A declarative approach based on Semantic Web technologies to specify and generate adaptive geovisualisations

Matthieu Viry

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

Spécialité : Informatique

Arrêté ministériel : 25 mai 2016

Présentée par

Matthieu VIRY

Thèse dirigée par **Marlène VILLANOVA-OLIVER**, MCF, Université Grenoble Alpes
et codirigée par **Paule-Annick DAVOINE**, Université Grenoble Alpes

préparée au sein du **Laboratoire d'Informatique de Grenoble**

dans **l'École Doctorale Mathématiques, Sciences et technologies de l'information, Informatique**

Une approche déclarative basée sur les technologies du Web sémantique pour spécifier et générer des géovisualisations adaptatives

A declarative approach based on Semantic Web technologies to specify and generate adaptive geovisualisations

Thèse soutenue publiquement le **16 décembre 2021**,
devant le jury composé de :

Monsieur EMMANUEL PIETRIGA
Directeur de Recherche, INRIA Saclay – Île-de-France, Rapporteur

Madame NATHALIE AUSSENAC-GILLES
Directrice de Recherche, CNRS Délégation Occitanie Ouest, Rapporteure

Madame ANA-MARIA OLTEANU-RAIMOND
Directrice de Recherche, Institut National de l'Information Géographique et Forestière, Examinatrice

Monsieur GHISLAIN AUGUSTE ATEMEZING
Docteur en Sciences, MONDECA SA, Examinateur

Monsieur JEROME EUZENAT
Directeur de Recherche, INRIA Grenoble – Rhône-Alpes, Président

Madame MARLENE VILLANOVA-OLIVER
Maîtresse de Conférences HDR, Université Grenoble Alpes, Directrice de thèse

Madame PAULE-ANNICK DAVOINE
Professeure des Universités, Université Grenoble Alpes, Co-directrice de thèse

Remerciements

J'exprime mes plus sincères remerciements à Madame Nathalie Aussenac-Gilles et à Monsieur Emmanuel Pietriga pour avoir accepté d'être rapporteurs de ce travail de thèse. Merci pour la qualité de vos remarques et de vos suggestions.

Je remercie également sincèrement Madame Ana-Maria Olteanu-Raimond, Monsieur Ghislain Auguste Atemezing et Monsieur Jérôme Euzenat pour avoir accepté de faire partie de mon jury en tant qu'examinateurs.

Mes plus vifs remerciements vont à mes directrices de thèse, Marlène Villanova-Oliver et Paule-Annick Davoine pour m'avoir permis de réaliser mon travail dans d'excellentes conditions, pour votre soutien de chaque instant et pour la bienveillance qui a toujours caractérisé votre encadrement. Je suis très reconnaissant pour l'acceuil chaleureux que vous m'avez réservé au sein de l'équipe STeamer et pour l'opportunité que vous m'avez donnée.

Je tiens également à remercier Éric Kergosien pour sa participation à mon comité de suivi de thèse.

De plus, j'adresse toute ma gratitude aux personnels administratifs et techniques du LIG, de l'école doctorale et du collège doctoral pour m'avoir permis de réaliser mon travail dans les meilleures conditions possibles, en particulier : Pascale Poulet, Maud Chorier, Michelle Hamm, Françoise Jeewooth, Chahla Domenget, Sandrine Ferrari et Nadine Masoch.

Merci aux enseignants-chercheurs m'ayant confié des enseignements, pour cette formidable opportunité et pour la liberté qui m'a été donnée : Marlène Villanova-Oliver, Paule-Annick Davoine, Sylvain Bouveret et Adriana Diaconu.

Merci à l'ensemble des membres du projet ANR CHOUCAS pour la richesse et la qualité de nos échanges.

Merci à tous les collègues et anciens collègues de l'équipe STeamer : Maeva Seffar, Aline Menin, David Noël, Matthew Sreeves, Camille Cavalière, Jacques Gautier, Oton Copi, Clément Chagnaud, Sonia Chardonnel, Jérôme Gensel, Sylvain Bouveret, Danielle Ziebelin, Philippe Genoud, Julie Dugdale, Carole Adam, Camille Bernard, Alice Jacquier, Armen Inants, Pier Castillo-Mora, Samandar Ibragimov, Rabih Fares, et mes actuelles voisines de bureau, Fatima Danash, Rouba Iskandar et Manon Prédhumeau. C'est, ou ce fut, un réel plaisir de travailler à vos côtés.

Merci à ma famille et à mes amis pour leur soutien sans faille.

Un grand merci à Marlène Villanova-Oliver, Paule-Annick Davoine et Flora Hayat pour leurs relectures assidues de ce manuscrit.

Merci à toute l'équipe de l'UMS RIATE pour m'avoir encouragé à entreprendre une thèse.

Merci à tous les contributeurs des nombreux projets *open source* utilisés directement ou indirectement par mon travail.

Abstract

The Semantic Web technologies proposed by the W3C have demonstrated their strength when it comes to exchanging data (with RDF), formalising various knowledge domains (through RDF-S/OWL ontologies) or querying this knowledge (with SPARQL). Thus, today, many models make use of Semantic Web technologies and describe a domain with a geospatial dimension possibly relying here on specific models to describe this spatial dimension at a high level, e.g. GeoSPARQL. In GI Science, besides knowledge modelling and representation, the main challenge remains that of the geovisual restitution of the information. Indeed, geospatial information and the interfaces for viewing and interacting with it, commonly referred to as geovisualisation applications, play an indispensable role in the understanding of various spatial phenomena and in the decision-making processes involving this information.

This works explores how Semantic Web technologies can facilitate the specification and generation of adaptive geovisualisations. Indeed, classical approaches for creating geovisualisation from RDF data involve numerous transformations of the data, which can lead to the loss of the semantics encoded in the models. In addition, approaches allowing the direct geovisual exploitation of RDF data described by ontological models are rare and have significant drawbacks. Besides, making cartographic representations and geovisualisation applications that make sense is a difficult task that is worth assisting, especially when the designer is not a cartographer.

In this thesis we present the CoViKoa framework. CoViKoa allows to describe, through a purely declarative specification document, how to create a geovisualisation for an RDF model and dataset. To do this, CoViKoa adopts Semantic Web technologies. First, it is based on an ecosystem of ten ontologies, six that we propose and four that we reuse from the literature. Secondly, it is instrumented by SHACL rules, derived from the specification document, which allow to establish the links between data and the way they appear in the components of a geovisualisation. Finally, the proposed framework allows the publication of the resulting RDF graph behind a SPARQL interface. This RDF graph contains all the necessary elements for building a geovisualisation application. It is exploited by a Web application that we propose and that allows to concretely build the corresponding geovisualisation.

The geovisualisations that can be described with CoViKoa are rich and complex: various mechanisms of data transformations, filters and selections are proposed to the user, directly in RDF. It is also possible to describe several types of interactions between component data. The CoViKoa framework is mainly intended for a knowledge engineer who would like to geovisually leverage the geospatial RDF data he or she has. Nevertheless, since it is based on rich ontologies that can describe many aspects of geovisualisation applications, it is also intended for geovisualisation researchers who can use these vocabularies as an analytical framework to describe and compare geovisualisation applications while benefiting from the power and expressiveness of the SPARQL query language to conduct their analyses. We also validate our proposal with two case studies. The first one focuses on the processing of the search for a lost person in mountain environnements, and the second one deals with the spatio-temporal evolutions of statistical territorial entities.

Keywords: knowledge representation, geovisualisation, Semantic Web, semantic rules

Résumé

Les technologies du Web sémantique proposées par le W3C ont démontré leur force lorsqu'il s'agit d'échanger des données (avec RDF), de formaliser différents domaines de connaissances (à travers les ontologies RDFS/OWL) ou d'interroger ces connaissances (avec SPARQL). Ainsi, aujourd'hui, de nombreux modèles utilisent les technologies du Web Sémantique et décrivent un domaine avec une dimension géospatiale, en s'appuyant éventuellement sur des modèles spécifiques pour décrire cette dimension spatiale à un haut niveau, par exemple GeoSPARQL. Dans les Sciences de l'Information Géographique, outre la modélisation et la représentation des connaissances, le principal défi reste celui de la restitution géovisuelle de l'information. En effet, les informations géospatiales et les interfaces permettant de les visualiser et d'interagir avec elles, communément appelées applications de géovisualisation, jouent un rôle indispensable dans la compréhension des phénomènes spatiaux et dans les processus de prise de décisions impliquant ces informations.

Ce travail explore comment les technologies du Web Sémantique peuvent faciliter la spécification et la génération de géovisualisations adaptatives. En effet, les approches classiques pour créer des géovisualisations à partir de données RDF impliquent de nombreuses transformations des données, ce qui peut conduire à la perte de la sémantique véhiculée par les modèles. De plus, les approches permettant l'exploitation géovisuelle directe de données RDF décrites par des modèles ontologiques sont rares et présentent d'importantes limites. Par ailleurs, réaliser des représentations cartographiques et des applications de géovisualisation qui ont du sens est une tâche difficile qui mérite d'être assistée, surtout lorsque le concepteur n'est pas un cartographe.

