Semantic Web technologies. Then, a digitalization tool combing the method is developed. The requirement specifications of the digitalization tool are specified. Follow the specifications, the tool is developed to create thematic data that are linked to a background map. All the thematic features that users created by the tool would be exported and saved in a standard format which are easy to share and reuse.

After implementation of the digitalization tool, it is applied to create thematic data in a use case. In the use case, six nature reserves in Västernorrland, Sweden, are created and linked to a background map. Finally, the tool is evaluated against the requirement specifications.

1.5 Disposition

The thesis is organized as follows. Chapter 2 contains related work about cartographic mashup development, multiple representations of geographic data, and application of the Semantic Web technologies for geographic data. Chapter 3 describes the background, method, and development of the digitalization tool. Then, the tool is applied in a case study and evaluated in Chapter 4. In chapter 5, the advantages and disadvantages related to methodology and results are discussed. This is followed by the conclusions in chapter 6.

2. Literature review

2.1 Cartographic mashups

The use of cartographic mashups has increased in the last decade with the rapid development of Google Map, Yahoo map, MapQuest APIs, etc. A map mashup combines various data sources and services with added information to create new maps *(Batty et al. 2010)*. *Li and Gong (2008)* summarized map mashup classification, technologies (Asynchronous JavaScript and XML (AJAX), map APIs, etc.), applications(business locations, transportation), and general development steps of mashups. Two map mashups are generated in their study. One is for reporting locations of dead birds related to spread of the West Nile virus on top of Google map. Another shows the locations of public notifications. In addition, technical issues, social problems related to privacy protection and data ownership were also discussed.

Boulos et al. (2008) categorized freely available services and tools of mashup into data formatter, data connector, data visualization, data sharing, and Web APIs and then applied them for four geo mashups to support public health research. The first two mashups simply demonstrated the integration of Web-based infectious disease surveillance data through *Yahoo! Piles* and visualization of the spread of virus in *Google Earth*. Then, the third example mentioned the possibility to contain ontology and other semantic web technologies in geo-mashups. In the last geo-mashup, they proposed some ideas to achieve interactivity and 3D visualization for mapping the latest health news items.

Karnatak et al. (2012) developed a spatial mashup application based on the Web 2.0 technologies for disaster management in the Assam State of India during the floods in 2010. The ground data received from mobile devices, satellite data, OpenStreetMap, and Google Earth, etc. were integrated into the mashup and then contributed to the generation of flood inundation maps and other GIS products. By integrating real-time technologies into a spatial mashup, an intelligence geo-web service can be very helpful for disaster monitoring and decision making.

Toomanian et al. (2013) proposed a method to solve topological and geometrical conflicts of multi-sources spatial data in a viewing service. When using a viewing service via a geoportal, thematic data are overlaid on top of a background map. However, many conflicts may prevent map data to be fully consistent. To adjust thematic data to background map in a viewing service, a system contained rules and relations was created based on semantic labels of data in basic services. After the iteration of rules in a forward chaining approach, it gave the rules for how to adjust the thematic data. According to these rules, an integration method called object refinement was implemented for thematic layers. If each link in thematic layers were sufficiently close to a link in the background map, the geometry of that link in the thematic

layer would be replaced by the geometry in the background map. In this way, the thematic layers can be consistently overlaid on top of a background map in a real-time system. Finally, the methodology was applied in a case study. It turned out that the method could solve any geometrical or topological conflicts between thematic layers and a background map.

The majority of past map mashup researches are based on the Web 2.0 technologies and around the combination of multiple data sources and services. There are only a few studies focusing on linking data layers in cartographic mashups. One example is *Jaara et al. (2011)* who defined a cartographic mashup as some geographic data of different sources, like thematic and background data, which are superimposed and related semantically. They came up with a novel idea which performed generalization on thematic data in a cartographic mashup to conserve and enhance the relationships between thematic and background data. Both semantic and spatial relationships were modeled into 'a reference system' which attaches the thematic data to the background data. By changing the scales and decreasing the level of details of mashup data, the consistency among them was increased to meet the context of use. Unlike building the reference system, this project explores a novel way to store relationships between data layers based on the linked data technologies.

2.2 Multiple representation of geographic data

Multiple representation is a data structure that consists of several representations (e.g. geometric and cartographic representation) of the same geographical object at different scales. In multiple representation databases (MRDBs), the representations are organized as levels as shown in Figure 2.1. The key requirement for an MRDB is that the representations at various scales for the same object are interconnected.

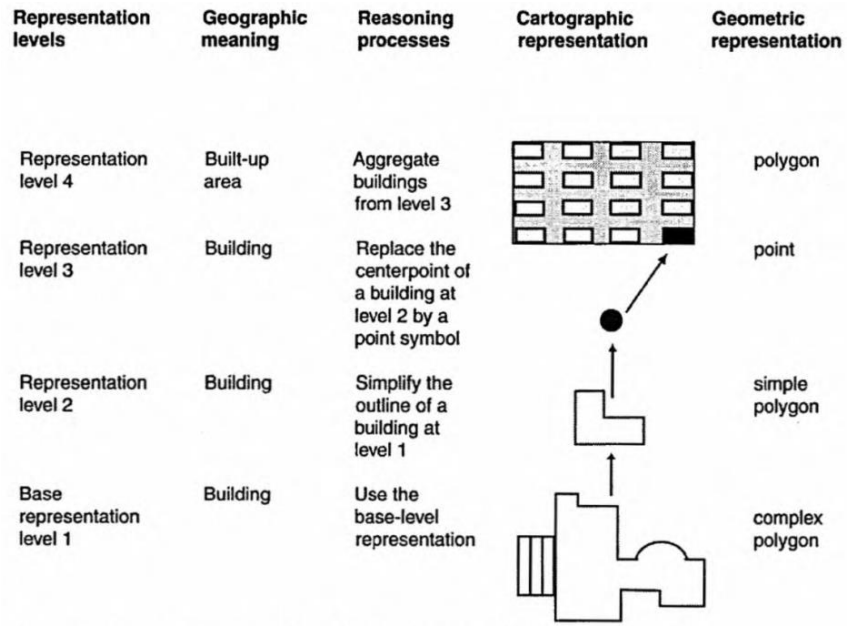


Figure 2.1 Representation level for an object 'building' (Kilpeläinen 2000, pp.104).

Kilpeläinen (2000) proposed an object-oriented approach to implement and manage the connectivity between representation levels for topographic datasets. Compared to cartographic map product databases, MRDBs offered a more flexible way for updating topographic data and performing analysis.

Dunkars (2004) described two different ways to construct an MRDB: one is automatically generalizing the large-scale dataset to create different levels of details, the other is importing separate datasets into different levels and then creating links between them through data matching. In his work, MRDBs were created for two areas in Sweden at three scales where hierarchical relations between datasets and relations between objects representing the same real-world entity are both manually created. However, to create links for nationwide datasets in MRDB, an automated matching procedure and methods were also discussed. Moreover, since cartographic generalization is critical for automatically generalizing large-scale datasets for MRDB, cluster analysis is helpful to understand how cartographic generalization is performed in an MRDB. In the end, he designed a system for automated propagation of updates for roads and buildings in an MRDB.

Bobzien et al. (2008) implemented an MRDB in a cartographic map production system by explicitly modelling three types of relationships: horizontal, vertical, and updates. Horizontal relationships, like partitions, neighborhood, patterns, etc., described the relations between features in a certain scale, while vertical relations created by matching operations were used to link features representing same real-world entities in different scales. The temporal changes of features caused by updating process or data matching were represented by update relations. The current methods for creating these relationships were provided by the authors. After introducing the concepts of relations, a novel approach was proposed to explicitly integrate these relations in a model to produce an enriched MRDB which would enhance support of generalization process, improve data analysis through scales and time, and consistently manage geographic and cartographic data. These three types of relationships can also be applied in generalization services. The establishment of this service-oriented architecture were discussed by *Burghardt et al. (2010)*.

2.3 Ontology and linked data

The Semantic Web technologies such as Ontology, RDF, etc., enable people to create and link data in common data formats which are available on the Web, and build vocabularies and rules for processing data.

2.3.1 Ontology

Ontology can semantically describe and annotate data by defining a number of concepts in a domain and then specify the relationships between them *(Hart and Dolbear 2013)*. Therefore,

the most typical kind of ontology has a taxonomy (define classes of objects and relations among them) and a set of inference rules (rules to manipulate the classes) *(Berners-Lee et al. 2001)*. Ontology is helpful to recognize two terms from different datasets that have common meanings and thus contributes to comparison and integration of databases. Moreover, it is easy to share, reused, and adapt for various situations as they are independent from the data.

Ontologies are usually encoded in RDFS (RDF Schema) or OWL (Web Ontology Language). RDF Schema is a data-modelling vocabulary for RDF data *(Ciobanu et al. 2016)*. For example, if we have a set of classes and property names for topographic domain which has "TopographicFeature", "River", "has X_coordinate", etc. In Ontology, it defines that "TopographicFeature" and "River" are classes, "River" is a subclass of "TopographicFeature", and class "TopographicFeature" has property "has X_coordinate". Then all these statements can be encoded through RDFS which includes standard vocabularies like, "rdfs:Class", "rdfs:subClassOf" and "rdf:Property", etc. The simple reasoning can be achieved in RDFS based on two rules: inheritance and transitivity. Inheritance refers to "a property of a class will be a property of its subclass", while transitivity means that A is a kind of B, B is a kind of C, then A is a kind of C *(Hart and Dolbear 2013).*

OWL, the Web Ontology Language, can express more complex and detailed statements and perform more logical inference than RDFS. OWL has the same basic language constructs as those in RDFS: classes, properties, individuals, and datatypes, where classes are regarded as sets of instances (i.e. individuals) *(Hart and Dolbear 2013)*. For example, in order to describe four scales of the geometries of a map, four sub-properties are defined under a property of '*hasGeometry*' in an OWL file. Figure 2.2 shows the hierarchy of the properties at left side and the OWL file opened in a text editor at right side.

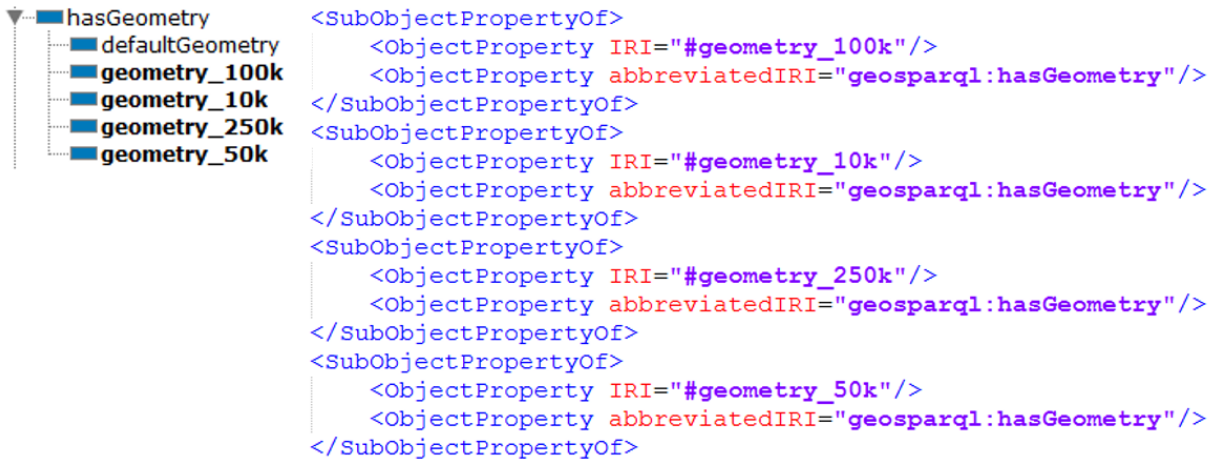


Figure 2.2 Example of an ontology encoded in OWL.

2.3.2 Linked data

Linked data refers to a collection of design principles and technologies around a publishing paradigm to interlink data on the Web *(Kuhn et al. 2014)*. There are four principles of Linked Data stated by *Berners-Lee (2006)* in his Linked Data Design Note:

1. Use Uniform Resource Identifiers (URIs) as names for things.
2. Use Hypertext Transfer Protocol (HTTP) URIs so that people can look up those names.
3. When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL).
4. Include links to other URIs so that they can discover more things.

Linked data concentrate on exposing data to the Web that was hidden in proprietary databases, allowing the data to be discoverable by Web crawlers and processed by different applications. To achieve that, data should be published on the Web in a standard format: RDF, which could provide descriptions of what the information is about and specify links between them *(Hart and Dolbear 2013)*.

RDF (Resource Description Framework) makes the Web or other applications interoperable and exchange information more easily since it is capable of working with, encoding any sort of data, and providing a simple way for users to read, write and query it. The RDF model is a directed graph that uses nodes to represent classes of things or an individual defined within a given Ontology, and edges in the model represent the properties or relations also from this Ontology *(Powell et al. 2015)*. In an RDF, there is a basic structure known as a triple consisting of three elements: a subject, a predicate, and an object to describe a resource which simply is a thing of interest to us. As described by *Heath and Bizer (2011 pp.15)*, "the subject is the URI identifying the described resource. The object can either be a simple literal value, like a string, number, or date; or the URI of another resource that is somehow related to the subject." The predicate also identified by a URI, indicates the relationship between subject and object. Many types of predicate URIs come from vocabularies and collections of URIs that represent hierarchical relations ("is a kind of"), or spatial ("is adjacent to"), and specify an individual as a member of a category ("is an instance of") *(Heath and Bizer 2011)*. Finally, a process called RDF serialization is implemented to encoding an RDF model graph into a data format and then publish it on the Web. Some standard syntaxes are available for developers to choose, like RDF/XML, TURTLE (Terse RDF Triple Language), and RDFa *(Hart and Dolbear 2013)*. TURTLE is the Terse RDF Triple Language being standardized by W3C, which encodes an RDF graph in a compact textual form. It is widely used for quickly reading and writing RDF triples. All the RDF triples written in TURTLE can be indexed by using a standard query language for RDF data called SPARQL *(Beckett and Berners-Lee, 2011).*

2.4 Ontologies for geographic applications

There are two kinds of usages of ontologies discussed in this section: data modelling and process modelling. The first two examples here are related to data modelling and the last one is process modelling. A third use of ontologies is geometrical modelling, which is described in section 2.5.