Dans cette thèse, nous présentons le framework CoViKoa. CoViKoa permet de décrire, à travers un document de spécification purement déclaratif, comment créer une géovisualisation pour un modèle et un jeu de données RDF. Pour ce faire, CoViKoa adopte les technologies du Web Sémantique. Premièrement, il s'appuie sur un écosystème de dix ontologies, six que nous proposons et quatre que nous réutilisons de la littérature. Ensuite, il est instrumenté par des règles SHACL, issues du document de spécification, qui permettent d'établir les liens entre les données et la manière dont elles apparaissent dans les composants d'une géovisualisation. Enfin, le framework proposé permet la publication du graphe RDF résultant derrière une interface SPARQL. Ce graphe RDF contient tous les éléments nécessaires à la construction d'une application de géovisualisation. Il est exploité par une application Web que nous proposons et qui permet de construire concrètement la géovisualisation correspondante.

Les géovisualisations qui peuvent être décrites avec CoViKoa sont riches et complexes : différents mécanismes de transformations, filtres et sélections de données sont proposés à l'utilisateur, et ce directement en RDF. Il est également possible de décrire plusieurs types d'interactions entre les données des composants. Le framework CoViKoa est principalement destiné à un ingénieur de la connaissance qui souhaite exploiter géovisuellement les données RDF géospatiales dont il dispose. Néanmoins, puisqu'il est basé sur des ontologies riches qui peuvent décrire de nombreux aspects des applications de géovisualisation, il est également destiné aux chercheurs en géovisualisation qui peuvent utiliser ces vocabulaires comme cadre analytique pour décrire et comparer les applications de géovisualisation tout en bénéficiant de la puissance et de l'expressivité du langage de requête SPARQL pour mener leurs analyses. Nous validons également notre proposition par deux études de cas. La première porte sur le traitement de la recherche d'une personne perdue en milieu montagneux, et la seconde traite des évolutions spatio-temporelles d'entités territoriales statistiques.

Mots-clés: représentation des connaissances, géovisualisation, Web Sémantique, règles sémantiques

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

1.1 Context: the Semantic Web and geospatial data

Semantic Web technologies originally targeted the description of Web resources. This scope has since broadened, allowing the Semantic Web stack (RDF, RDF-S/OWL, SPARQL, etc.) to be used by knowledge management applications and for conceptual modelling in various domains such as social networks (Bojārs et al., 2008; San Martín and Gutierrez, 2009), data portability (Bojārs et al., 2008), life trajectories (Noël et al., 2017), medical decision support (Lamy, 2017) or scientific research data sharing (Rafes and Germain, 2015).

The widening of this scope, coupled with the increasing availability of data with a geographic dimension (Miller and Goodchild, 2015; Lee and Kang, 2015; Liu et al., 2016), has led to the production of specific means to describe, at a high level, geographic data in the Semantic Web such as the W3C Geo Vocabulary and GeoSPARQL (OGC®, 2012). This was combined with a significant amount of research evaluating the contributions of Semantic Web technologies to encode and exploit geospatial data which, as defined by Linda van den Brink, "*refers to any data that has a location in terms of geographic coordinates within Earth*" (Brink et al., 2018). Key themes of these researches are notably the formalisation of geographical data in a broad sense (Abadie et al., 2010) or of specific domains such as the maritime domain (Vandecasteele and Napoli, 2012), land uses (Montenegro et al., 2012) and transport networks (Houda et al., 2010; Sanchez and Ordinez, 2020) to only cite a few. These key research themes also cover the querying of such geospatial data (Zhang et al., 2015) as well as finding (Wiegand and García, 2007) and integrating heterogeneous geospatial data (Maué and Schade, 2009; Zhang et al., 2010; Prudhomme et al., 2020).

In geographic information science (GIScience), besides knowledge modelling and knowledge representation, one of the main challenges remains the design of geovisual

interface for rendering information to the users. Indeed, geospatial information and the interfaces for viewing and interacting with it, commonly referred to as geovisualisation applications, play an indispensable role in the understanding of various spatial phenomena and in the decision-making processes involving this information (MacEachren et al., 2004).

The topic of combining traditional geovisualisation approaches with knowledge system, in order to produce knowledge-based geovisualisation, was identified early by Fabrikant (2001), but when geospatial semantics issues are tackled (e.g. by Janowicz et al., 2013 and by Hu, 2018), the use of Semantic Web technologies for the specific purpose of building geovisualisation is rarely addressed. Concrete proposals in the field of Semantic Web have initially focused on its use as a way of formally representing cartographic knowledge (Iosifescu-Enescu and Hurni, 2007; Smith, 2010; Brus et al., 2010; Ruzicka et al., 2013; Penaz et al., 2014). In contrast, few exist that also allow to portray data available in Semantic Web formats (León et al., 2012; Varanka and Usery, 2018; Huang et al., 2020; Huang and Harrie, 2020). This reflects the fact that RDF does not define a presentation model for the data and that there is no obvious presentation for Semantic Web data (Pietriga and Lee, 2009). However, with the rise of the Semantic Web and the recommendations of best practices in terms of publication, it is to be expected that geospatial data will be more and more available on the Semantic Web (Brink et al., 2018). While it is entirely possible, applying traditional methods of geovisualisation creation to Semantic Web data can be particularly burdensome. Indeed, this involves transforming the Semantic Web data to a classical vector format (such as GeoJSON or KML) supported by the various tools of the geovisualisation ecosystem, which is less straightforward than between two classical vector formats. Above all, this shortcoming is not only technical as it can result in a loss of the semantics associated with the data, which can be particularly detrimental.

In this thesis, our special interest in knowledge-based geovisualisation approaches and in particular with Semantic Web technologies is thus motivated by one main factor. The renewal of the geospatial socio-technical system is leading to changes in practices regarding the dissemination and the publication of geospatial data on the Web, or at least to changes in practice recommendations (Brink et al., 2018). These changes translate into new needs of methods and tools to visually represent the data published using the new formalisms that are those of the Semantic Web.

In our work, we explore how to take advantage of Semantic Web technologies with the aim to facilitate the creation of truly knowledge-based geovisualisations. Indeed, we postulate that we can reason, thanks to the models that describe a real-world domain, about the data to be depicted in order to translate them efficiently and quickly into a geovisual portrayals. Using the description of the concepts that make up a specific domain, we succeed to specify at a declarative level how to create a geovisualisation that is particularly tailored and embeds some kind of intelligence, ultimately allowing to lighten the cognitive process linked to the geovisual analysis of this support. As such, we believe that we can propose a process that can help to build a map that will favour its interpretation. In addition, providing a method

that directly addresses domain-specific semantics and which targets an ecosystem already used to encode knowledge of various domains can be beneficial to a persistent challenge of geovisualisation which is the transfer of methods and processes between application domains (Çöltekin et al., 2017). Finally, we fit perfectly into the goals of the Semantic Web and Linked Data (Berners-Lee, 2006) if we can give geospatial data an additional meaning, a geovisual meaning, that is understandable by both machines and humans, in the form of linked and reusable RDF knowledge.

The interest expressed here is also part of an existing field which aims to facilitate the creation of geovisualisation from general spatio-temporal concepts and by exploiting the properties of existing domain or data models (Moisuc, 2007; Zaki et al., 2011; Bimonte, 2014; Huang and Harrie, 2020; Huang et al., 2020). Indeed, existing works are already interested in the generic creation of adaptive geovisualisations, but they give an important part of their focus to the modelling of spatial information (Moisuc, 2007) and they put less emphasis on their versatility in an ecosystem with growing and novel needs such as the Semantic Web (only Huang and Harrie, 2020 and Huang et al., 2020 belong to some extent to this category). We distinguish ourselves by choosing to join an ecosystem that already provides basic building blocks for geospatial with GeoSPARQL. In addition, our work can be based on proven ideas in terms of map feature description (OGC®, 2007) and feature selection (OGC®, 2007), although the latter is not transposable to Semantic Web technologies. Therefore we propose in this thesis to directly exploit the geospatial semantics of an existing model, targeting RDF geospatial datasets described by ontologies, in order to build adaptative geovisualisation interfaces that are themselves entirely described in RDF.

For this purpose we have a prime case of application with the CHOUCAS project (presented in the next section) in which Semantic Web technologies are used to encode domain knowledge with a geospatial component and a human reasoning component. This seems particularly suitable for testing the creation of knowledge-based geovisualisation aimed at supporting or alleviating the reasoning process of its users.

1.2 Specific needs related to locating victims in the mountains: ANR CHOUCAS

CHOUCAS is an interdisciplinary research project (Olteanu-Raimond et al., 2017) aiming to respond to a need expressed by the PGHM (*Peloton de Gendarmerie de Haute-Montagne* - high-altitude rescuers of the French National Gendarmerie) of Grenoble to help localising victims in mountain area. The project is funded by the French National Research Agency[1] and is divided into 5 work packages (WP) involving several partners:

- the MEIG team from the LASTIG (*LAboratoire en Sciences et Technologies de*

[1] ANR-16-CE23-0018: Heterogeneous data integration and spatial reasoning for locating victims in mountain areas – CHOUCAS.

l'Information Géographique pour la ville intelligente et les territoires durable), working on fuzzy spatial reasoning (WP3) and service oriented architecture (WP5),
- the STeamer team from LIG (*Laboratoire d'Informatique de Grenoble*), working on knowledge representation (WP1) and geovisualisation tools (WP4),
- the Proba/Stat team from the LMAP (*Laboratoire de Mathématiques et de leurs Applications de Pau*) working on mining spatial data from the web (WP2) and service oriented architecture (WP5),
- Grenoble PGHM as contributor and end user of the methods developed within the project.

The PGHM is composed exclusively of high-altitude rescuers. One of their core missions is to bring help to victims in response to alerts, whose one of the specificities is the terrain of intervention, high-mountain area, requiring a helicopter ride in more than 85% of cases[2]. Another specificity relates to the process leading to the identification of the location of the victim. Generally, the alert is triggered by telephone. In some cases, it is possible to locate the victim directly and with precision, for instance using the GendLoc tool[3]. However, in other situations, especially when no technology such as GNSS is available, mountain rescuers who receive the emergency call have to process by their own to localise the victim. The research phase starts with the exploitation of what the caller (the person who triggers the alert, who is not necessarily the victim) says, trying to describe where he or she is currently or which path has been followed until there. An example of such descriptions is: *"I left two hours ago from the Refuge de la Pra and walk towards the Cascade de l'Oursière. Now, I see a lake just below"*. All these pieces of information are clues analysed by the rescuer who looks in different data sources for corresponding elements that could help to locate the victim. Data sources may be printed maps or digital maps or guidebooks. The alert processing also has significant reliance on the expertise of the rescuers and on their knowledge of the terrain. Several elements complicate the processing of the alert: the caller often describes the current position and the path followed with imprecise relative positions in relation to sometimes ambiguous reference objects (ambiguity can be in the naming of these objects, in their description, in the expression of the relative position itself). The data needed by rescuers are heterogeneous, multi-source, dispersed and insufficiently structured so they are efficiently queried.

The issue of locating a victim with the help of a digital medium reflects the evolution of the organisation of emergency services, which has undergone two main changes in the last 30 years:

- the use of mobile phones by victims, allowing them to alert the emergency services themselves (Johnson, 2004), coupled with the existence of a telephone network allowing them to initiate a call to an emergency service (such as 112 or 911) without depending on the specific network of their operator (Weinlich et

[2]According to the 2015-2019 activity reports of the French National Observation System for Mountain Safety (Système National d'Observation de la Sécurité en Montagne - SNOSM).

[3]https://fr.wikipedia.org/wiki/GendLoc

al., 2018), as well as the growing presence of GPS/GNSS devices within mobile phones.
- the use of digital media by rescuers as a result of the increased availability of geospatial information and the development of GIS techniques.

In addition, this seems to be part of a trend that gives an increasingly important place to geospatial technologies, as a complement or replacement of traditional media in emergency situations. This is embodied both in works such as *Successful Response Starts with a Map* (NRC, 2007) and in work to improve rescue assistance in the aerial (Abi-Zeid et al., 2010; Riveiro, 2016), maritime (Marven et al., 2007) or terrestrial (Konstantopoulos et al., 2006; Konstantopoulos et al., 2009; Michal and Robert, 2015) domains. The choice of geovisualisation as a support for the reasoning of rescuers is part of this overall context. In addition, this choice is part of the logical continuation of the currently in effect practices of the rescuers who consult paper documents (such as maps and guides) and already use a Web application with a simple user-interface that allows them to display topographic maps and to overlay some types of thematic layers on them.

When tackling the challenges of the CHOUCAS project, a lack of a common vocabulary between rescuers and members of the project was noticed, specially to formalise the description, in a way that can be understood by humans and machines, of the different conceptual aspects mobilised during the processing phase of the victim search alert. Regarding the context of this thesis, such a formalisation could serve as a basis for the creation of a geovisualisation interface dedicated to this task of localisation. In the case described here, the modelling to be carried out is not intended to automate the entire task that is performed by the rescuers as it is desirable to let them take decisions by themselves. Nevertheless, some intermediate subtasks, particularly in relation to the processing and to the portrayal of geospatial information, can be automated. Above all, we believe that this modelling should describe in detail the elements that are handled by the rescuer. It is then thanks to the use of particularly tailored geovisual portrayals that we will accompany his or her reasoning towards decision making.

This thesis is funded by the ANR CHOUCAS project which provides a concrete case of application: the development of a geovisualisation tool that supports the reasoning of the rescuers. This work addresses concerns from both WP1 and WP4 and its originality lies in the fact of positioning itself at the interface between the two realms: that of knowledge representation and that of geovisualisation to support the reasoning of the rescuer.

1.3 Positioning and postulate of the thesis

This thesis work is positioned in the field of **geovisualisation** and particularly at the interface of two disciplines: **computer sciences**, and **geographic information sciences**. This work is also carried out with a particular sensitivity brought to

cognitive sciences taking into account fundamentals of attention and reasoning.

1.4 Research questions

Our general context introduced so far is the one of the creation of intelligent, knowledge-based, geovisualisations. This raises several questions that are at the origin of this work:

(i) How to formalise the concepts that make up a geovisualisation interface (map and other components, data symbolisation, etc.) and the links between them?
(ii) With regard to the previously identified concepts, which geovisual techniques (graphical semiology, interactions, etc.) are appropriate when it comes to support a user's geovisual reasoning ?
(iii) How can we automate the creation of geovisualisation powered by the previously identified techniques while making a reusable and generic proposal ?

These questions are legitimised by the fact that there seems to be little literature on the explicit link that can be made between the spatial dimension of these models and their effective exploitation in a geovisualisation interface for decision support.

As such, our research question is the following:

How to facilitate and automate the creation of intelligent geovisualisations using semantics of existing data and Semantic Web technologies ?

We believe that it is useful and necessary to focus on the explicit link that can be made between models with a geospatial dimension and geovisualisation interfaces that would exploit the semantics of this model in several ways to allow the construction of geovisualisation tools for decision support. The obtained geovisualisation should include and use this semantics in order to accompany the user's reasoning thanks to the geovisual encoding of the semantics previously existing in the model.

This idea is based on the strong postulate detailed in the previous sections: we believe that it is possible to favour the implementation of a map relevant towards its goals, by fostering, from the preparation of the geovisualisation, the description at the semantic level, of the way of visualising the data. The implementation of this idea implies several prerequisites: it will indeed be necessary to be able to describe, with the help of controlled vocabularies compatible with the models we wish to exploit, all the elements that make up a geovisualisation interface and in particular its central element, the interactive map. These elements are detailed in what follows.

1.5 Objectives and contributions

The first objective of the thesis is to formalise, in the form of ontological vocabularies, the various elements that make up a geovisualisation (its components, the way data appears in it, the interactions that can take place in it, etc.) and which enable the implementation of geovisual techniques useful to support human reasoning. The first contribution is a stack of compatible OWL ontologies for this purpose: six of them that we propose and four that we reuse and sometimes adapt from the literature.

Once these elements have been completed, the second objective is to create a specification document model to describe the geovisualisation one wishes to create from a semantic model. This specification document must allow rich selection mechanisms that are adapted to RDF. In addition, it must be possible to use it directly to obtain the corresponding geovisualisation. The second contribution is therefore a framework, instrumented by semantic rules using the Shapes Constraint Language - Advanced Features (SHACL-AF), which transforms the specification document, written entirely in RDF by the user of our approach, into a knowledge graph containing all the necessary links between the data and the way they should be portrayed. This graph is published as a SPARQL endpoint so that semantic queries can be performed. Another of our contributions is a generic client, in the form of a Web application, which allows the graph published by our framework to be transformed into the corresponding geovisualisation interface.

A final objective is to ensure that this system works properly and is usable for both CHOUCAS project data and Semantic Web data. The final contribution is to demonstrate this usability, in the form of various specification documents that will be used to generate geovisualisations that portray data from various models using our approach.

Beyond the question we asked ourselves (Section 1.4), our contributions can be valuable at various levels in GIScience and computer science:

- By contributing to propose transferable methods and processes for the creation of geovisualisation (this echoes a persistent challenge identified by Çöltekin et al., 2017).
- By contributing to ease the creation of useful and usable geovisualisation tools (this also echoes a long standing challenge as stated by Andrienko and Andrienko, 2006).
- By linking geovisual semantics to existing geospatial Semantic Web data (as opposed to hiding it in the code of an application) in order to give them more meaning (this is in line with one of the objectives of Linked Data as defined by Berners-Lee, 2006).

1.6 Outline

The document contains two parts, respectively dedicated to the state of the art and its analysis, and to the presentation of our contributions and their validation.

We begin Part I, by Chapter 2 in which we review the various useful geovisualisation techniques as well as the insights offered by the scientific literature regarding the contribution of geovisualisation for decision support. This chapter also focuses on the methods for creating and implementing geovisualisation interfaces and more precisely those used for implementing the techniques presented above. We also emphasise the creation methods that allow formal and reusable encoding of visualisation choices.

In Chapter 3 we look at the solutions offered by Semantic Web technologies for formally representing knowledge. A comparison between the existing possibilities (both in terms of storing and querying geospatial knowledge and in terms of applying semantic rules to this knowledge) is proposed in order to guide our choices.

Chapter 4 synthesises the knowledge accumulated from the work discussed in the previous two chapters and focuses specifically on the work available in the literature that aims to formalise or create geovisualisations using Semantic Web technologies. From this synthesis, which can be seen as a requirements analysis, we can identify the shortcomings in terms of geovisualisation creation and present the challenges our proposal faces.