In the context of the LinkedGeoData project, which aims at providing a rich integrated and interlinked geographic dataset for the Semantic Web, *Stadler et al. (2012)* introduced a tag mapping approach to transform the OpenStreetMap (OSM) data into an RDF data model. A tag mapper comprised a tag pattern that specifies what tags to match, and a transformation function for generating the RDF. After implemented four tap mappers: Resource, Text, Datatype, and Language, relative OSM information would be represented. According to the OSM tags, OWL ontology was derived. In addition, they also interlinked the LinkedGeoData and DBpedia, GeoNames and the Food and Agriculture Organization of the United Nations. Finally, considering OSM data is constantly being updated by its contributors, a live-synchronization module was performed to convert the minutely change of OSM to RDF.

Linked data provides a means for sharing knowledge of a domain, like cities, and thus enable the development of smart city applications. Smart city refers to an intelligent system that combines city-related data and technologies to share information and solve problems for citizens and public administrations. *Consoli et al. (2017)* produced linked data for Catania by constructing a semantic data model and extracting ontologies for the smart city. The data model integrated several data sources, including, geo-referenced data, public transportation, urban fault reporting, road maintenance, and municipal waste collection. In order to ensure the semantic interoperability, the methods, novel ontology patterns, and RDF representation are described in detail.

Gould and Mackaness (2016) described the methodology and steps for building the ontology of a cartographic generalization process that produces road accident maps. The method started with defining a use case to limit the scope of the ontology which is mapping traffic accidents in this case. Then the important concepts for both feature type (e.g. Road Feature, Accident Feature Type, etc.) and generalization model (e.g. operator, algorithm, etc.) were listed in “Information conceptualization” step. Next, the classes and the hierarchy were defined in accordance with the concepts. Finally, the ontology was created in OWL and tested to generate output maps. When the road accident maps remained same after “Crime Feature Type” was added in the ontology, it turned out that explicitly and formally modelling cartographic generalization in an ontology is beneficial to reuse and expansion of it.

2.5 GeoSPARQL

GeoSPARQL is an OGC (Open Geospatial Consortium) standard, which defines a vocabulary for representing and querying geospatial data on the Semantic Web. SPARQL is a W3C (World Wide Web standard for query RDF data. GeoSPARQL extends the SPARQL query language to process geospatial RDF data*(Perry and Herring 2012).*

GeoSPARQL is comprised of:

1. an RDF/OWL vocabulary for representing spatial information
2. a set of functions for spatial calculations
3. a set of query transformation rules.

Some spatial concepts are modeled in GeoSPARQL ontology. As shown in Figure 2.3, *SpatialObject* is the top class of GeoSPARQL, with two direct subclasses of *Feature* and *Geometry*. GeoSPARQL represents geometry objects by dividing them into three class: *Point*, *Curve*, *Surface*. *(Hart and Dolbear 2013)* The property 'geo:hasGeometry' connects a feature with its geometry. Then, 'geo:asWKT' relates a geometry to its WKT (Well Known Text) serialization, while 'geo:asGML' ties a geometry to its information written in GML (Geography Markup Language) *(Perry and Herring 2012)* .

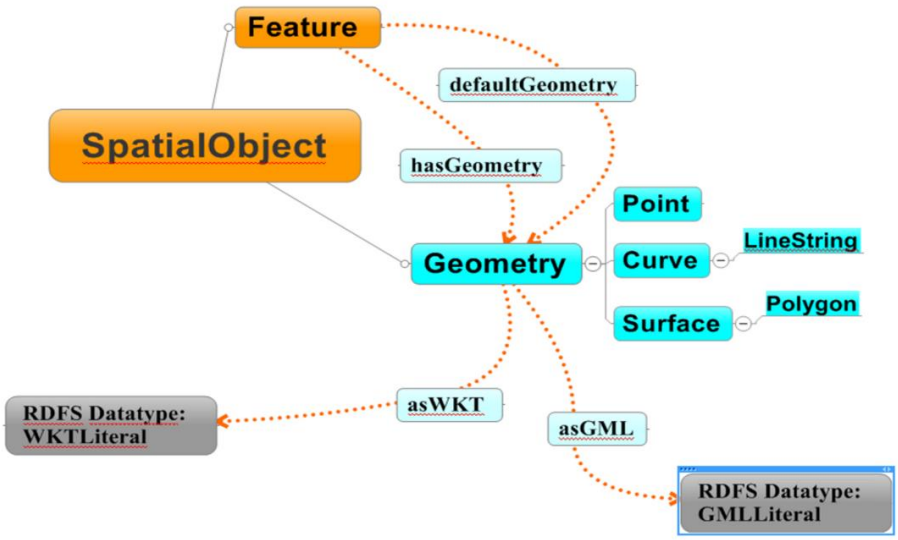


Figure 2.3 Some geometry classes and properties in GeoSPARQL (GeoSPARQL support — GraphDB SE 8.1 documentation, 2017).

A number of vocabularies and functions are available in GeoSPARQL for retrieving geometries and topographic relationships. Other query functions are also available for calculating distance, generating convex hull, buffering objects, etc.

Battle and Kolas (2012) introduced GeoSPARQL and other relevant information like critical spatial concepts, topological relationships, and current state of modelling and query geospatial data in the Semantic Web. They provided some query examples like looking for entities within a particular distance of either other entities or a current location. These

queries are performed to some points of interest such as monuments, parks, restaurants, museums, etc. Topology property 'geo:within', 'geo:hasGeometry' and function 'geof:distance' that calculated the distance were used to answer the query. Then, they updated an open source triple store called *Parliament* that they created before and enabled it to support GeoSPARQL query. Finally, two datasets accessed via *Parliament* were linked together by adding GeoSPARQL to both of them. Thus, spatial distance questions can be proposed for two datasets and combined information can be obtained by GeoSPARQL query.

The INSPIRE Directive are the legal, institutional, and technical standards for interoperable Spatial Data Infrastructures (SDIs) across European member states. Patroumpas et al. (2015) proposed a method to automatically expose INSPIRE-aligned SDIs as linked data based on the GeoSPARQL standard. According to the GeoSPARQL vocabulary, metadata of INSPIRE that describe datasets, dataset series, services and thematic layers across Europe were converted to RDF. As a result, INSPIRE data can be accessed and discovered as linked data. In order to discover INSPIRE data via Catalogue Services, CSWs (Catalogue Services for the Web) were exposed through a CSW-to-RDF middleware which was coined TripleGeo-CSW. TripleGeoCSW is capable of accepting, converting, sending a SPARQL query to all CSW services and returning the combined results as RDF triples. Then the INSPIRE data were transformed to RDF. Finally, a case study was discussed to publishing a real-world INSPIRE SDI that contains geospatial data and metadata for Greece through GeoSPARQL endpoints and an interface for GeoSPARQL query was developed.

2.6 Geographic data published as linked data

Below three examples of publishing different geographic dataset as linked data are described. The first one is found in Finland *(Hietanen et al. 2016)*. In their study, geographic names data from Web Feature Service (WFS) are transformed to the RDF format in a prototype service. The study begins with assigning Uniform Resource Identifiers (URI) to all the spatial objects in the dataset. Then they created an ontology by using GeoSPARQL vocabularies which describe topological relations and different coordinate reference systems. A custom-made data model named UML/GML (Unified Modeling Language) is used in this case to organize the original geographic names datasets. The ontology is designed to improve the UML/GML model in order to understand the linked data principles. Next, a prototype of a Geographic Names as Linked Data Service is created to response to the user requests at client side in real-time. When a request containing URIs of the needed spatial objects is sent to the prototype service, a query is created and sent to the data source – Geographic Names Service (WFS). The output of a WFS is a GML file containing the selected features, which is converted to RDF data in the prototype service and return to client side in an RDF serialization format. Using the WFS guarantees the data provided by the services is up-to-date. This solution enables all the spatial objects to be retrieved from their URIs and facilitates the use of a dataset in different

applications without replicating the whole dataset. The linked data that returns to users can be browsed on the web where search engines can also index all the objects.

The U.S. Geological Survey (USGS) established an ontology for The National Map, converted point, vector and raster data of test areas to RDF, and made the converted data available for download and query using SPARQL to generate graphic output in the form of maps or images. In the process of RDF conversion, the point and vector data were first converted to the Geography Markup Language (GML). Then, the RDF statements were created by following the general approach of defining the subjects, predicates and the objects. In order to implement SPARQL query, coordinates from GML files were connected with the RDF resource. The final output of the SPARQL query would be used to create a map from the coordinates in GML files. Each point feature was converted as a name linking with a location. The RDF statements of vector data semantically describing the information and relations of hydrography, transportation, boundary and structures for the test areas. The traditional approach of converting raster data to RDF is treating whole raster grid as a coverage or extracting vector objects from raster images. In their study, the first step of raster data conversion is identifying features from raster images. For example, a pixel is regarded as a point feature, a linear set of pixels can be used to represent a line feature, and some contiguous or non-contiguous groups of pixels may form a feature that spans areas, like Meteor Crater. In the meantime, the relations between the identifies feature and other neighboring features should be specified as well. Then a term 'ODP' was introduced to provide the definitional characteristics of features that can be used to classify and identify features by similarity matching. Once the features and relations based on the ontology were confirmed, an ODP was matched to a feature and additional attributes and relationships are determined for the feature instance. After the feature instance was connected to the geometric pixel patterns of the raster image, the RDF triples can be defined for the feature. Finally, the USGS developed a server to access the test data converted to RDF and allows direct query of the data using SPARQL. *(Usery and Varanka 2012)*

Shvaiko et al. (2012) published some linked open geo-metadata and geo-data of the Italian Trentino region according to the open government data and the linked open data paradigms. In the 161 geographic datasets, metadata was in XML format and data in shapefiles were transformed to XML. Then geo-metadata were converted to RDF using Dublin Core (DC) 16 and DCMI-BOX17 standard vocabularies. To convert geo-data to RDF, some vocabularies were created to supply the standard vocabularies, such as length, area, perimeter and polyline terms. The objects in point, polyline and polygon in the data are all encoded in the RDF on a web server and can be downloaded from the Trentino geo-portal. Finally, the converted RDF data were manually linked to external vocabularies to ensure the interoperability among different datasets.

3. Technical development of the digitalization tool

3.1 Background

As mentioned in Chapter 1, development of the digitalization tool is a part of the Linked Thematic Data project which aims to link geospatial thematic data with background maps and synchronize scale ranges between them. Figure 3.1 shows the architecture of the Linked Thematic Data project and its relation with the digitalization tool.

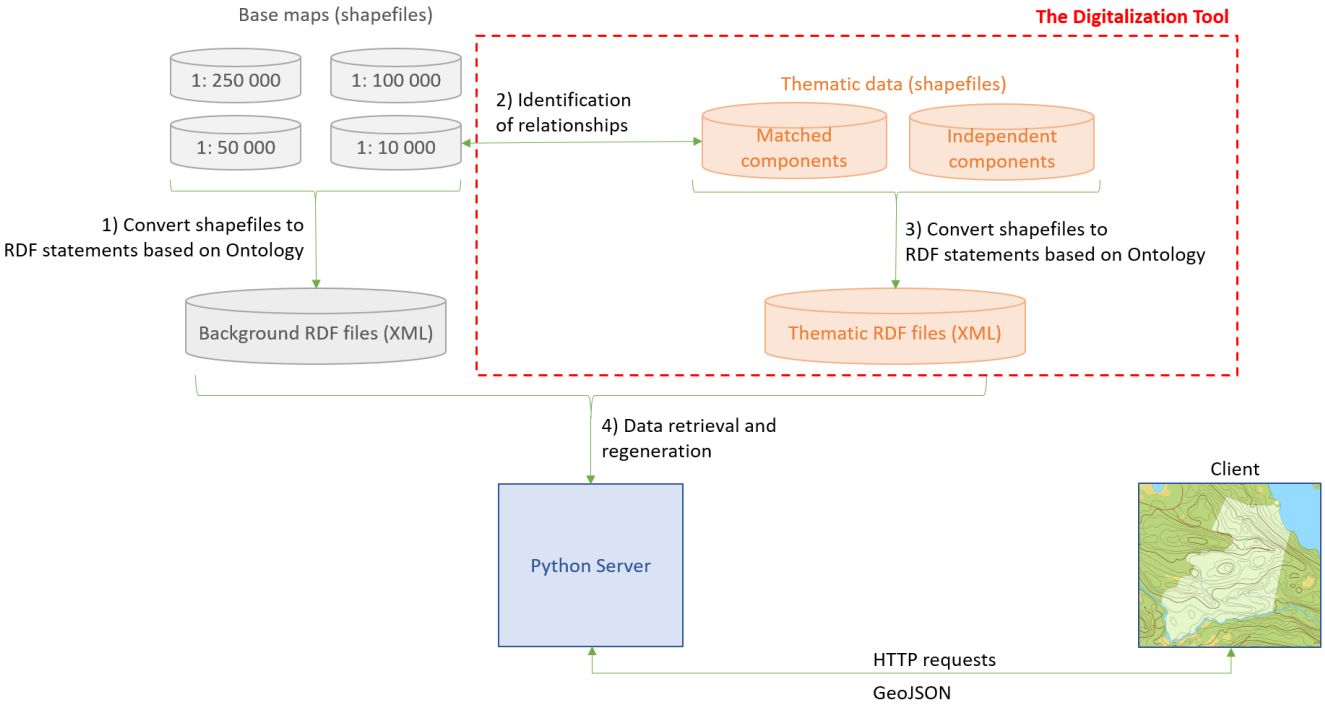


Figure 3.1 Architecture of the Linked Thematic Data project.

The background map has multiple scales which are already interlinked based on an ontology and converted to RDF triples. In order to synchronize the scale ranges, the thematic data should be linked to the background map in the largest scale. As described in section 1.2, there are two kinds of component of thematic features: the matched component is using geometries from the background map, or the independent component is a new line or polygon created by the users. The specific method to connect the matched thematic components with the background map are discussed in section 3.3. After all the thematic features are created, they are converted to RDF statements and exported as a separate RDF file. The python server created in the Linked Thematic Data project can directly read the background RDF files and display them in different scales at client side. For matched thematic components that are selected from the background maps, the server needs to search their counterparts in the background RDF files first and then display the counterparts at the client side. Therefore, these thematic data would change the detail levels along with the

background map when users zoom in/out at client side. However, the independent thematic components only have the most detailed level.