The Part II starts with Chapter 5 that describes our first contribution: it deals with the formalisation of the elements that make up a geovisualisation interface. We focuses on the main media of the geovisualisation: the interactive map, and we provide a formal representation of the necessary concepts as ontological vocabularies, from the most general, the geovisualisation application, to detailed elements such as the graphical properties of the geometric primitives to be displayed on the map.

Chapter 6 continues our proposal by introducing a framework, CoViKoa, that uses the previously defined vocabularies to specify, in a descriptive approach and in RDF, a geovisualisation interface. This framework is composed of semantic rules which allow the passage from a descriptive specification document to a knowledge graph containing all the elements necessary for the creation of an intelligent geovisualisation interface. This knowledge graph is exploited by a generic interface prototype.

Next we use this framework in a case study applied to the location of victims in mountains environments in Chapter 7. A conceptual model dedicated to the field of victim search in mountain environments is first proposed. This chapter then shows how the descriptive specification of geovisualisation and its processing in CoViKoa together allow the encoding of powerful elements to support the user's reasoning.

We validate the genericity of the approach in Chapter 8 by using our framework

in order to create a geovisualisation for an existing model whose data is located in the Web of Data and that is dedicated to describe changes in territorial statistical nomenclatures.

In Chapter 9, we conclude by discussing the main contributions of the thesis as well as the possible follow-up.

Part I

State of the art

Chapter 2

Geovisualisation for decision support

Geovisualisation, sometimes also called geographic visualisation or even cartographic visualisation, can be a challenging term to define because it inconsistently refers to "*a map, a display type, a process, a technique, a way of using maps, and an academic discipline*" (Çöltekin et al., 2018). However, the realities that this term covers are always related to the interactive and/or dynamic visualisation of a cartographic medium mediated by a computer interface[1].

If cartography, by the static nature of the medium, emphasises communication through the map, geovisualisation emphasises the exploration of the spatial information presented and the construction of knowledge from it by its user. Indeed, because of its interactive nature, geovisualisation gives a more important place to the person for whom it is intended: the reader in cartography becomes the user of a geovisualisation, able to manipulate the data according to his or her needs and hypotheses in order to facilitate his or her reasoning, for example during a decision-making process.

To achieve this, geovisualisation needs to bring in techniques other than those of cartography. As such, leading authors in the field describe geovisualisation as "*a loosely bounded domain that addresses the visual exploration, analysis, synthesis and presentation of data that contains geographic information by integrating approaches from disciplines within GIScience, including cartography, with those from scientific visualization, image analysis, information visualization and exploratory data analysis*" (Dykes et al., 2007, p. 694). As a result, a geovisualisation interface does not necessarily contain only one map, but may contain several and of different types (planar, terrain), and may also mobilise other resources such as diagrams, graphs or tables (Crampton, 2002).

In order to fully understand the techniques used by geovisualisation and the uses

[1] Although it is technically possible to do this on paper - some early definitions mention it - this consideration is very uncommon in the literature.

that can be made of it, we present in this chapter the central concepts of this field as well as contributions from the scientific literature, but also the possible limitations, that specifically address geovisualisation for decision support. As our work is part of the CHOUCAS project, we focus on approaches supporting a spatial reasoning that should lead to decision-making.

Firstly, we look at a central element of geovisualisation: its cartographic aspects. We then tackle the specific techniques of geovisualisation that give it its strength when it comes to supporting human reasoning for decision-making. In addition, we then review existing geovisualisation approaches aimed at assisting in various location tasks, such as the search for lost victims in particular. Since our research objectives aim to help the geovisualisation designer, we will finally present and discuss the existing approaches for developing these geovisualisations as well as how the knowledge about these design choices can be shared interoperably.

2.1 Cartographic aspects of geovisualisation: foundations and general considerations

As stated before, cartographic aspects are among the main components that make a geovisualisation. We discuss here them more specifically as in our context, what will be presented to the user by means of the map is of first importance in his or her reasoning process. In this, we recognise ourselves in the thinking process of by Lambert and Zanin (2020) when they confront the terms cartography, geovisualisation and geomatics. They find that the term cartography offers advantages, in particular that of being a rich and abundant discipline allowing to create meaningful representation. Moreover, they particularly identify the interactive and digital medium as the essential feature of geovisualisation. We agree with this observation and ultimately we also see the map as the main medium of geovisualisation. It is therefore natural that we should devote particular attention to it.

In this section, we begin a review of the various basic elements dealing with the visual encoding of geospatial information to create maps that make sense to the user. We then present the different types of portrayals that are traditionally used on a map. As these maps are intended to be used in geovisualisations, through a computer-mediated interface, we finally discuss the various aspects relating to interactivity with the map: this will symbolise our passage into the realm of geovisualisation.

2.1.1 Visual encoding of information

In order to depict the real-world information to be represented on a cartographic material, it is necessary to use **map symbols**. These are graphic forms that can be of different kinds: point markers, lines, areas or texts for example. In order to transcribe different kinds of information and to give meaning to the map produced,

these symbols can vary in different ways: colour and shape are some of the graphic variables that can qualify these map symbols. The process of representing these symbols on the map is called *figuration* or *symbolisation* (Lambert and Zanin, 2020, p. 64).

The choice of these symbols and their properties is based on a methodology aimed at producing effective and meaningful symbolisation. To achieve this, map designers are interested in how the brain interprets these symbols. Applying Charles Sanders Peirce's theory of semiosis to cartography, in order to make sense from map symbols, map readers must be able to establish a connection between the graphic shape on the map, a general concept and a real world phenomenon (Peirce, 1907).

The principles of graphic semiology specific to cartography, and particularly to statistical cartography, were formalised by Jacques Bertin in his book "Sémiologie Graphique" (Bertin, 1967). He defines 6 *retinal variables* used for point, linear and zonal primitives, which are also the primitives of 2D vector objects used in GIScience. They are as follows: *size*, *value*, *grain*, *colour*, *orientation* and *shape*. He also defines a *planar variable* that is the *position* (this variable is sometimes directly associated with the concept of retinal variable explaining why we sometimes refer to the 7 variables proposed by Jacques Bertin).

These retinal variables were studied according to their perceptual properties, depending on whether they allow the perception of a quantitative dimension (symbols are perceived as proportional to the value depicted), an order (symbols are perceived as ordered, allowing the reader to sort the entities depicted), an association (symbols are perceived as similar) or a selection (the symbols are perceived as different, allowing the reader to form families). The results of this work (Figure 2.1) establish conventions allowing the best use of the various retinal variables according to the type of data to be depicted and the meaning to be given to the map.

The concept of retinal variables, which has come to be known as **visual variables**, describes the way in which graphic symbols can vary in order to visually convey variations in the data or phenomenons depicted. Since Jacques Bertin's work, other authors have been interested in visual variables and have also made proposals. A major difference between Bertin and the following authors concerns the separation of the visual variable *colour* into two variables *hue* and *saturation* (sometimes also called *intensity*). This is found as early as 1974 in the work of Morrison (1974) as well as in the majority of the works that follow. It should be noted that some of these visual variables have been introduced with the aim of presenting other phenomena than those studied so far, for example with MacEachren (1995) which suggests the use of three new visual variables to depict uncertainty: *crispness*, *transparency* and *resolution*. Other proposals result notably from the progress made in the process of creating maps through computerisation and some authors propose the use of *perspective height* (Slocum et al., 2009; Lambert and Zanin, 2020). These conventions are included and sometimes revisited in the majority of works dealing with thematic mapping and geovisualisation (such as MacEachren, 1995; Krygier and Wood, 2005; Dent et al., 2009; Slocum et al., 2009; Kraak and Ormeling, 2010; Tyner, 2010; Lam-