In order to create RDF triples from shapefiles, ontology is required to be designed first. Figure 3.2 shows the hierarchy and relations of the ontology classes and properties created in the Linked Thematic Data project. Both *Feature* and *Geometry* are sub-classes of *Spatial Object*. *Feature* has three subclasses: *Background_Feature*, *Thematic_Feature*, and *Themaitc_Component* which represent the geospatial objects in background and thematic map. The subclasses of *Geometry* are used to describe the geometry types of geographic objects both in thematic and background map.

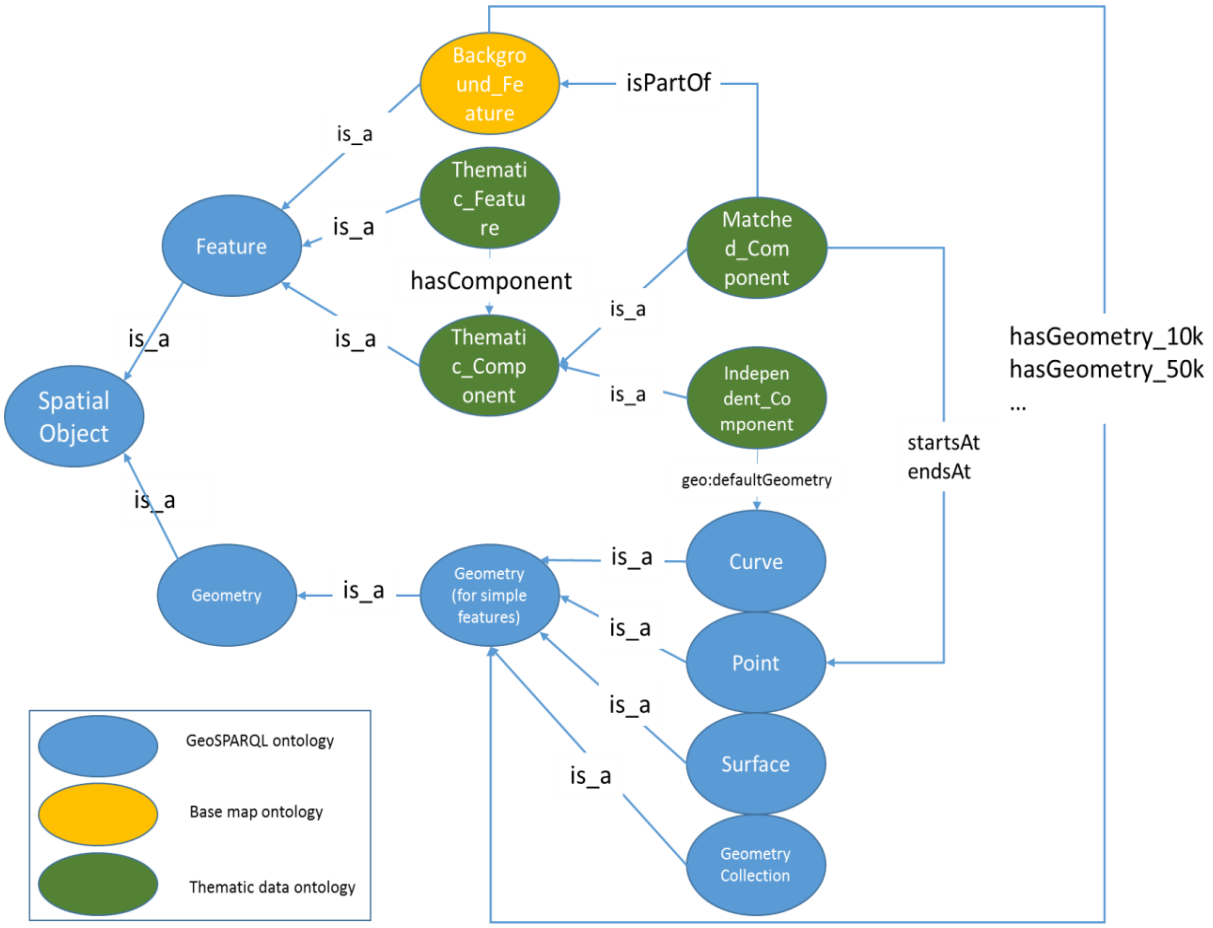


Figure 3.2 Ontologies of the Linked Thematic Data project (Huang 2017).

The Base map ontology describes the background data at four scales. The more detailed explanation about the Base map ontology is shown in Table 3.1. Thematic data ontology describes the relationships between thematic data and background data, and the relationships between thematic feature and thematic component. The Thematic data ontology is explained in section 3.3.

Table 3.1 Descriptions of the Base map ontology.

Ontology classes	Property	Description
Background_Feature	hasGeometry_10k hasGeometry_50k hasGeometry_100k hasGeometry_250k	Describe the background data at four scales, e.g. River I hasGeometry_10k < riverI_10K > River I hasGeometry_50k < riverI_50K > ……

The GeoSPARQL ontology provides the properties of geometry types and other information. The object property *defaultGeometry* is used to save geometry instances for thematic data and background data at a default scale. For example, River II *defaultGeometry* < river II >. If the river II is a line feature, then the < river II > would be an instance of *LineString* which is a sub-property of *Curve* (see Figure 3.2). If the river II is a polygon, then the < river II > would be an instance of *Surface* (see Figure 3.2). Finally, all the coordinates of these instances would be stored as WKT (Well Known Text) literals in RDF. Table 3.2 shows the properties of the GeoSPARQL ontology that are used in this study.

Table 3.2 Descriptions of the GeoSPARQL ontology.

Ontology classes	Property	Description
Geometry	defaultGeometry	Save the coordinate object of a geometry at a default scale.
	Point	Define a geometry type for a point feature.
	LineString	Define a geometry type for a line feature.
	asWKT	All the coordinates of geometries are saved in WKT, e.g. < riverI_10K > *asWKT* < coordinates saved in WKT>

3.2 Requirement specification of the digitalization tool

According to the role of the digitalization tool in the Linked Thematic Data project, the requirement specifications for the tool are:

1. The tool should provide two ways to generate thematic data: select a geometry from the background map (matched thematic component) and create a new geometry (independent thematic component).
2. The background feature should be selectable and highlighted after selection.
3. For a matched thematic component, the tool should allow users to mark a start point and end point on the selected background feature.
4. The tool should save the URI of the selected background feature, coordinates of start point and end point, component order, URI of the matched thematic components, etc. in RDF mainly according to the Thematic data ontology (see Figure 3.2).
5. To create an independent thematic component, the tool should provide a function to draw a line geometry on top of the background map.

6. The new geometry, URI, and component order of the independent components should be organized and stored in RDF mainly according to the Thematic data ontology (see Figure 3.2).
7. The tool should export a separate RDF file after thematic features are accomplished.

3.3 Method

To link thematic data to a background map, Semantic Web technologies such as ontology and RDF, are applied in this case. Figure 3.3 shows the flowchart of the methodology. The first step is to identify all the relationships between thematic components and background features, and thematic components and thematic features. Then, these relationships would be summarized and explicitly defined in an ontology called Thematic data ontology.

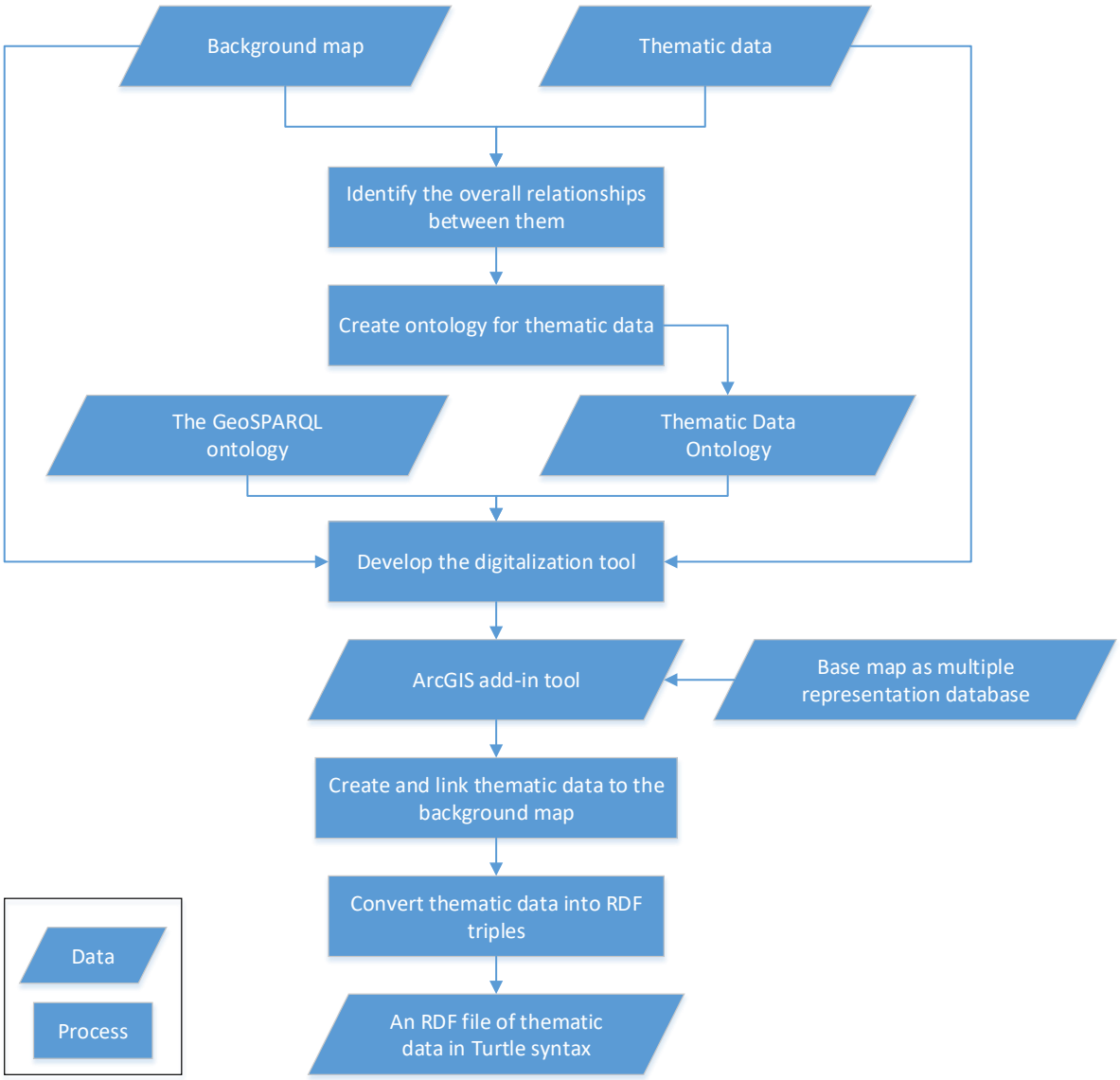


Figure 3.3 Flowchart of linking thematic data to a background map.

The hierarchy of the Thematic data ontology is illustrated in Figure 3.2. There are two classes in the Thematic data ontology: *Thematic_Feature* and *Thematic_Component*. Moreover, according to the two ways of creating thematic components, there are two sub-classes: *Matched_Component* and *Independent_Component* within *Thematic_Component*. As shown in Table 3.3, *'hasComponent'* saves all the thematic components that form a thematic feature. *Matched_Component* is *aPartOf* a background feature represents the relationships between matched components and their corresponding background features; *startsAt, endsAt, and verticesOrder* are used to determine the correct geometrical part of the background features for matched components. The geometry of independent components can be described by '*defaultGeometry*' in the GeoSPARQL ontology (see Table 3.2). Finally, *componentOrder* is used to sequentially access all the thematic components within each thematic feature. Occasionally, there are interior rings within a thematic feature, which are comprised of matched component and independent component as well. In order to distinguish them from the normal component, *InnerRingNo* saves the ID of the interior rings of a thematic feature. These properties could semantically save the relationships and the geometries of all the thematic data.

Table 3.3 Descriptions of the Thematic data ontology.

Ontology classes	Property	Description
Thematic_Feature	hasComponent	List all the thematic segments that compose the current thematic feature.
Thematic_Component	isPartOf	List the corresponding background feature for matched component.
	startsAt	Save the coordinate of start point for matched component.
	endsAt	Save the coordinate of end point for matched component.
	verticesOrder	Represent the vertices direction of matched component is inverse or same as the corresponding background feature.
	componentOrder	Represent the sequence of thematic component.
	InnerRingNo	Save the ID of each interior ring of a thematic feature.

The third step of the method is to develop the digitalization tool. The tool is developed to provide two ways for creating thematic geometries and convert them into RDF statements based on the Thematic data ontology and the GeoSPARQL ontology. The processes of the development are explained in section 3.4. Finally, to evaluate the tool, a use case is designed to add nature reserve areas on top of a topographic map in chapter 4.

3.4 Implementation

ArcGIS[5] is a set of geographic information software for working with geospatial data and maps.

ArcMap[6] is one of the products in ArcGIS, which can create maps, manage and analyze data. Moreover, Python[7] is a high-level programming language which has been incorporated into ArcGIS. Users is able to write Python scripts in ArcGIS to perform data conversion, spatial analysis, etc.

ArcGIS provides a way to extend the functionality of ArcMap by customizing an external Python add-in. Python Add-in Wizard is the application that simplifies the customization and supports many add-in types, such as toolbars, menus, extensions, buttons, combo boxes, tools, etc. In this case, the digitalization tool is developed as a Python add-in of ArcMap.

A Python add-in[8] is a single compressed file with an .esriaddin extension which is easy to share. It is comprised of three items: a config.xml file containing the static add-in properties, a Python script defining the functionalities of the add-in, and resource files that are necessary for the add-in to work, like images.

3.4.1 Start an Add-in Project

To start developing the digitalization tool, the first step is to create a Python add-in project which mainly consists of three steps: new or choose a folder to store the whole project, select a product such as ArcMap to plug in the add-in, and enter the project setting, like name, description, and author of the add-in, etc. Figure 3.4 shows the result after creation of the add-in project.