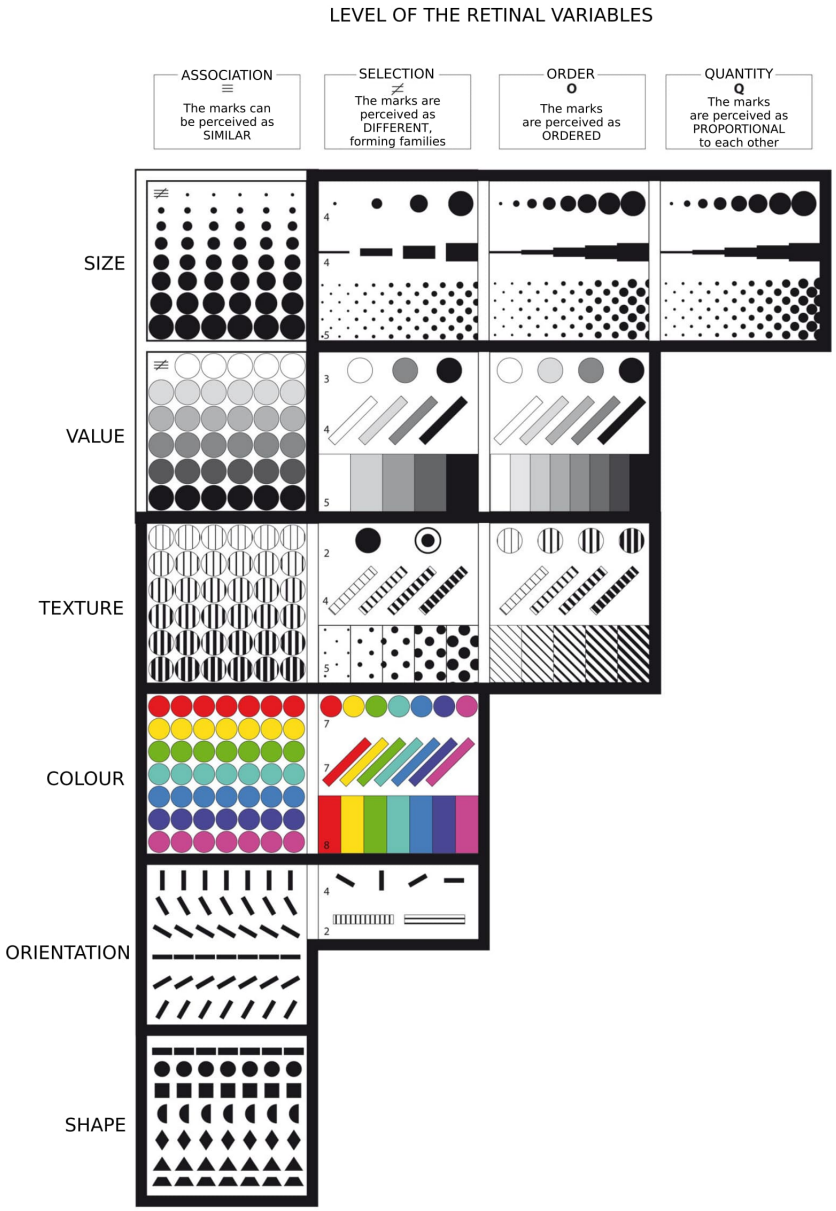

Figure 2.1: Retinal variables according to Jacques Bertin. Source: adapted from the French figure "*Niveau des variables rétiniennes*" in Bertin (1967).

bert and Zanin, 2020). In order to have an synthetic overview, we provide a table of the main visual variables encountered in the literature (Table 2.1).

Static visual variables	**Author, date**
Size	Bertin, 1967; Morrison, 1974; MacEachren, 1995; Krygier and Wood, 2005; Dent et al., 2009; Slocum et al., 2009; Tyner, 2010; Kraak and Ormeling, 2010; Lambert and Zanin, 2020
Shape	Bertin, 1967; Morrison, 1974; MacEachren, 1995; Krygier and Wood, 2005; Dent et al., 2009; Slocum et al., 2009; Tyner, 2010; Kraak and Ormeling, 2010; Lambert and Zanin, 2020
Value	Bertin, 1967; Morrison, 1974; MacEachren, 1995; Krygier and Wood, 2005; Dent et al., 2009; Slocum et al., 2009; Tyner, 2010; Kraak and Ormeling, 2010; Lambert and Zanin, 2020
Colour (hue + saturation)	Bertin, 1967
Orientation	Bertin, 1967; Morrison, 1974; MacEachren, 1995; Krygier and Wood, 2005; Dent et al., 2009; Slocum et al., 2009; Tyner, 2010; Kraak and Ormeling, 2010; Lambert and Zanin, 2020
Texture	Bertin, 1967; Morrison, 1974; MacEachren, 1995; Krygier and Wood, 2005; Dent et al., 2009; Tyner, 2010; Kraak and Ormeling, 2010; Lambert and Zanin, 2020
Position	Bertin, 1967; MacEachren, 1995; Krygier and Wood, 2005; Dent et al., 2009; Slocum et al., 2009; Tyner, 2010; Kraak and Ormeling, 2010
Hue	Morrison, 1974; MacEachren, 1995; Krygier and Wood, 2005; Dent et al., 2009; Slocum et al., 2009; Tyner, 2010; Kraak and Ormeling, 2010; Lambert and Zanin, 2020
Saturation	Morrison, 1974; MacEachren, 1995; Krygier and Wood, 2005; Dent et al., 2009; Slocum et al., 2009; Tyner, 2010; Lambert and Zanin, 2020
Arrangement	Morrison, 1974; MacEachren, 1995; Dent et al., 2009; Slocum et al., 2009; Tyner, 2010
Crispness / Focus	MacEachren, 1995
Transparency	MacEachren, 1995
Resolution	MacEachren, 1995
Spacing	Slocum et al., 2009
Perspective height	Slocum et al., 2009; Lambert and Zanin, 2020

Table 2.1: Visual variables from the literature over the last fifty years. Adapted from Neuville et al. (2016), itself inspired from Halik (2012).

These conventions are particularly suitable for statistical or thematic cartography (e.g. the portrayal of socio-economic data). However, this is not the only use of visual variables. Indeed, they can be found to some extent in the field of cartography for urban planning, for example in land-use maps which use colour selectivity to distinguish broad categories, value ordering to suggest differences in density, and texture-structure and orientation for sub-categories (Steinberg, 2000). In addition, other of these codes can be found in topographic maps, which are another important

type of map encountered: they are intended to depict, with precision, the elements of the terrain that are for example relief, hydrology, vegetation, administrative areas and various human-made elements such as buildings, transportation routes, etc. These topographic maps, which are generally characterised by their large scale, also adopt common conventions to facilitate their reading and understanding: colour selectivity to distinguish differences in land-use with intuitive colours (wooded areas in green, water surface in blue, etc.), the size of the symbol (e.g. to symbolise differences in the importance of the road network represented, or the difference in importance of the administrative entities encountered), etc.

All these principles have been formalised with the intent of enabling the reader to interpret the information on the map correctly. As such, cartography is already, by nature, a discipline that takes into account cognitive sciences and aims to trigger the appropriate mechanisms in its reader (intuitively understanding the differences between absolute values, being able to order the values of a ratio or an ordinal qualitative variable, etc.).

Finally, in order to allow an accurate interpretation of the elements of the map, many authors are interested in the question of the perception of colours by the reader. Indeed, a part of the population suffers from colour vision deficiency (also called colour blindness or dyschromatopsia). This is a disorder of the perception of some colours or of the difference between colours, which can manifest itself in 8 different ways, each of which has its own impact on colour perception. Colour blindness can profoundly alter the way in which colour ranges are perceived (confusion between colours intended to distinguish entities, failure to perceive the colour progression of a palette, etc.) and it is important to take these readers or users into account when making maps. In the field of cartography and geovisualisation, these problems have been tackled in particular by the famous work of Brewer et al. (2003) in which are proposed colour palettes of different schemes (diverging, sequential or categorical) on a given number of steps[2]. Some of these palettes are labelled as colour blindness compatible. Light and Bartlein (2004) also address this issue and give advices on designing various schemes of colour palettes for colour-blind individuals as well as practical examples of palettes that meet these constraints.

2.1.2 Cartographic portrayals

The correct manipulation of visual variables in order to construct cartographic representations that make sense to the reader has led to the creation of solutions, in the form of cartographic portrayals, to represent given phenomena. Their realisation requires several manipulations of the geographical information beforehand.

For example, the literature unanimously recommends the use of choropleth maps to represent relative quantitative data (population density, unemployment rate, etc.) on zonal entities such as administrative divisions or regular grids for example. This is

[2]These palettes can be found on the dedicated website: https://colorbrewer2.org/.

done, for example, by colouring the regions in question according to ordered colours. Choropleth maps are thus intended to allow the comparison of a statistical measure between different regions. The discretisation step that has taken place facilitates the reading of the map and highlights the major spatial structures. The communicative role of cartography takes on its full meaning with this type of map.

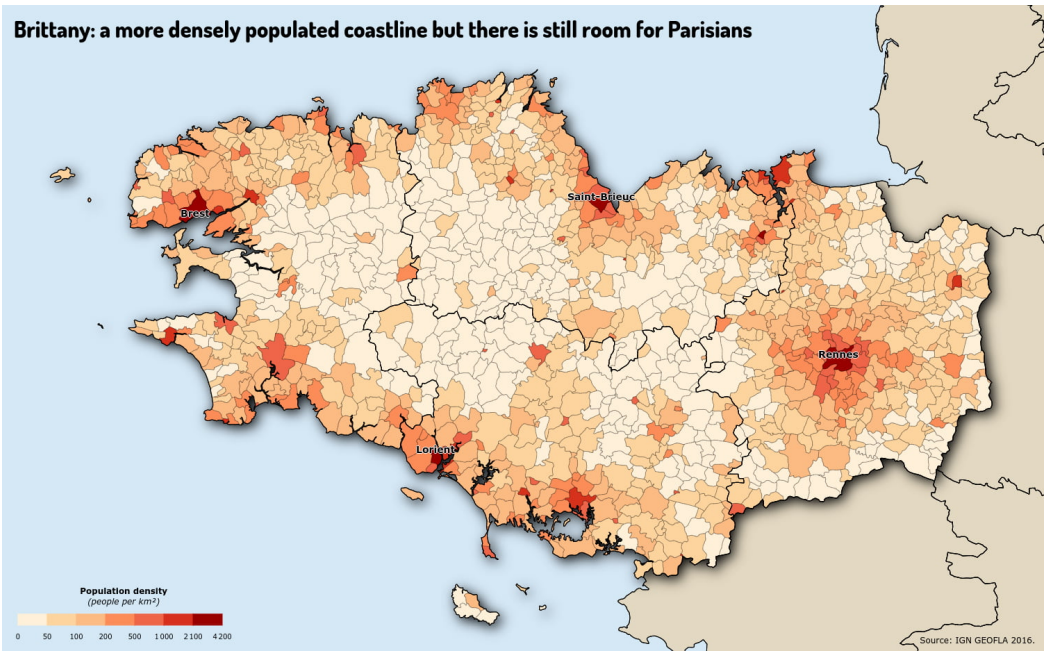

Figure 2.2: Example of choropleth map: Population density in Brittany.

Thus, the *choropleth map* solution (Figure 2.2) requires the implementation of several successive treatments necessary to find suitable class boundaries, map the continuous values to these discrete classes, and colour the entities with the colour that corresponds to the class in question. In this specific case, there is no geomatic process to create new geometries. In other cases (e.g. making a discontinuity map or a cartogram, two other well-known cartographic solutions) an additional step producing new geometries may come into play. We do not review all existing solutions. Extensive lists of them can be found in the literature, for example in Lambert and Zanin (2020) for thematic mapping or in Dykes et al. (2007) and Kraak and Ormeling (2010) for geovisualisation. These solutions apply to the joint representation of several variables of different type or of same type and to the representation of data with a temporal dimension, for example. Here we mainly want to outline the general process leading to the creation of a given portrayal for a given dataset. Indeed, the creation of a cartographic representation, and this also applies when using geovisualisation, may require the creation of new attribute values (for example, here concerning the class of each entity), the creation of new geometries (for example, the calculation of the centroid to locate a proportional symbol, or an anamorphosis calculation in the case of a cartogram) as well as the choice of appropriate colour progressions.

These processes are well known and can be chained together as part of the geo-

graphic information processing chain to produce each type of portrayal. This element is of major importance: it is because there is a precise way of making these types of maps (e.g. the choropleth map or proportional symbols) that it is possible to develop tools dedicated to implementing these cartographic methods (PhilCarto[3] and Magrit[4] softwares as well as mapsf[5] R package illustrate this point). We will see in chapter 4 how authors have tried to formalise these elements and how we can mobilise them in our approach in order to produce correct and meaningful representations.

2.1.3 Other design elements

The *dressing* of the map (this expression in English is coined by Lambert and Zanin, 2020, p. 144) is the step where the map is finalised and a number of elements are added around the map to allow the reader to get a sense of the map and understand it. These elements are of different kinds: title, legend, sources, orientation, scale, etc. Some of these elements are indispensable (e.g. correct display of data sources and attributions) while others are optional and depend greatly on the context (e.g. orientation).

This dressing issue is central to cartography but this is one of the points on which geovisualisation differs from cartography. When it comes to geovisualisation, this stage tends to disappear, as noted by Mericskay (2016), or at least to be substantially modified in our view. Indeed, traditional elements such as the display of scale and sources are easily integrated into the map and are frequently encountered in geovisualisation in their dynamic version. Other elements, however, become less relevant or tied to the integration of the map within an application or Web page. For example, the map title and legend can be integrated elsewhere in the geovisualisation interface, and the legend need sometimes to be modified dynamically according to user actions.

Finally, since we are beginning to discuss the specifics of geovisualisations, it is worth noting that the conventions presented above are now sometimes mixed together, or even largely deconstructed by neophyte users, to produce what are commonly called *cartographic mash-ups*. Cartographic mash-up (or sometimes *mapping mash-up*) refers "*to hybrid web applications that combine data or software from two or more sources*" (Monmonier, 2007).

This phenomenon became widely democratised with the availability, from 2005 onwards, of solutions such as Google Maps, which allow users to display their data on the company's base-maps. This possibility, which was quickly rivalled by a multitude of services and technologies offering similar functionalities, led to the appearance of numerous web-maps combining a base-map from Google or OpenStreetMap with data from one or several sources and displaying mainly points (for example in the form

[3] http://philcarto.free.fr/
[4] http://magrit.cnrs.fr/
[5] https://riatelab.github.io/mapsf/

of marker corresponding to specific amenities). The points represented in this way are then generally referred to as Points of Interest (POIs). These base-maps, typical of maps made available on the Web, are one of the foundations of what Mericskay (2016) describes as the *geoweb*. These maps, far from being limited to the standard conventions of topographic maps, also have strong aesthetic appeal (Figure 2.3).

Figure 2.3: Example of geoweb base-maps. From left to right: OpenStreetMap standard†. Toner† by Stamen Design. Watercolor‡ by Stamen Design. Terrain by Stamen Design†.

†Data by OpenStreetMap, under ODbL.
‡ Data by OpenStreetMap, under under CC BY SA.

In the end, cartographic mash-ups made on these base-maps are not only intended to display points. Like cartography, where the contextualisation of data is carried out with the help of specific background layers, geovisualisation frequently uses these geoweb base-maps to carry out the contextualisation. More and more designers are now displaying statistical data on top of this type of background map, combining conventions of statistical maps and those of topographic maps in geovisualisation interface (this can be seen for example in the following of this chapter in Figures 2.8 and 2.12).

2.1.4 Interactivity

We have seen that *interactivity* is one of the characteristic elements of geovisualisation. To get a broad picture of the concept, in geovisualisation, interactivity can be firstly defined as "*a system that changes its visual data display in response to user input*" (Crampton, 2002). The dialogue that takes place between the user and the map, mediated through a computing device, is defined as an *interaction* (Roth, 2011). Generally speaking, these interactions make it possible to implement the design principle for the creation of data visualization systems described by Shneiderman (1996) as "*overview first, zoom and filter, then details-on-demand*".

Theses interactions are the various techniques, some of which are well known, for zooming (changing the visualisation scale), panning (dragging the map), filtering the displayed data (both at the level of layer selection and within a layer), modifying the displayed representations (both automatically according to the zoom level and at the request of the user), querying the data (whether in the form of displaying metadata

or highlighting features with specific characteristics) and linking the map to other resources (such as graphs, diagrams and tables).

Various authors have endeavoured to list and organise the different existing techniques for geovisualisation. Roth (2012) organises the various taxonomies of interaction primitives that can be found in the scientific literature. He identifies three dominant approaches to classify these works: "*(1) an objective-based approach, compartmentalizing cartographic interaction according to the kinds of tasks the user may wish to complete with a cartographic interface; (2) an operator-based approach, compartmentalizing cartographic interaction according to the unique cartographic interfaces that make manipulation of a cartographic representation possible; and (3) an operand-based approach, compartmentalizing cartographic interaction according to characteristics of the digital/virtual object with which the user is interacting*" (Roth, 2012). These categorisations highlight the different dimensions that are present in the concept of interaction when it comes to qualifying its primitives.

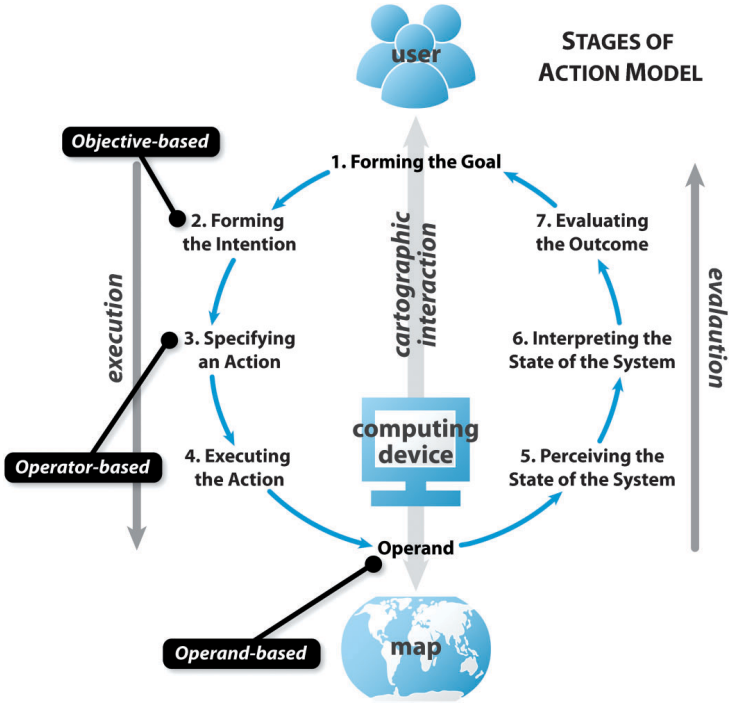

Figure 2.4: The stages of action model and the three O's of cartographic interaction. Source: Image from Roth (2012).

They are also worthwhile in the context of a proposal such as ours that is twofold: to assist the designer of geovisualisation to accompany the reasoning of the end user. We will thus build on these considerations when it comes to identifying interactions that can be implemented in our proposal.

2.2 Geovisual techniques to support human reasoning

In the previous section we gave a quick overview of the constituent elements of geovisualisation, from the conventions it borrows from cartography to the interactivity that makes it unique. We have also seen that geovisualisation, by its definition, emphasises the construction of some knowledge by its users. The process resulting from the use of a geovisualisation and leading to a reasoning, for example prior to decision-making, is designated by different authors as *geovisual analysis* (Andrienko and Andrienko, 2007; Kraak and Ormeling, 2010; Robinson et al., 2017).

This geovisual analysis is made possible by various display and data interaction techniques that are specific to the discipline. Some of these techniques, of interest either because of their versatility or because they would be useful for CHOUCAS, are presented in this section.

2.2.1 Multi-view approaches

Approaches combining several maps or combining a map with tabular or data-visualisation resources are classic in geovisualisation and are particularly well suited to the visual data exploration role of geovisualisation (as stated in the definition of Dykes et al., 2007 mentioned above for example).