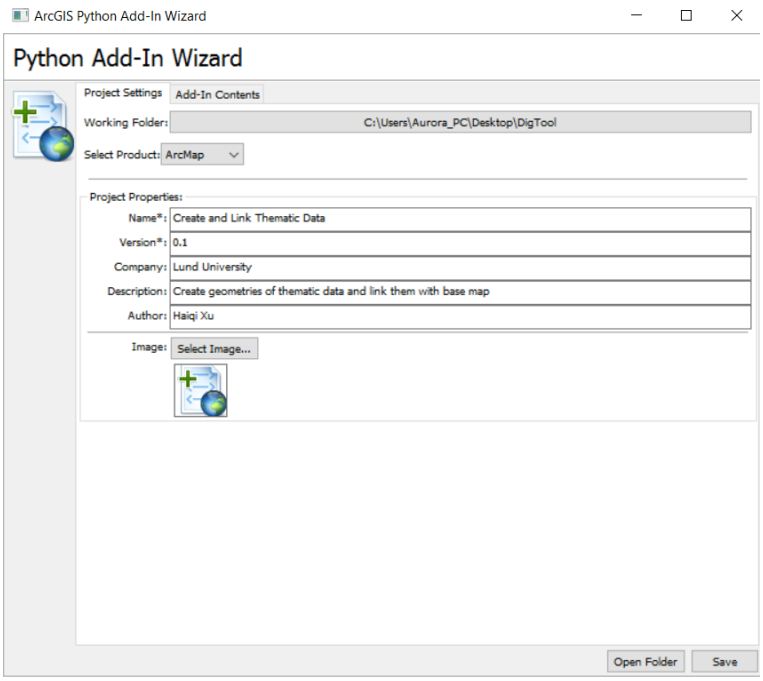


Figure 3.4 Create a Python Add-in project for the digitalization tool.

Next step is to add contents in the Python add-in based on the requirement specifications. The interface of the digitalization tool would be defined by these add-in contents. As shown in Figure 3.5, a toolbar is created as a container for buttons, tools, combo boxes, and menus. The right panel of the wizard shows the properties of the toolbar and each add-in types which allow developers to write the caption, message and help information, and import icons to symbolize add-in types, etc. Table 3.4 elaborates the add-in type and functions of each add-in content. By clicking the Save button after adding all the needed contents, the wizard generates all the required files and folders that are used to support the add-in.

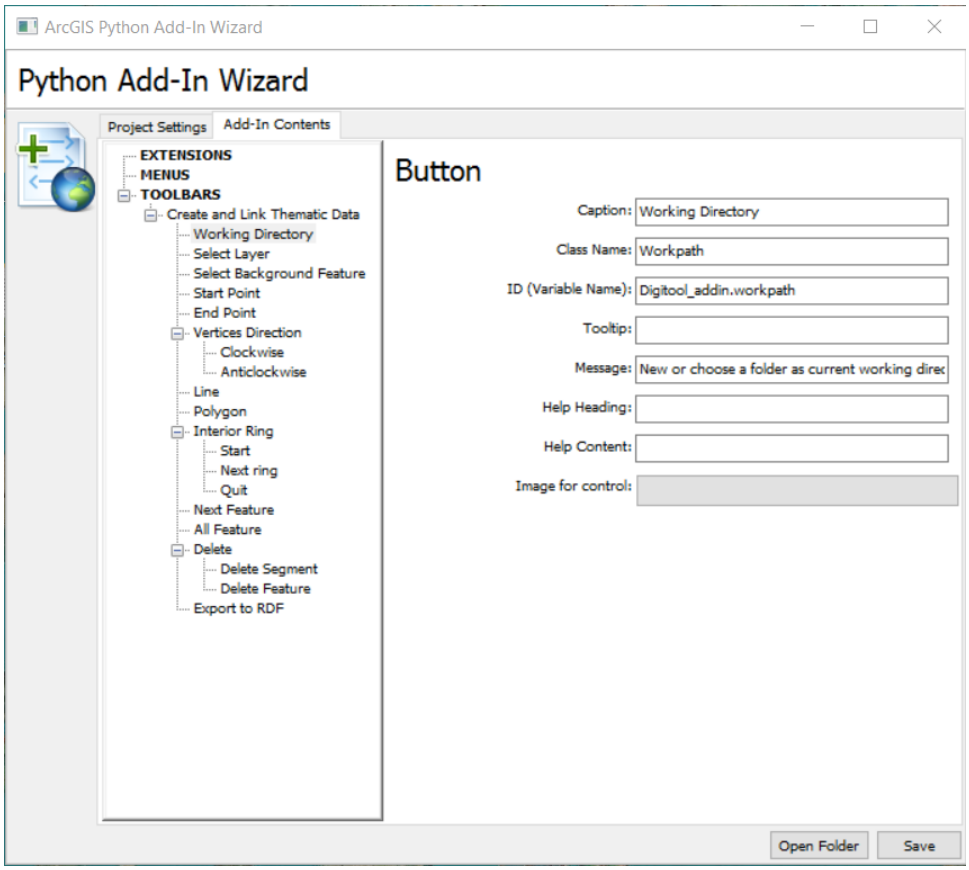


Figure 3.5 Create the Add-in Contents for the digitalization tool.

Table 3.4 Descriptions of all the Add-in Contents for the digitalization tool.

Add-in Content	Add-in Type	Description
Working Directory	Button	New or choose a folder as the working directory.
Select Layer	Combo Box	Select a layer which contains the background feature of interest.
Select Background Feature	Tool	Select the background feature in that layer and read URI of it.
Start Point	Tool	Mark a start point of matched components on the background feature.
End Point	Tool	Mark an end point of matched components on the background feature.
Vertices Direction	Menu	A container for buttons.
Clockwise	Button	Save the vertices direction of the geometry between the start point and end point according to the selected background feature.
Anticlockwise	Button	
Line	Tool	Create a line geometry as a thematic segment.
Polygon	Tool	Create a polygon geometry.
Interior Ring	Menu	A container for buttons.
Start	Button	Start creating an interior ring.
Next Ring	Button	Finish editing the current interior ring and move to the next one.
Quit	Button	Stop creating interior rings.
Next Feature	Button	Finish editing the current thematic feature and move to the next one.
All Feature	Button	Print all the thematic features in Python window.
Delete	Menu	A container for buttons.
Delete Segment	Button	Delete the latest thematic component.
Delete Feature	Button	Delete the latest thematic feature.
Export to RDF	Button	Create RDF triples for all the thematic features and save them in a separate RDF file.

3.4.2 Design a data structure

The data structures are used to store geospatial information and attributes of the thematic data during the creation process. They are designed based on the Thematic data ontology (see Table 3.3), the GeoSPARQL ontology (see Table 3.2) and functionalities of the tool. There are four data structures in this case, called *SelectDictionary*, *LineDictionary*, *Feature*, and *AllFeature*. The hierarchy of the data structure is illustrated in Figure 3.6. *ALLFeature* contains all the thematic features created in the tool. *Feature* saves the thematic components that constitute each thematic feature. *SelectDictionary* is declared for the matched components those are selected from the background map, while *LineDictionary* saves the independent components. A thematic feature may contain both matched and independent components or only one kind of them. *SelectDictionary*, *LineDictionary*, and Feature are Python dictionaries. As shown in Table 3.5, 3.6, 3.7, the keys of each dictionaries are set according to the ontology classes and properties. It is noteworthy that *VDirection* and *InnerRingNum* are two optional keys for the dictionaries. *VDirection* is added to the *SelDictionary* when the selected background feature is a polygon. *InnerRingNum* only works for the components of the interior rings of a thematic feature. *AllFeature* is created as a Python list. Once the cartographers accomplish a thematic feature, it would be appended to *AllFeature*. In the end, all the thematic features are well-organized in the *AllFeature* list and prepared to be converted as RDF triples.

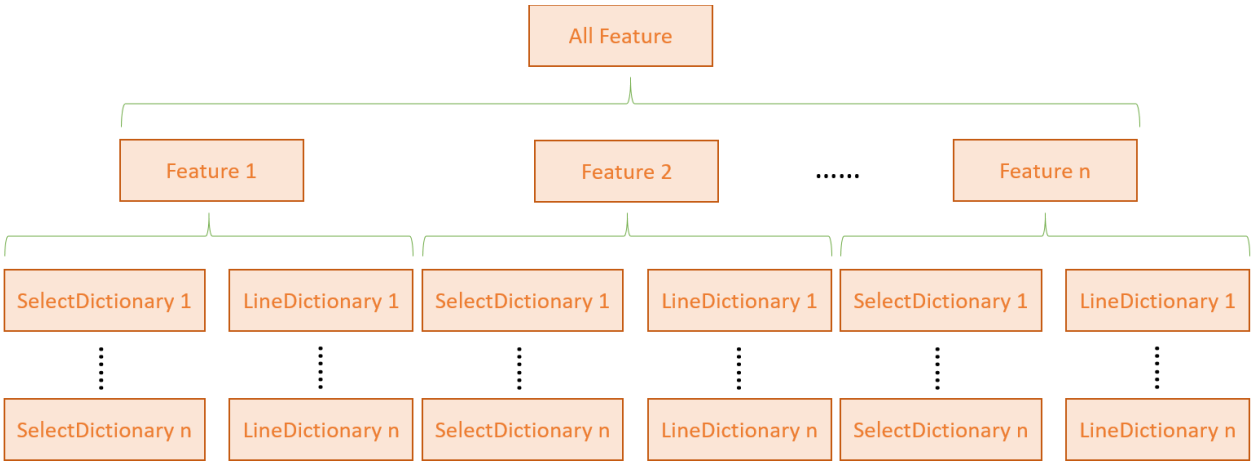


Figure 3.6 Hierarchy of the data structure.

Table 3.5 Description of SelectDictionary.

Dictionary Key	Initialization	Description
URI	Inf	Save the URI of the current thematic component as a string.
BaseURI	Inf	Save the URI of corresponding background feature as a string.
StartP	Inf	Save coordinates of the start point as a tuple.
EndP	Inf	Save coordinates of the end point as a tuple.
SegOrder	Inf	Save the component order as an integer.
VDirection	null	Save 'inverse' or 'same' to represent the vertices direction compared with the corresponding background feature.
InnerRingNum	null	Save the interior ring ID as an integer.

Table 3.6 Description of LineDictionary.

Dictionary Key	Initialization	Description
URI	Inf	Save the URI of current thematic component as a string.
LineGeometry	Inf	Save the coordinates of independent components as WKT.
SegOrder	Inf	Save the component order as an integer.
InnerRingNum	null	Save the interior ring ID as an integer.

Table 3.7 Description of Feature.

Dictionary Key	Initialization	Description
URI	Inf	Save the URI of current thematic component as a string.
LineGeometry	Inf	Save the coordinates of independent components as WKT.
SegOrder	Inf	Save the component order as an integer.
InnerRingNum	null	Save the interior ring ID as an integer.

3.4.3 Develop the Digitalization Tool

The aim of this step is to achieve the functionalities of the digitalization tool by programming in Python. After accomplishing the interface of the tool in Python Add-in Wizard, it gives the frames of the Python script which contains classes and empty functions. Each content has a corresponding class in the Python script and the name of class has been defined when add that content in the wizard. Figure 3.7 shows the default functions given in a tool class. These functions such as *onMouseDownMap()*, etc. achieve the interaction between users and the tool by responding to mouse activities. For example, when a mouse button is double-clicked, a tool function called '*onDblClick()*' is activated and codes written under it is run. Different content types have different default functions.

```
class StartPoint(object):
    """Implementation for New folder (3)_addin.tool (Tool)"""
    def __init__(self):
        self.enabled = True
        self.shape = "NONE" # Can set to "Line", "Circle" or "Rectangle"
    def onMouseDown(self, x, y, button, shift):
        pass
    def onMouseDownMap(self, x, y, button, shift):
        pass
    def onMouseUp(self, x, y, button, shift):
        pass
    def onMouseUpMap(self, x, y, button, shift):
        pass
    def onMouseMove(self, x, y, button, shift):
        pass
    def onMouseMoveMap(self, x, y, button, shift):
        pass
    def onDblClick(self):
        pass
    def onKeyDown(self, keycode, shift):
        pass
    def onKeyUp(self, keycode, shift):
        pass
    def deactivate(self):
        pass
    def onCircle(self, circle_geometry):
        pass
    def onLine(self, line_geometry):
        pass
    def onRectangle(self, rectangle_geometry):
        pass
```

Figure 3.7 Functions of a tool class in the Python script.

1) Creation of thematic components

One of the most important functionalities is to create the thematic features. To create the matched thematic components, the add-in contents: *Select Layer*, *Select Background Feature*, *Start Point*, *End Point*, and *Vertices Direction* are involved (see Table 3.4). Figure 3.8 illustrates the development process of creating the matched components.

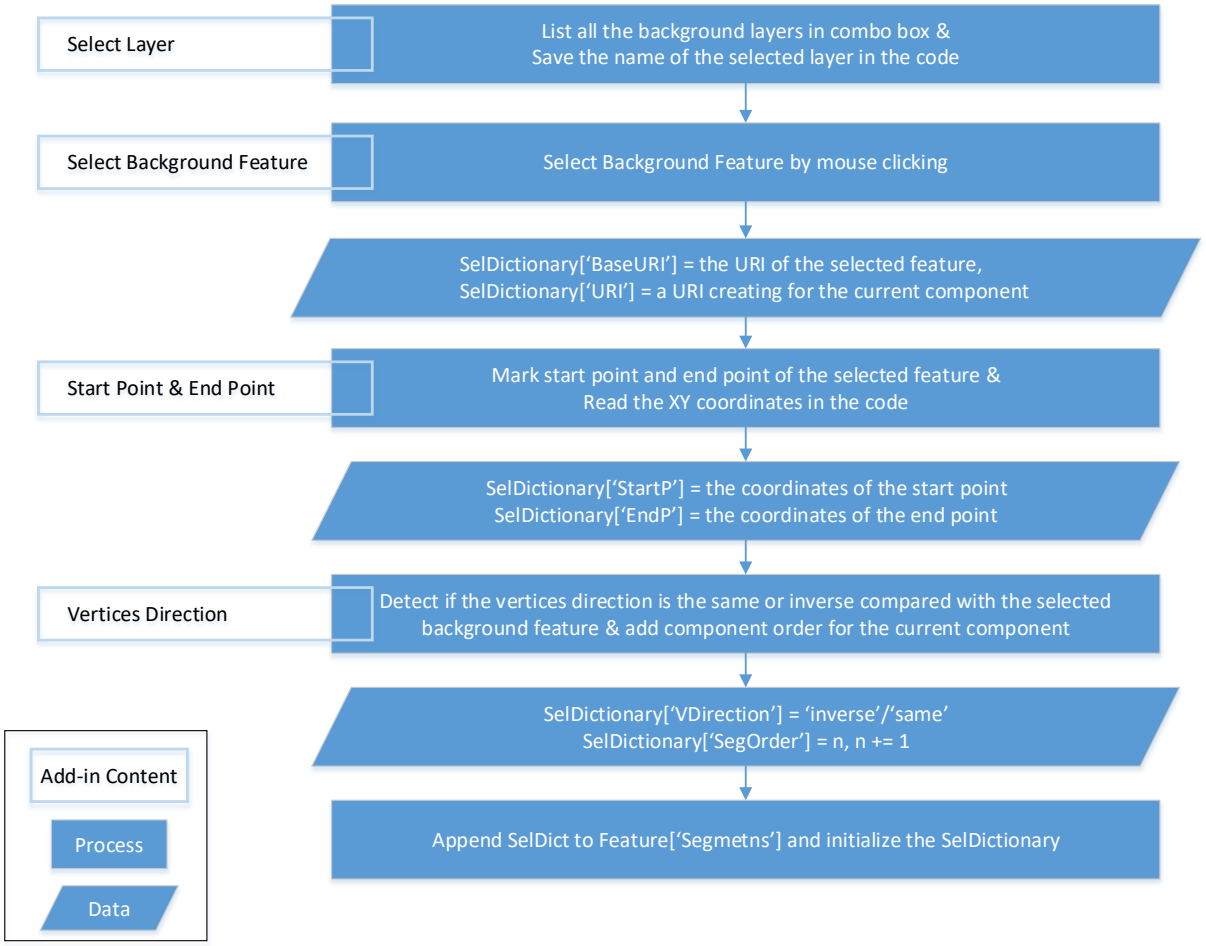


Figure 3.8 Development of the matched thematic components in the digitalization tool.

Select Layer is a combo box that lists the background layers in Table of Contents of ArcMap. It can read the name of the selected layer and pass it to the code. *OnFocus()* function can filter items in the dropdown list. In this case, the filter is the visibility of layers.

Select Background Feature is a tool that can read coordinates based on mouse activities. *OnMouseDownMap()* function within the tool class is capable of obtaining coordinates of a mouse-clicking. Therefore, it is possible to select a feature in the selected layer which is within a certain distance of that mouse-clicking. Furthermore, the URI of the selected background feature is saved in the *SelDictionary['BaseURI']*. And *SelDictionary['URI']* saves the URI of the current component created by adding a string of random numbers and letters to a domain.

The tools of *Start Point* and *End Point* get the coordinates when users click on the screen. To visualize the two points, *InsertCursor* of the Arcpy[9] package would insert a pair of XY coordinates as a record in attribute table of a point layer and thus show the point on screen.

In this study, vertices direction describes whether a matched component has the same or inverse direction as the corresponding background feature. As shown in Figure 3.9, take the background feature (a lake, the blue area) as reference, the anticlockwise arc between the

start point and end point is the correct part (illustrated in red arrow), not the arc that follow the blue arrows. Considering the exterior ring in ArcGIS is clockwise, the correct arc has an inverse direction as the background feature, so *SelDictionary['VDirection']* equals to 'inverse'. The vertices direction only applies to the selected background features which are polygons.

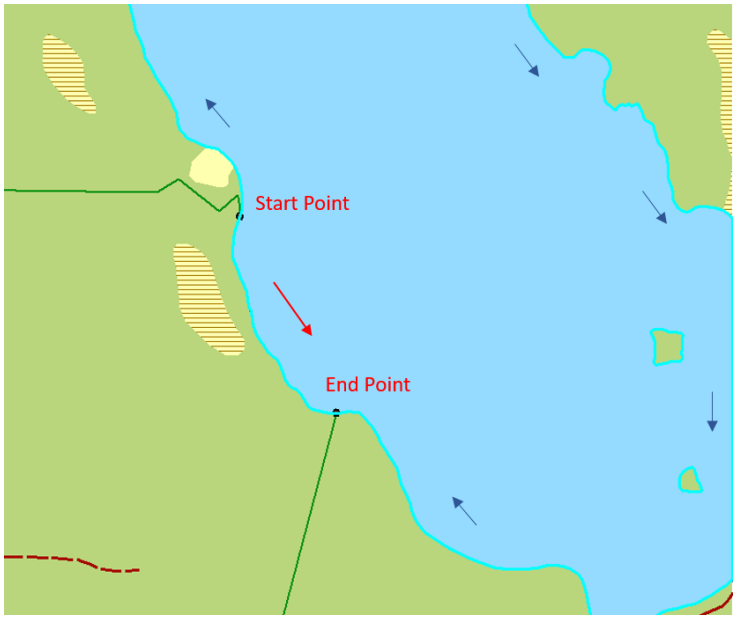


Figure 3.9 Illustration of vertices direction.

The development of the independent component is relatively simple. In the *Line* content, the geometry of a line is created by calling the *online()* function. Then the relative information is saved in the *LineDictionary*.

The snap functionality supplied by Ehsan Abdolmajidi, my second supervisor, can erase the gap between the created thematic components. After creating a matched or independent component, it is appended to the *Feature['Segments']* and then *SelDictionary* or *LineDictionary* is initialized.

2) Add RDF statements

Another important functionality of the digitalization tool is to convert the thematic data into RDF. A Python package known as *RDFLib*[10], is installed to work with RDF. Because the RDF model is a directed graph, RDFLib exposes a Graph as the primary interface. Therefore, an empty graph should be created first in the Python script (e.g. g = rdflib.Graph()) and then all the RDF triples will be added to it. Figure 3.10 shows the code of creating RDF statements for thematic data.

```
for feature in GenInfo.AllFeature:
    thematicFeature = rdflib.URIRef(feature['URI'])
    g.add((thematicFeature, rdflib.RDF.type, p.Thematic_Feature))
    g.add((thematicFeature, p.CRS, rdflib.Literal(CRS)))
    for seg in feature['Segments']:
        if len(seg) == 5 or len(seg) == 6 or len(seg) == 7:
            segment = rdflib.URIRef(seg['URI'])
            g.add((thematicFeature, p.hasComponent, segment))
            g.add((segment, rdflib.RDF.type, p.Matched_Component))

            startPoint = rdflib.URIRef(domain + str(uuid.uuid4()))
            g.add((startPoint, rdflib.RDF.type, sf.Point))
            g.add((segment, p.startsAt, startPoint))
            startGeo = EPSG + '\n' + arcpy.PointGeometry(arcpy.Point(seg['StartP'][0], seg['StartP'][1])).WKT
            g.add((startPoint, geom.asWKT, rdflib.Literal(startGeo, datatype=geom.wktLiteral)))

            endPoint = rdflib.URIRef(domain + str(uuid.uuid4()))
            g.add((endPoint, rdflib.RDF.type, sf.Point))
            g.add((segment, p.endsAt, endPoint))
            endGeo = EPSG + '\n' + arcpy.PointGeometry(arcpy.Point(seg['EndP'][0], seg['EndP'][1])).WKT
            g.add((endPoint, geom.asWKT, rdflib.Literal(endGeo, datatype=geom.wktLiteral)))

            BaseFeat = rdflib.URIRef(seg['BaseURI'])
            g.add((segment, p.isPartOf, BaseFeat))

            g.add((segment, p.componentOrder, rdflib.Literal(seg['SegOrder'])))

            if seg.has_key('VDirection'):
                g.add((segment, p.verticesOrder, rdflib.Literal(seg['VDirection'])))
            if seg.has_key('InnerRingNum'):
                g.add((segment, p.innerRingNo, rdflib.Literal(seg['InnerRingNum'], datatype=xsd.int)))

        if len(seg) == 3 or len(seg) == 4:
            segment = rdflib.URIRef(seg['URI'])
            g.add((thematicFeature, p.hasComponent, segment))
            g.add((segment, rdflib.RDF.type, p.Independent_Component))

            line = rdflib.URIRef(domain + str(uuid.uuid4()))
            g.add((line, rdflib.RDF.type, sf.LineString))
            g.add((segment, geom.defaultGeometry, line))
            lineGeo = EPSG + '\n' + seg['LineGeometry'].WKT
            g.add((line, geom.asWKT, rdflib.Literal(lineGeo, datatype=geom.wktLiteral)))

            g.add((segment, p.componentOrder, rdflib.Literal(seg['SegOrder'])))

            if seg.has_key('InnerRingNum'):
                g.add((segment, p.innerRingNo, rdflib.Literal(seg['InnerRingNum'], datatype=xsd.int)))

g.serialize('thematic_data.ttl', format="turtle")
print 'The RDF file named thematic_data is saved in {}'.format(GenInfo.workpath)
```

Figure 3.10 Python script for adding RDF statements and exporting the RDF file.

Each feature in *AllFeature* should be declared as an instance of the *Thematic_Feature*. The matched component in *Feature['segments']* should be declared as an instance of the *Matched_Component*, while the independent component should be declared as an instance of the *Independent_Component*. *A*ll the components in each feature are traversed and converted to RDF triples. Before converting thematic information in *AllFeature* to RDF statements, the relevant ontologies such as the Thematic data ontology and GeoSPARQL ontology, etc. need to be bound in the RDF file.

Finally, a separate RDF file organized in TURTLE syntax contains all the thematic information is exported and saved to the workspace that users set in the *Working Directory* content.

3) Assistant functions in the digitalization tool

he *Working Directory* content (see Table 3.4) is developed to set a work directory, create an ESRI geodatabase, the feature classes (*Point, Line, Polygon*) and a table by using functions in Arcpy package, such as *CreateFileGDB_management()*, *CreateFeatureclass_management()*, and *CreateTable_management()*, etc. Plus, A field called URI domain is created in the table.

The *Interior Ring* menu (see Table 3.4) has three button contents. In the *Start* button, the variable named *startRing* equals to 1 and the number of the components of exterior rings are saved. When the button of *Next Ring* is clicked, the variable for assigning the ID of interior rings adds 1 to itself. *Stop* button makes *startRing* equals to 0 which implies the creation of interior rings is finished.

In the *Next Feature* content (see Table 3.4), the URI domain in the table is read by function *SearchCursor()* in Arcpy package and used to form the URI for the latest thematic feature. Then the *Feature* dictionary which saves this latest thematic feature is appended to the *AllFeature* list. In addition, the *Feature* dictionary are initialized and the variable for counting component order equals to zero in the end.

All Feature content (see Table 3.4) is only used to print *AllFeature* list in the Python window of ArcMap. Users can check all the created thematic features by clicking *All Feature* button.

Delete menu (see Table 3.4) contains two buttons: *Delete Segment* and *Delete Feature*. There are two steps in the deleting operations. At first, information saved in the data structure have to be removed. To delete a segment, *SelDictionary* or *LineDictionary* for thematic components are wiped, or the last element in *Feature['Segments']* is deleted. The whole Feature dictionary or the last element of *AllFeature* list is canceled when *Delete Feature* activated. Secondly, the corresponding geometries of the points and lines users created have to be removed from the screen. This can be achieved by deleting the records in the attribute table of *Point* and *Line* feature class.

Once the Python script is accomplished, a makeaddin.py file provided by the Python Add-in Wizard is executed to generate an .esriaddin file for installing the tool into ArcMap.

4. A use case of the digitalization tool

4.1 Background

The aim of this use case is to create six nature reserves and link them to a background (topographic) map. Actually, these six nature reserves have already been created by the Swedish Mapping Agency. However, these thematic data are not connected to the background map. Consequently, there are some displacements between the nature reserves and the topographic map. And the detail level of the nature reserves remains same when the topographic map changes scale. By implementing the digitalization tool in the use case, these two problems of the thematic data could be solved. Moreover, the tool would be evaluated according to the requirement specifications listed in section 3.2.

4.2 Study area and input data

The study area is located in Västernorrland, a county in the north of Sweden, and has an area of approximately 550 km^2. The six natural reserves in this area are shown in Figure 4.1 (green polygons illustrated in red dashed rectangles).

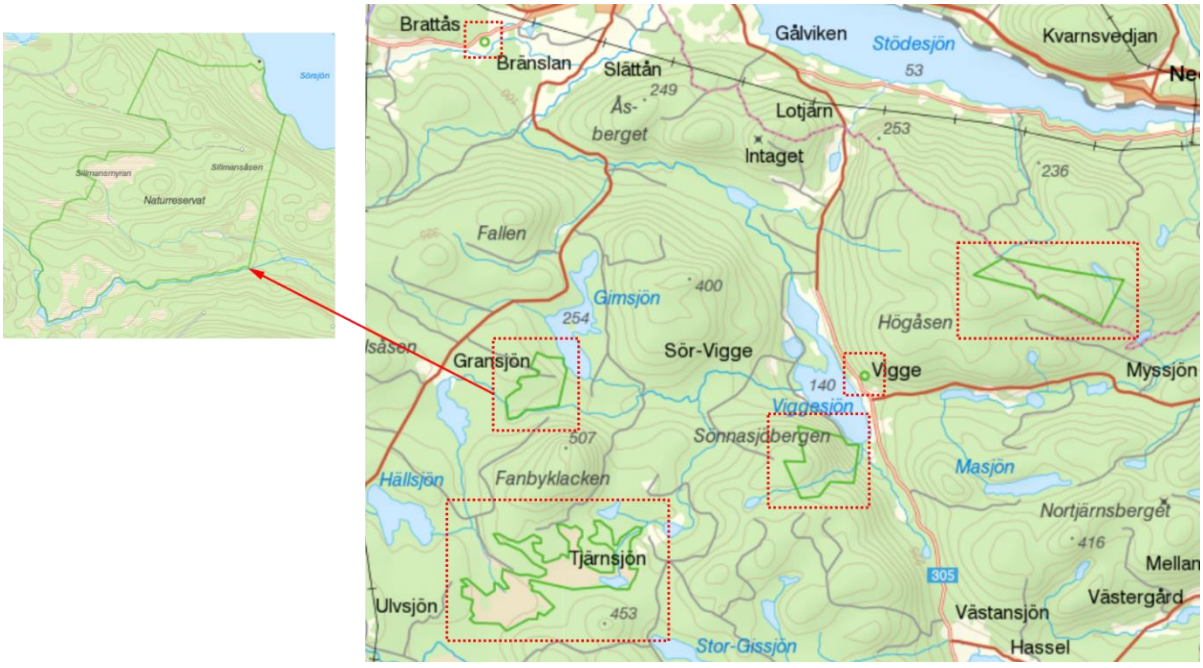


Figure 4.1 Study area in the use case, the maps are from Lantmäteriet (© Lantmäteriet, Dnr: I2014/00579).