The display of two synchronised maps is common to facilitate the comparison of data, for example across the temporal dimension (Figure 2.5). It is also possible to encounter the presence of a map view synchronised with a 3d scene depicting the terrain (Figure 2.6). In the first case, it is frequent that the synchronisation specifically targets pan and zoom actions (moving on a map moves the other map) whereas, in the second case, it is frequent that the synchronisation targets the display of overlayed layers (adding data is reflected on the map and on the 3d scene).

Approaches integrating a cartographic view with other data-viz or info-viz views are common, especially for exploring socio-economic or mobility data and for monitoring different types of activities (Figure 2.7). These approaches, sometimes known as dashboards, have been widely democratised for their ability to synthesise information in a single computer window (they are for example often used in territorial administrations, or for monitoring of COVID-19 indicators for the general public). They are broadly encountered on the Web thanks to software products (such as Operations Dashboard for ArcGIS by ESRI[6]) that offers tools for creating these dashboards without having to write any computer code (Figure 2.8). The creation takes place in a dedicated interface, the panels are dynamically arranged by the creator and it is possible to choose analysis and portrayal functionalities from a large catalogue. This design approach cover a wide range of use cases but it also have limitations beyond the cost of purchasing the tool licence. The interactions that can be created are nec-

[6] https://www.esri.com/en-us/arcgis/products/arcgis-dashboards/overview

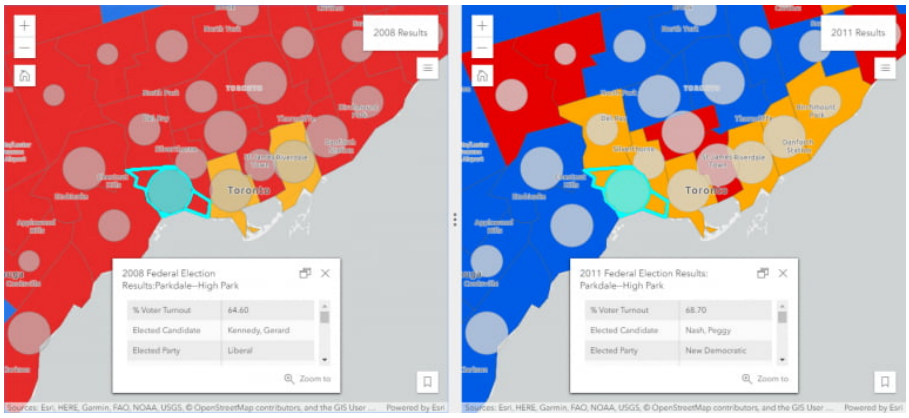

Figure 2.5: Synchronized maps (source: https://www.esri.com/arcgis-blog/wp-content/uploads/2018/12/Pop-Up-768x346.png, available from https://www.esri.com/arcgis-blog/products/arcgis-online/announcements/explore-maps-and-scenes-with-the-compare-configurable-app/, consulted on May 20, 2021).

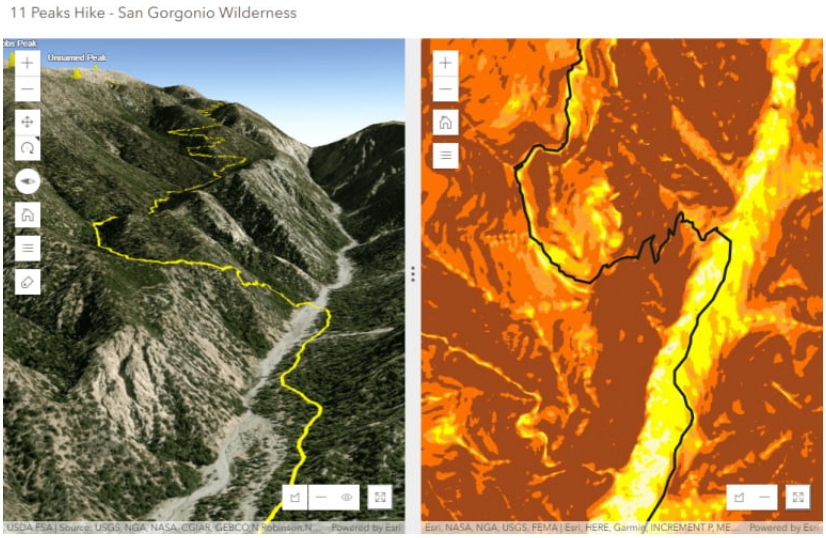

Figure 2.6: Synchronized map and 3d scene (source: https://www.esri.com/arcgis-blog/wp-content/uploads/2018/12/Trails-768x510.png, available from https://www.esri.com/arcgis-blog/products/arcgis-online/announcements/explore-maps-and-scenes-with-the-compare-configurable-app/, consulted on May 20, 2021).

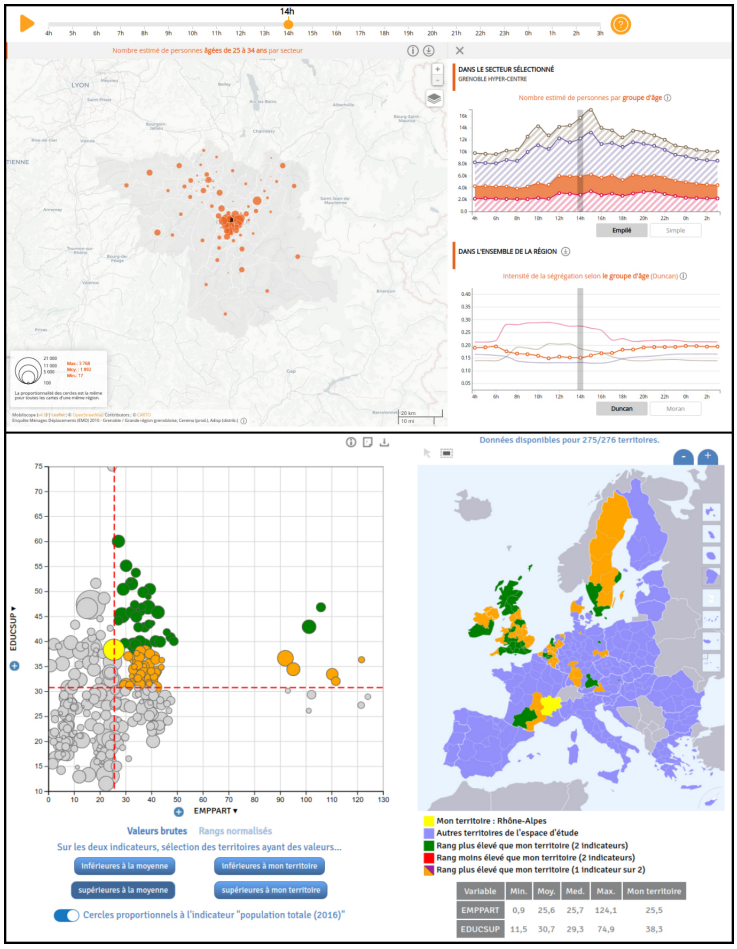

Figure 2.7: Two examples of geovisualisation interfaces with synchronized views (from top to bottom, screenshots from https://mobiliscope.parisgeo.cnrs.fr/fr/geoviz/grenoble and https://riatelab.github.io/rgvzall/build/?europe - accessed on May 20, 2021).

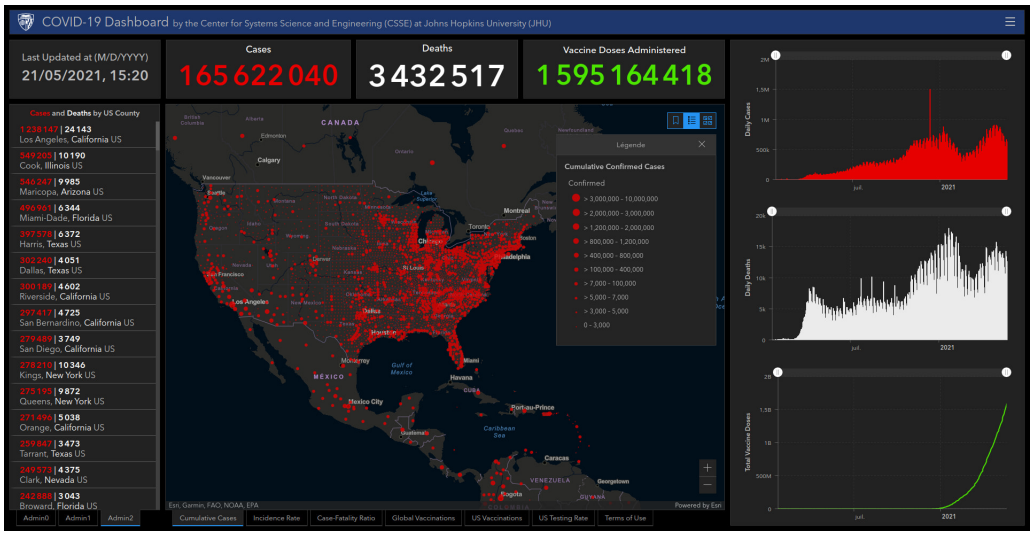

Figure 2.8: Example of geospatial dashboard created with Operations Dashboard for ArcGIS (screenshot from http://covidtracker.com - accessed on May 20, 2021).

essarily more limited than when designed manually with computer code and many choices are not left to the designer (e.g. the possibility to choose "per week" time filtering options without being able to specify the start and end days of a "week"). In addition, it remains difficult to implement another workflow than exploratory data analysis or storytelling.