To create the six nature reserves in the digitalization tool, the input data are the WMTS (Web Map Tile Service) provided by the Swedish Mapping Agency (see Figure 4.1), the topographic map of Västernorrland as the background map, and the ontologies (see Figure 3.2). The WMTS contains the boundary of the nature reserves created earlier, which can provide reference when visualize them in the digitalization tool. The background map is provided in

four scales: 1: 250,000, 1: 100,000, 1: 50,000, and 1: 10,000. The one in the largest scale would be used to generate the nature reserves. In this case, the main components of the topographic map are cadastral lines, roads, rivers and lakes. As for the ontologies, the Thematic data ontology and the GeoSPARQL ontology are both required.

4.3 User interaction

The .esriaddin file is the key to installing and sharing a Python add-in. The digitalization tool is installed to ArcMap when the Digtool.esriaddin file is executed. When users open ArcMap, the digitalization tool can be found after the installment. Users can simply uninstall it by using the 'Delete this Add-In' on 'Add-In Manager' dialog in ArcMap.

The background map needs to be preprocessed and organized in a multiple representation database. For instance, URIs for the interesting background features have to be generated and added in the attribute table as a value of the 'URI' field. After adding the topographic map in ArcMap, Figure 4.2 shows the final interface of the digitalization tool.

The first step to start creating and linking thematic data is to set a workspace by clicking the *Working Directory* button. In this step, users create or choose a folder as the working directory. Then three feature classes called *Point*, *Line*, *Polygon* and a table called *Domain* are created and shown in the Table of Contents automatically. The feature classes are used to display the geometries when creating thematic data. Considering the Python embedded in ArcMap does not support *input* function, and Python Add-In Wizard does not have textbox as an add-in content, the *Domain* table is created for users to input information. Users have to input the domain for building URIs of thematic data in the table before moving to next step. The related ontology files are required to be saved in the working directory as well.

To create matched thematic segments, users need to select a layer first in the *Select Layer* combo box that lists all the visible layers. Then the *Select Background Feature* is clicked to choose a background feature only in the selected layer. The selected feature is highlighted and the URI of it is obtained in the code. In order to determine the correct part of the selected feature, the coordinates of the start point and end point are saved after using the *Start Point* and *End Point* tool to click on the selected background feature. If the background feature is a polygon, users need to choose whether the correct part is clockwise or anticlockwise according to the background feature, and then the vertices direction is detected based on the choice of users.

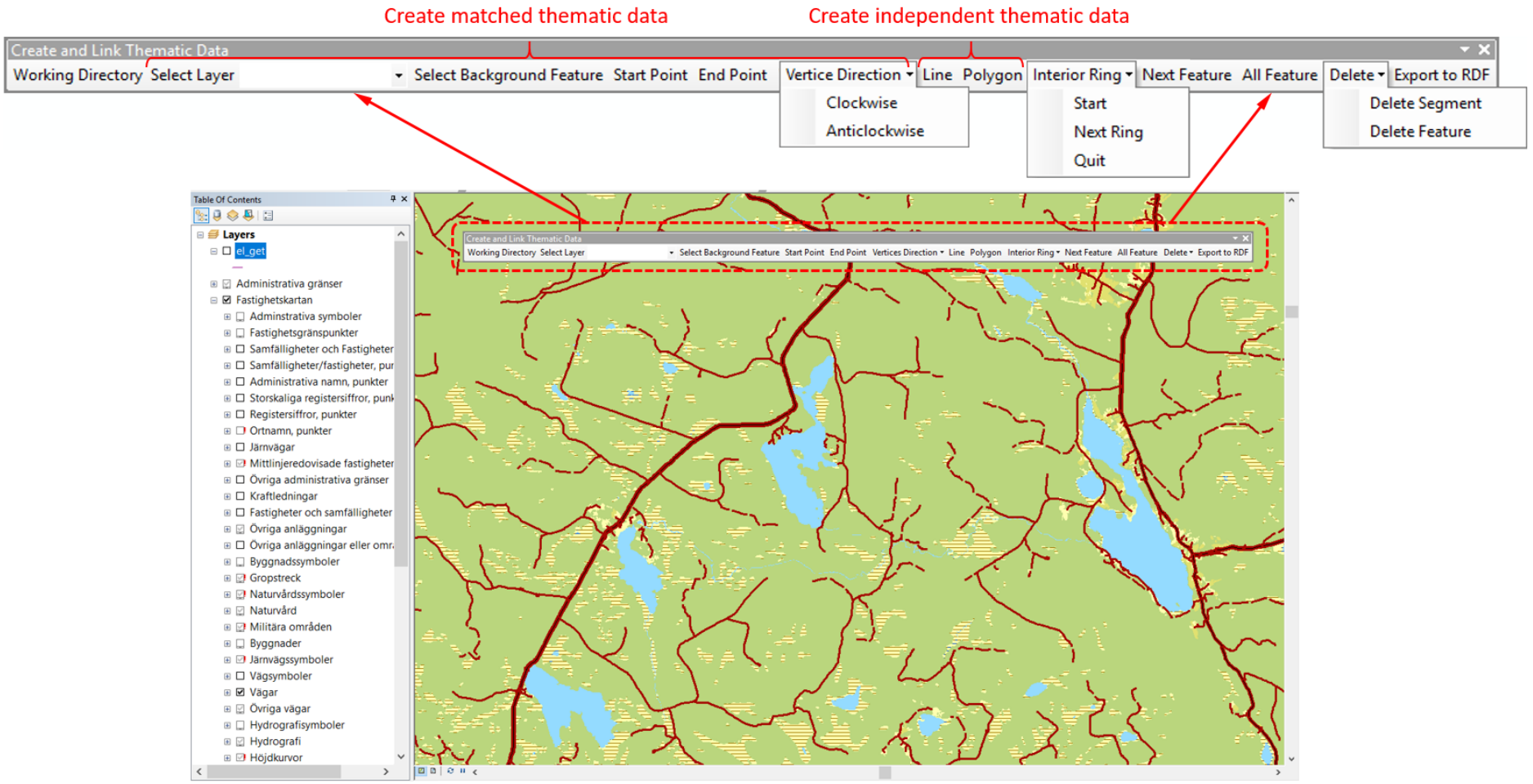


Figure 4.2 Interface of the digitalization tool in ArcMap.

There are two types of geometries for the independent thematic segment: line and polygon. Users can simply click on the screen to create a line geometry as a thematic segment and stop the creation by double-clicking. Then, the coordinates and component order of the line are saved in the code. Although users can draw polygons on screen, the polygon data are not converted to RDF statements, because saving coordinates of multipart polygons in WKT is difficult to organize in this case.

There might be more than one ring in a thematic feature as shown in Figure 4.3. Above the operations to create exterior rings are discussed. Then, the operations for interior rings are explained. The first step to create interior rings is to click the *Start* button under *Interior Ring* menu. Similar to exterior rings, interior rings are comprised of matched components and/or independent components. Therefore, after clicking the *Start* button, users can follow the operations of exterior rings to create interior ones. The *Next Ring* is activated to finish the current ring and move on to the next. In the end, click the *Quit* to stop the operations for interior rings. Every created interior ring is assigned an integer as the ID to distinguish them. Considering that an exterior ring is a clockwise ring and an interior ring is defined as an anticlockwise ring in ArcGIS, it is noteworthy that the users need to create thematic components in a certain direction. For the exterior ring in Figure 4.3, the thematic components need to be created in clockwise order, while components of the interior ring have to be created in anticlockwise order.

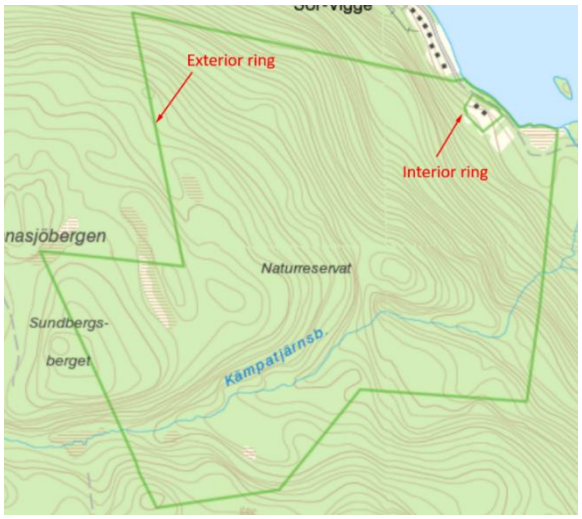


Figure 4.3 A thematic feature that contains an interior ring.

Next Feature is clicked when all the components of a thematic feature are accomplished. Then users can move to create the components of the next thematic feature or move to export RDF files if all the features are finished. *All Feature* is used to print the information of all the created thematic features in the Python window of ArcMap. If there are errors of the current or the latest segment, users can click *Delete Segment* to remove it. *Delete Feature* can remove the previous feature.

After accomplishing all the thematic features, clicking *Export to RDF* will activate the operation that converts thematic data into RDF statements. Finally, the RDF file named 'thematic data' is saved in the working directory that users set in the first step.

4.4 Output of the digitalization tool

The output of the digitalization tool is an RDF file in TURTLE syntax which can be opened by a text editor. As shown in Figure 4.4, a thematic feature is exported in the RDF file. The first two lines tell the URI of data and the type of it (Thematic_Feature). On line 3, the CRS information is specified. The relationship between the thematic feature and the thematic components is 'hasComponent' which can be found in line 4. From line 5 to line 10, the six thematic components of this thematic feature are listed. Each thematic component has its unique URI.

Figure 4.5 shows an example of a matched thematic component in the RDF file. The first two lines always tell the URI and type of it. Line 3 states that it is the third component in a thematic feature (component order starts with 0). On line 6, the properties 'isPartOf' is used to denote that this component uses a part of the geometry of a background feature: Road_1_0. Line 4 and line 5 specify that the geometry ends at an object represented in a URI. Details of the object can be found in line 9 to line 11, saying that the object type is Point and the coordinates of the end point are represented in WKT. The geometry starts at (see line 7) an object represented in line 8 and the details of it are specified in line 12 to line 14.

The example of an independent thematic component in the RDF file is given in Figure 4.6. The first two lines are the URI and type of it. Line 3 says the component has a default geometry and then the geometry object is shown in line 4. Line 7 to line 12 state that the geometry object is a line string (see line 8), and the coordinates of the geometry are represented in WKT (see line 9-12). On line 5, the component gets its own order as 5 which means it is the sixth component of a thematic feature (component order starts with 0). Line 6 tells that this component belongs to the first inner ring of a thematic feature.

```
<http://www.semanticweb.org/weiming.huang/ontologies/2016/11/thematic_map_data#0e4ac7b1-bb09-42df-b37f-680d6876945d> a
thematic_map:Thematic_Feature ;
    thematic_map:CRS "http://www.opengis.net/def/crs/EPSG/0/3006" ;
    thematic_map:hasComponent
    <http://www.semanticweb.org/weiming.huang/ontologies/2016/11/thematic_map_data#1893337b-08a6-4a40-81d4-5bfbb201fdd6>,
        <http://www.semanticweb.org/weiming.huang/ontologies/2016/11/thematic_map_data#2f92ed98-df03-4fcd-a322-57b91a3f9628>,
        <http://www.semanticweb.org/weiming.huang/ontologies/2016/11/thematic_map_data#894f571c-be91-4de8-843d-3b42fbbb5d24>,
        <http://www.semanticweb.org/weiming.huang/ontologies/2016/11/thematic_map_data#8e8e9473-a8ec-4fa2-b987-de5b97645173>,
        :cae4e23e-0a88-4582-a861-51bfc41c33c3,
        :f8b960da-7878-4d60-90f9-8bb54c028726 .
```

Figure 4.4 Example of a thematic feature in RDF file.

```
<http://www.semanticweb.org/weiming.huang/ontologies/2016/11/thematic_map_data#0ddee249-a334-4fff-a17c-b0613a09f830> a
thematic_map:Matched_Component ;
    thematic_map:componentOrder 2 ;
    thematic_map:endsAt
    <http://www.semanticweb.org/weiming.huang/ontologies/2016/11/thematic_map_data#4b3b25e7-5ab3-4966-8829-a18bda7fe40e> ;
    thematic_map:isPartOf <http://www.semanticweb.org/weiming.huang/ontologies/2016/11/base_map_data#Road_1_0> ;
    thematic_map:startsAt
    <http://www.semanticweb.org/weiming.huang/ontologies/2016/11/thematic_map_data#764d2dcd-f161-4eba-8cf5-9bbe570cfe54> .
<http://www.semanticweb.org/weiming.huang/ontologies/2016/11/thematic_map_data#4b3b25e7-5ab3-4966-8829-a18bda7fe40e> a sf:Point ;
    geosparql:asWKT """<http://www.opengis.net/def/crs/EPSG/0/3006>
POINT (578031.72261024523 6919217.4039005088)"""^^geosparql:wktLiteral .
<http://www.semanticweb.org/weiming.huang/ontologies/2016/11/thematic_map_data#764d2dcd-f161-4eba-8cf5-9bbe570cfe54> a sf:Point ;
    geosparql:asWKT """<http://www.opengis.net/def/crs/EPSG/0/3006>
POINT (577900.9743996904 6919187.034025941)"""^^geosparql:wktLiteral .
```

Figure 4.5 Example of a matched thematic component in the RDF file.

```
<http://www.semanticweb.org/weiming.huang/ontologies/2016/11/thematic_map_data#894f571c-be91-4de8-843d-3b42fbbb5d24> a
thematic_map:Independent_Component ;
    geosparql:defaultGeometry
    <http://www.semanticweb.org/weiming.huang/ontologies/2016/11/thematic_map_data#7fe5ee5f-74e5-405d-8341-d95975f737e2> ;
    thematic_map:componentOrder 5 ;
    thematic_map:innerRingNo "0"^^xsd:int .
<http://www.semanticweb.org/weiming.huang/ontologies/2016/11/thematic_map_data#7fe5ee5f-74e5-405d-8341-d95975f737e2> a
sf:LineString ;
    geosparql:asWKT """<http://www.opengis.net/def/crs/EPSG/0/3006>
MULTILINESTRING ((587038.80700503336 6909233.8512238637, 587038.87139756465 6909233.5013377769, 587007.6505017411
6909184.0241554128, 587016.3817692172 6909153.5970111778, 587073.53188360599 6909094.3302258858, 587149.99661998719
6909156.772017533, 587150.15789453487 6909156.9277020255))"""^^geosparql:wktLiteral .
```

Figure 4.6 Example of an independent thematic component in the RDF file.

4.5 Evaluation

The tool is evaluated according to the requirement specifications of the digitalization tool in section 3.2. Below follows an evaluation of the tool against these requirements.

For the first requirement, Figure 4.2 shows the two ways of generating thematic data in the interface of the digitalization tool and section 4.3 elaborates the generating operations in the tool. As shown in the Figure 4.7, a background feature is selected and highlighted, and a start point and end point are marked on it. Hence the second and third requirements are achieved. Figure 4.7 also illustrates that the tool is capable of drawing a line geometry above the background map which implies the fifth requirement is met. Figure 4.8 gives examples of saving a matched thematic component and an independent component data in the Python script, and then they are converted to RDF statements. The results are explained in section 4.4. Therefore, the fourth and sixth requirements are met. The exported RDF file can be opened in an application known as Protege which displays the relationships, data types, geometries and other information of the published thematic data in the interface. Figure 4.9 (a) shows that the created nature reserves are saved as instances of *Thematic_Feature.* Figure 4.9 (b) shows the components and CRS (Coordinate Reference System) of one thematic feature. It turns out that the last requirement is met.

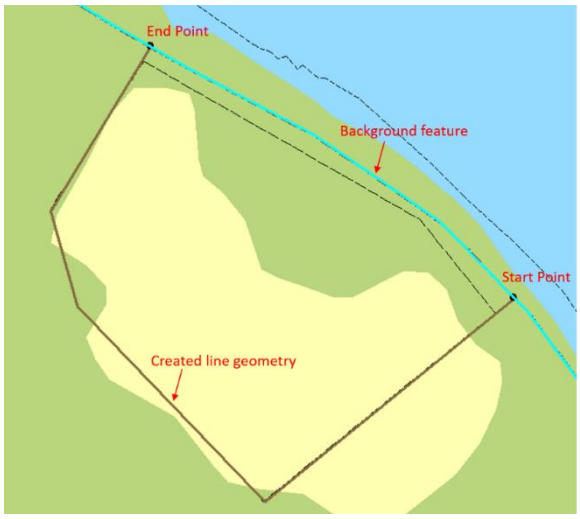
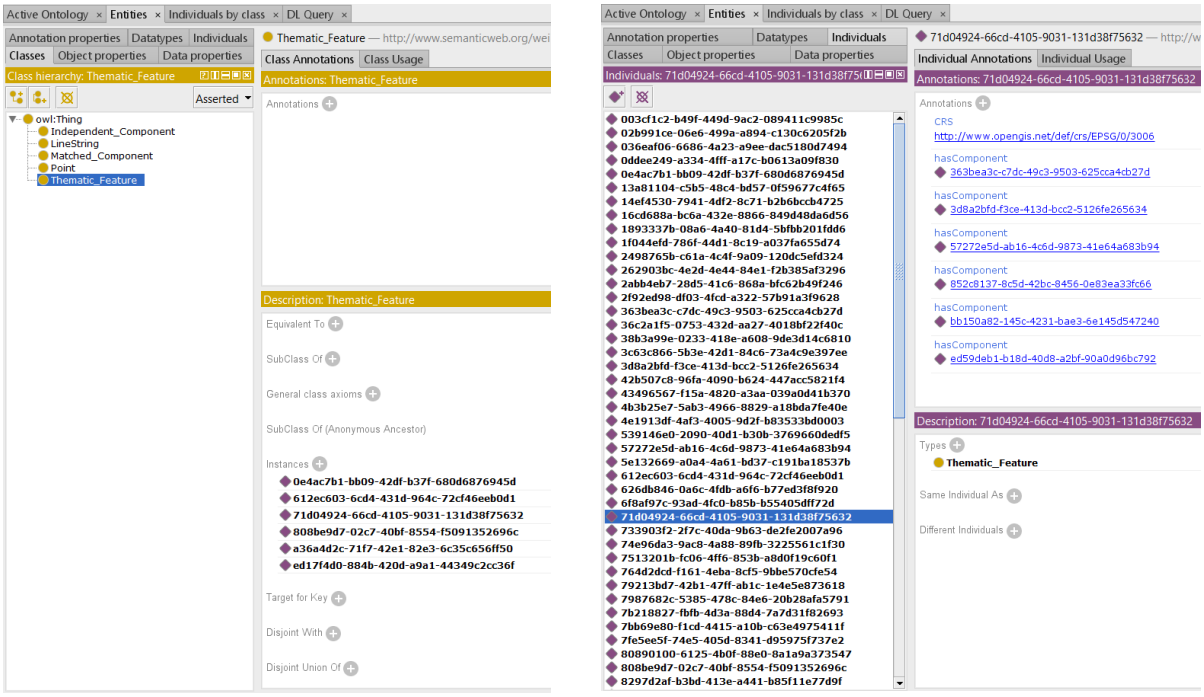


Figure 4.7 Example of selecting background feature, mark points and drawing a line in the digitalization tool.

```
{'BaseURI': u'http://www.semanticweb.org/weiming.huang/ontologies/2016/11/base_map_data#Road_3_0', 'EndP':
 (587038.50515336, 6909234.023931756), 'StartP': (587150.1442233116, 6909156.941864132), 'URI':
 u'http://www.semanticweb.org/weiming.huang/ontologies/2016/11/thematic_map_data#a837c859-a88d-487b-8b51-926ea95b1984',
 'SegOrder': 0}

{'SegOrder': 1, 'URI': u'http://www.semanticweb.org/weiming.huang/ontologies/2016/11/thematic_map_data#ab0c9cf0-59b8-
 486e-b287-8042c514aecc', 'LineGeometry': <Polyline object at 0x185f3810[0x1860d500]>}
```

Figure 4.8 Examples of saving a matched thematic component and an independent component in the Python script.



(a) (b)

Figure 4.9 Examples of opening the thematic RDF file in the Protege.

Finally, as described in section 3.1, the published RDF file is imported in the Python server developed in the Linked Thematic Data project and then displayed as a thematic map together with the multi-scale background map at the client side. This process has been verified and one example of a final cartographic mashup is given in Figure 4.10.

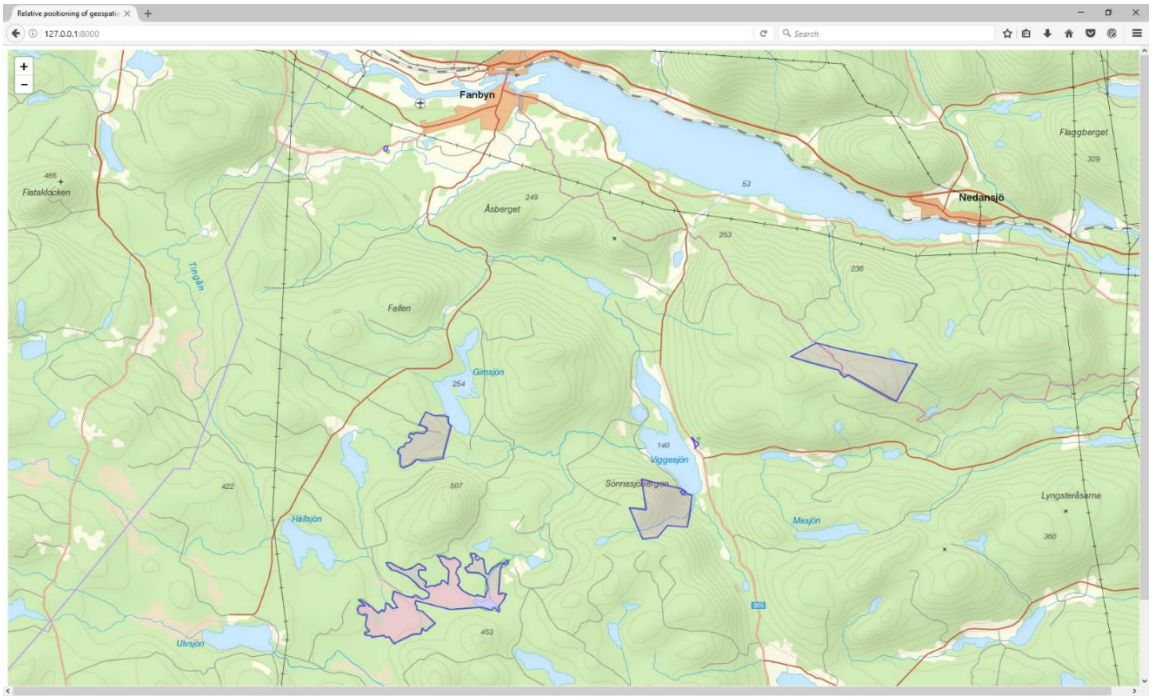


Figure 4.10 The thematic map shown together with the background map at the client side.

To conclude, all the seven requirements in section 3.2 have been fulfilled.

5. Discussion

5.1 Cartographic mashups

There are two possible solutions for developing a cartographic mashup: one is simply adding thematic data on top of background data, another one is creating thematic data linked with background data. The former solution is widely used in current cartographic mashups where thematic data are completely independent with background data. Displacement problem and scale differences between two datasets may exist in this case. The latter one is a novel idea that generates linkages in cartographic mashups by sharing some common geometries from background map to thematic data. Generating linkages through the common geometries of the two datasets in cartographic mashup can solve the displacement problem and synchronize the scale ranges between them.

Cartographic mashups which have linked datasets rely on multiple representation databases (MRDBs) of background map. The background data tend to have multiple scales. By linking each background object at different scales in an MRDB, a cartographic mashup can display different detail levels of background data. After linking thematic data to the background data, the cartographic mashup integrates the multi-scale data in real time and shows the thematic map in the same scales as the background map.

5.2 Multiple representation databases

In this study, the background data have multiple scales and are modeled in an MRDB. It is possible that polygon features in an MRDB become point features in very small scales, such as the lake feature in Figure 5.2. If a thematic component uses a part of geometry of the lake feature, when the lake becomes a point feature in the smallest scale, the Python server developed in the Linked Thematic Data project cannot find a line geometry to regenerate the thematic component, because the geometry between start point and end point disappears with the decrease of the detail level of the lake. In this case, the Python server could skip the geometry of the lake in the regeneration process.

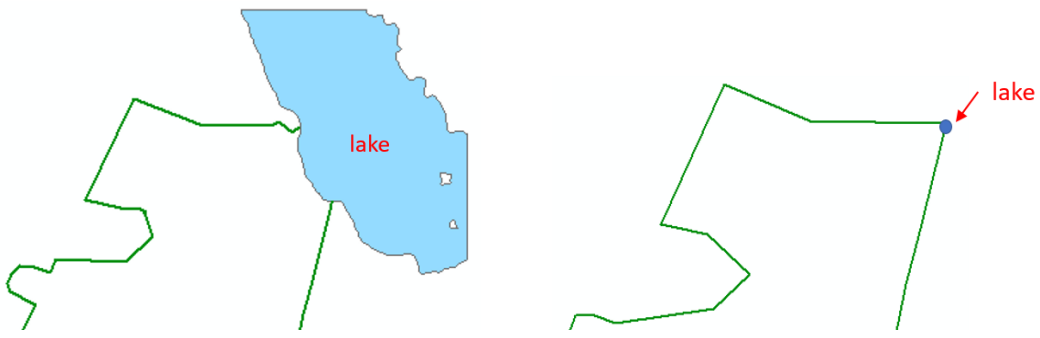


Figure 5.1 An example of background features in an MRDB.

However, if the lake point does not adjoin other features in the smallest scale, as shown in Figure 5.2, there will be gaps in the thematic feature consisting of geometries of the green lines and the lake. In this case, when the Python server starts to retrieve and regenerate the thematic feature from one green line, it could skip the lake and find the next background feature (i.e. another green line) that shares its geometry to thematic data, and generate new a line geometry between two green lines to fill the gaps.

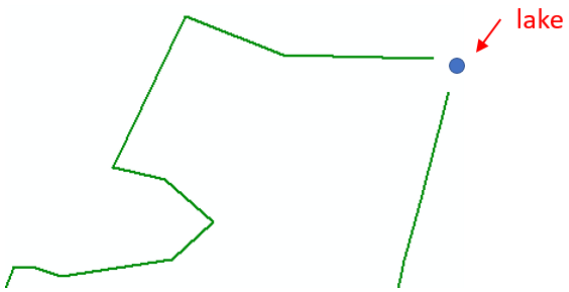


Figure 5.2 An example of background features in the smallest scale in an MRDB.

5.3 Applications of the Sematic Web technologies for geographic data

The Semantic Web technologies such as Ontology, RDF, and GeoSPARQL, provide the principles and techniques to publish geographic data as Linked data in a standard format on the Web. In this case, the linked thematic data generated from the digitalization tool can be programmatically retrieved and queried using GeoSPARQL. By sending queries and receiving results, the geometric and topological relationships between geospatial objects in the thematic data are exposed by GeoSPARQL. For example, potential queries over the thematic data could be 'Which nature reserves are within 3km of rivers?', 'Which nature reserves interact with a certain lake?' Functions of GeoSPARQL such as '*geof:distance*' and '*geo:intersects*' can be used to answer these queries.

Although GeoSPARQL has a good performance in some spatial queries and calculations when explicit relations provided, there are limitations of it. GeoSPARQL cannot support complicated spatial analysis (e.g. network analysis) and arithmetic operations (e.g. calculate the area percentage of nature reserves in the study area) for the published data.

Using the Semantic Web technologies for geographic data is still a relatively novel field. There are few browsers or WebGIS clients that support data discovery and query using GeoSPARQL. However, considering its capability of explicitly and semantically describe the interrelations of data, GeoSPARQL and other Sematic Web technologies will likely become more used in the future.

5.4 Prerequisites for the method

Below three prerequisites for the method of linking thematic data to a background map are discussed. Firstly, background data have to be organized in MRDBs (multiple representation

databases) which ensure the different detail levels at different scales. It can make thematic data have the same scales once they are linked to the background map. Secondly, a common geometry between thematic data and background map is the key to linking them together. The method is inapplicable for the cases that the geometries of thematic features are completely different from the background map. Finally, thematic data have to trust the data quality of the background map, because some of them share the common geometries. The errors of the background map might propagate to the thematic data and thus influence the accuracy of the thematic data.

5.5 The digitalization tool

The digitalization tool is capable of creating and linking thematic data to a background map. It takes into account the majority situations of creating thematic data and allows users to create and delete geometries flexibly. In addition, as a Python add-in for ArcGIS, the digitalization tool is easy to share, install and remove as long as users have ArcGIS available.

There are disadvantages of the digitalization tool. Firstly, when users start drawing a line geometry, the screen cannot be zoomed in/out and panned through any mouse activities. It may cause errors and displacements of the created thematic data if the one component has a large spatial extent that could only be shown at a small scale. Although it does bring inconvenience for users, data quality can be ensured by splitting one line geometry to several pieces and creating a small piece each time at a bigger scale. Therefore, a line geometry of an independent component would consist of several line components which are precisely connected to each other thanks to the snap functionality. Secondly, every instance of thematic feature and component is not named when publishing them as linked data. Hence it may be difficult to distinguish one from another in the RDF file.

In addition to addressing the disadvantages of the digitalization tool, the future work will complement another important functionality of the digitalization tool which is enabling users to create polygon geometry as an independent thematic component.

6. Conclusions

The overall aim of this study is to develop a digitalization tool of thematic data that is linked to the background map. Specifically, the digitalization tool should provide two ways to generate the geometry of thematic data: selecting geometries from a background map and creating new geometries. Plus, the created thematic data should be published as linked data according to a certain ontology.

The digitalization tool is developed as a Python add-in of ArcMap. The generation of geometries of thematic data has been achieved by developing the functionalities in the digitalization tool. For the thematic data that rely on the background features, the tool allows users to select the corresponding features and determine the common geometries from it. By saving the URI of the background feature, coordinates of the start and end points, and other information, the common geometries are shared between background map and the thematic data. For the independent thematic data, a line functionality is provided to create polylines.

In order to link the thematic data with the background map data, a novel method is proposed by leveraging the Semantic Web technologies like ontology, RDF, and GeoSPARQL. First, by identifying the relations between the background map and the thematic data, the Thematic data ontology are established as a subclass of *Feature* of GeoSPARQL ontology. Then, the digitalization tool converts the thematic data into RDF triples based on the Thematic data ontology and thus links them with the background map.

By using the tool to create six nature reserves as thematic data on a topographic map, it turns out that the method and the tool are both applicable to create and link thematic data to a background map.

Reference

Battle, R., and D. Kolas. 2012. Enabling the geospatial Semantic Web with Parliament and GeoSPARQL. *Semantic Web* 3. IOS Press: 355–370. doi:10.3233/SW-2012-0065.

Batty, M., A. Hudson-Smith, R. Milton, and A. Crooks. 2010. Map mashups, Web 2.0 and the GIS revolution. *Annals of GIS* 16. Taylor & Francis : 1–13. doi:10.1080/19475681003700831.

Beckett, D., and T. Berners-Lee. 2011. Turtle - Terse RDF Triple Language. Retrieved June 18, 2017, from https://www.w3.org/TeamSubmission/turtle/#sec-intro

Berners-Lee, T. 2006. Linked Data design issues. Retrieved February 18, 2017, from https://www.w3.org/DesignIssues/LinkedData.html

Berners-Lee, T., J. Hendler, and O. Lassila. 2001. The Semantic Web. *Scientific American* 284: 34–43. doi:10.1038/scientificamerican0501-34.

Bobzien, M., D. Burghardt, I. Petzold, M. Neun, and R. Weibel. 2008. Multi-representation Databases with Explicitly Modeled Horizontal, Vertical, and Update Relations. *Cartography and Geographic Information Science* 35. Taylor & Francis Group : 3–16. doi:10.1559/152304008783475698.

Boulos, M., M. Scotch, K.-H. Cheung, and D. Burden. 2008. Web GIS in practice VI: a demo playlist of geo-mashups for public health neogeographers. *International Journal of Health Geographics* 7. BioMed Central: 38. doi:10.1186/1476-072X-7-38.

Burghardt, D., I. Petzold, and M. Bobzien. 2010. Relation Modelling within Multiple Representation Databases and Generalisation Services. *The Cartographic Journal* 47. Taylor & Francis: 238–249. doi:10.1179/000870410X12699418769035.

Ciobanu, G., R. Horne, and V. Sassone. 2016. A descriptive type foundation for RDF Schema. *Journal of Logical and Algebraic Methods in Programming* 85: 681–706. doi:10.1016/j.jlamp.2016.02.006.

Consoli, S., V. Presutti, D. Reforgiato Recupero, A. G. Nuzzolese, S. Peroni, M. Mongiovi', and A. Gangemi. 2017. Producing Linked Data for Smart Cities: The Case of Catania. *Big Data Research* 7: 1–15. doi:10.1016/j.bdr.2016.10.001.

Dunkars, M. 2004. Multiple representation databases for topographic information. Infrastructure. PhD Thesis. Stockholm, Sweden: KTH.

GeoSPARQL support — GraphDB SE 8.1 documentation. 2017. Retrieved June 19, 2017, from http://graphdb.ontotext.com/documentation/standard/geosparql-support.html.

Gould, N., and W. Mackaness. 2016. From taxonomies to ontologies: formalizing generalization knowledge for on-demand mapping. *Cartography and Geographic Information Science* 43. Taylor & Francis: 208–222. doi:10.1080/15230406.2015.1072737.

Hart, G., and C. Dolbear. 2013. *Linked data: a geographic perspective*. CRC Press.

Heath, T., and C. Bizer. 2011. Linked Data: Evolving the Web into a Global Data Space.

Synthesis Lectures on the Semantic Web: Theory and Technology 1: 1–136. doi:10.2200/S00334ED1V01Y201102WBE001.

Hietanen, E., L. Lehto, and P. Latvala. 2016. PROVIDING GEOGRAPHIC DATASETS AS LINKED DATA IN SDI. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLI-B2: 583–586. doi:10.5194/isprsarchives-XLI-B2-583-2016.

Huang, W. Relative positioning of geospatial data using a linked data approach – a case study of cartographic mashups. Personal communication, May 20, 2017

Huang, W., A. Mansourian, and L. Harrie. 2016. On-demand mapping and integration of thematic data. In *19th ICA Workshop on Generalisation and Multiple Representation*. Helsinki, Finland.

Jaara, K., C. Duchêne, and A. Ruas. 2011. Toward the generalisation of cartographic mashups: Taking into account the dependency between the thematic data and the reference data throughout the process of automatic generalisation. In *14th ICA workshop on generalisation and multiple representation*. Paris, France.

Karnatak, H. C., R. Shukla, V. K. Sharma, Y. V. S. Murthy, and V. Bhanumurthy. 2012. Spatial mashup technology and real time data integration in geo-web application using open source GIS – a case study for disaster management. *Geocarto International* 27. Taylor & Francis : 499–514. doi:10.1080/10106049.2011.650651.

Kilpeläinen, T. 2000. Maintenance of Multiple Representation Databases for Topographic Data. *The Cartographic Journal* 37: 101–107. doi:10.1179/0008704.37.2.p101.

Kuhn, W., T. Kauppinen, and K. Janowicz. 2014. Linked Data - A Paradigm Shift for Geographic Information Science. In *Geographic Information Science: 8th International Conference, GIScience 2014, Vienna, Austria, September 24-26, 2014. Proceedings*, ed. M. Duckham, E. Pebesma, K. Stewart, and A. U. Frank, 173–186. Cham: Springer International Publishing. doi:10.1007/978-3-319-11593-1_12.

Li, S., and J. Gong. 2008. Mashup: A new way of providing web mapping/GIS services. In *ISPRS Congress Beijing*, 639–649.

Merrill, D. 2006. Mashups: The new breed of Web app. *IBM Web Architecture Technical Library*.

Patroumpas, K., N. Georgomanolis, T. Stratiotis, M. Alexakis, and S. Athanasiou. 2015. Exposing INSPIRE on the Semantic Web. *Web Semantics: Science, Services and Agents on the World Wide Web* 35: 53–62. doi:10.1016/j.websem.2015.09.003.

Perry, M., and J. Herring. 2012. OGC GeoSPARQL-A geographic query language for RDF data. *OGC Implementation Standard, ref: OGC*.

Powell, J., M. Hopkins, J. Powell, and M. Hopkins. 2015. 5 – Ontologies. In *A Librarian's Guide to Graphs, Data and the Semantic Web*, 31–43. doi:10.1016/B978-1-84334-753-8.00005-1.

Shvaiko, P., F. Farazi, V. Maltese, A. Ivanyukovich, V. Rizzi, D. Ferrari, and G. Ucelli. 2012. Trentino Government Linked Open Geo-data: A Case Study. In , 196–211. Springer, Berlin,

Heidelberg. doi:10.1007/978-3-642-35173-0_13.

Stadler, C., J. Lehmann, K. Höffner, and S. Auer. 2012. LinkedGeoData: A core for a web of spatial open data. *Semantic Web* 3. IOS Press: 333–354. doi:10.3233/SW-2011-0052.

Toomanian, A., L. Harrie, A. Mansourian, and P. Pilesjo. 2013. Automatic integration of spatial data in viewing services. *Journal of Spatial Information Science*. doi:10.5311/JOSIS.2013.6.87.

Usery, E. L., and D. E. Varanka. 2012. Design and development of linked data from the National Map. *Semantic Web* 3: 371–384. doi:10.3233/SW-2011-0054.

W3C Semantic Web Activity Homepage. 2017. Retrieved May 19, 2017, from https://www.w3.org/2001/sw/

Web Page:

1. https://www.lonelyplanet.com/sweden/stockholm/map
2. https://maps.ngdc.noaa.gov/viewers/hazards/
3. https://maps.nyc.gov/crime/
4. https://www.w3schools.com/sql/sql_intro.asp
5. http://www.esri.com/arcgis/about-arcgis
6. http://desktop.arcgis.com/en/arcmap/
7. https://www.python.org/about/
8. http://desktop.arcgis.com/en/arcmap/10.4/analyze/python-addins/what-is-a-python-add-in.htm
9. http://pro.arcgis.com/en/pro-app/arcpy/get-started/what-is-arcpy-.htm
10. https://rdflib.readthedocs.io/en/stable/gettingstarted.html

Department of Physical Geography and Ecosystem Science, Lund University

Lund University GEM thesis series are master theses written by students of the international master program on Geo-information Science and Earth Observation for Environmental Modelling and Management (GEM). The program is a cooperation of EU universities in Iceland, the Netherlands, Poland, Sweden and UK, as well a partner university in Australia. In this series only master thesis are included of students that performed their project at Lund University. Other theses of this program are available from the ITC, the Netherlands (www.gem-msc.org or www.itc.nl).

The student thesis reports are available at the Geo-Library, Department of Physical Geography and Ecosystem Science, University of Lund, Sölvegatan 12, S-223 62 Lund, Sweden. Report series started 2013. The complete list and electronic versions are also electronic available at the LUP student papers (https://lup.lub.lu.se/student-papers/search/) and through the Geo-library (www.geobib.lu.se).

1 Soheila Youneszadeh Jalili (2013) The effect of land use on land surface temperature in the Netherlands

2 Oskar Löfgren (2013) Using Worldview-2 satellite imagery to detect indicators of high species diversity in grasslands

3 Yang Zhou (2013) Inter-annual memory effects between Soil Moisture and NDVI in the Sahel

4 Efren Lopez Blanco (2014) Assessing the potential of embedding vegetation dynamics into a fire behaviour model: LPJ-GUESS-FARSITE

5 Anna Movsisyan (2014) Climate change impact on water and temperature conditions of forest soils: A case study related to the Swedish forestry sector

6 Liliana Carolina Castillo Villamor (2015) Technical assessment of GeoSUR and comparison with INSPIRE experience in the context of an environmental vulnerability analysis using GeoSUR data

7 Hossein Maazallahi (2015) Switching to the “Golden Age of Natural Gas” with a Focus on Shale Gas Exploitation: A Possible Bridge to Mitigate Climate Change?

8 Mohan Dev Joshi (2015) Impacts of Climate Change on *Abies spectabilis*: An approach integrating Maxent Model (MAXent) and Dynamic Vegetation Model (LPJ-GUESS)

9 Altaaf Mechiche-Alami (2015) Modelling future wheat yields in Spain with LPJ-GUESS and assessing the impacts of earlier planting dates

10 Koffi Unwana Saturday (2015) Petroleum activities, wetland utilization and livelihood changes in Southern Akwa Ibom State, Nigeria: 2003-2015

11 José Ignacio Díaz González (2016) Multi-objective optimisation algorithms for GIS-based multi-criteria decision analysis: an application for evacuation planning

12 Gunjan Sharma (2016) Land surface phenology as an indicator of performance of conservation policies like Natura2000

13 Chao Yang (2016) A Comparison of Four Methods of Diseases Mapping

14 Xinyi Dai (2016) Dam site selection using an integrated method of AHP and GIS for decision making support in Bortala, Northwest China

15 Jialong Duanmu (2016) A multi-scale based method for estimating coniferous forest aboveground biomass using low density airborne LiDAR data

16 Tanyaradzwa J. N. Muswera (2016) Modelling maize (Zea Mays L.) phenology using seasonal climate forecasts

17 Maria Angela Dissegna (2016) Improvement of the GPP estimations for Sudan using the evaporative fraction as water stress factor

18 Miguel G. Castro Gómez (2017) Joint use of Sentinel-1 and Sentinel-2 for land cover classification: A machine learning approach

19 Krishna Lamsal (2017) Identifying potential critical transitions in a forest ecosystem using satellite data

20 Maimoona Zehra Jawaid (2017) Glacial lake flood hazard assessment and modelling: a GIS perspective

21 Tracy Zaarour (2017) Application of GALDIT index in the Mediterranean region to assess vulnerability to sea water intrusion

22 Stephania Zabala (2017) Comparison of multi-temporal and multispectral Sentinel-2 and UAV (unmanned aerial vehicle) imagery for crop type mapping

23 Ximena Tagle (2017) Radiometric calibration of unmanned aerial vehicle remote sensing imagery for vegetation mapping

24 Tesfaye Gebeyehu Admasu (2017) Monitoring trends of greenness and LULC (land use/land cover) in Addis Ababa and its surrounding using MODIS time-series and LANDSAT data

25 Haiqi Xu (2017) Development of a digitalization tool for linking thematic data to a background map