In contrast, building powerful dashboards that do not have these limitations requires a lot of design work (see for example the work of Menin, 2020 on the creation and evaluation of a complex geospatial dashboard application for the analysis of mobility data). These dashboard approaches will not be discussed further in this manuscript. However, we will keep in mind that these dashboards are operational examples of multi-view approaches and that their success demonstrates the value of having multiple and linked views.

2.2.2 Attention-focusing techniques

In order to support users in their reasoning, it can sometimes be useful to draw their attention to particular elements of the data.

Robinson (2011) is interested in ways of capturing the user's attention in geovisualisation interfaces composed of multiple views and in particular on methods to highlight observations through multiple views. In this context, the author defines the *highlighting* as a transient visual effect that is applied on observations across views of the geovisualisation. This concern is strongly linked to the concept of interaction defined above: indeed, the highlighting obtained is generally a method of presenting

the result of a query made with the cursor on the data represented (sometimes called *brushing*).

The author proposes a set of highlighting methods based on traditional visual variables. He counts 12 of them, which are *hue*, *value*, *saturation*, *focus*, *transparency*, *resolution*, *shape*, *arrangement*, *texture*, *orientation*, *size* and *location* (Figure 2.9). He then proposes three other highlighting techniques, namely the use of a *leader line* (in order to link observations in multiple views), *style reduction* (which works by simplifying the style of the non-highlighted entities) and *contouring* (which allows to visually promote an entity with a 3d-like effect). Each of these techniques is evaluated according to five criteria: being visually salient, applying to all forms, preserving symbology, preserving position and/or size, preserving context. All of this work makes it possible to design effective highlighting techniques according to the specific needs of the context of use.

This type of use is therefore particularly suitable for the exploration of socio-economic data. However, we believe that its scope of application can be broader and can address a less transitory aspect by going beyond the transitory effect of these techniques and use some of them in a more permanent way. Indeed, part of these techniques consists in darkening, blurring or removing details about the area that we want to show less. We believe that this visual effect could be used incrementally, to progressively hide information that is not or no longer useful to show.

In fact, the brain, when it carries out a visual search, has a tendency to use what could be called filters, to suppress what it does not need to see, rather than a mechanism of highlighting (these elements are taught to us by works in the field of neurosciences such as those of Desimone and Duncan, 1995 and Nakajima et al., 2019). However, to make sure that it does not miss anything, it must frequently check that it is not missing information on the periphery of the objects of interest that it is observing. We believe that if we can, with certainty, deactivate objects that are not useful to show, we can considerably lighten this task.

2.2.3 Prioritising or ordering information according to the level of zoom

One of the issues in cartography that is amplified when it comes to geovisualisation is that of the level of details of the background used. Indeed, a large-scale map and a small-scale map cannot depict the same amount of information without degrading its informational quality or readability.

In cartography, this subject (which is however a complicated one with a vast literature) can be dealt with in a one-off manner, during the map creation process. In the context of the dynamic and interactive map that makes up geovisualisation, it will be necessary to have techniques that are themselves dynamic and responsive to changes in scale requested by the user. This can then be handled by geomatic

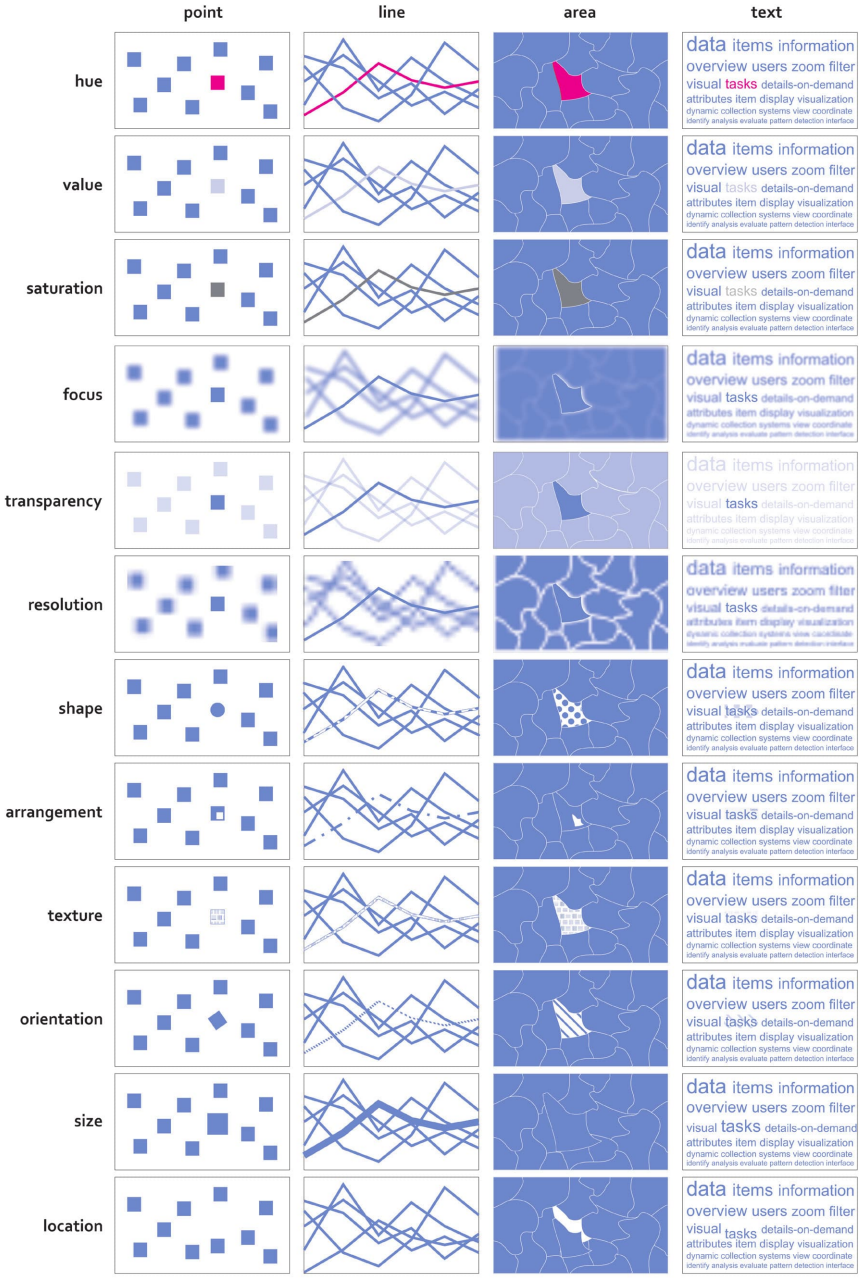

Figure 2.9: Highlighting methods based on common visual variables. Image from Robinson (2011) .

processes, such as the *simplification* (also called *point reduction*) of the lines composing linear and zonal geometries (Figure 2.10), carried out on the fly (and whose results can be cached if necessary), to display them with the appropriate level of generalisation.

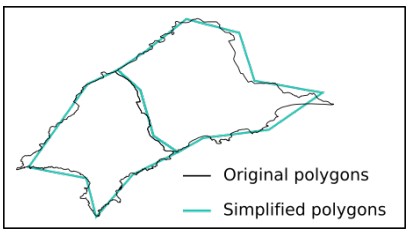

Figure 2.10: Generalisation of polygonal entities by point reduction.

Other data may present additional challenges: simplification is not possible for data with a punctual implementation and it is necessary to use techniques from fields such as information visualisation. The clustering technique aims to address this problem by grouping points according to common characteristics (e.g. spatial proximity and type) in order to reduce the amount of information to be depicted and possibly to show trends in the distribution. In geovisualisation, this technique will be used only at the zoom levels that require it, when the display of all points overloads the map. At higher zoom levels, a classical representation of these points, via individual symbols, can be made (Figure 2.11).

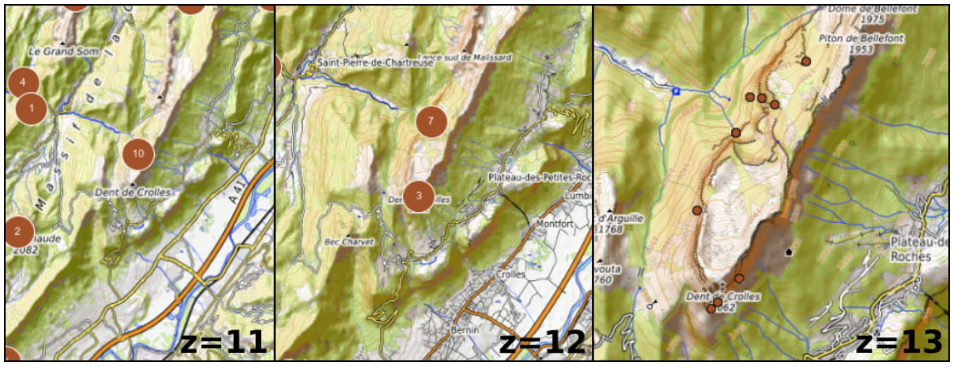

Figure 2.11: Visualisation of points at different zoom levels. The points are grouped together at the lowest zoom levels.

Beyond the issues of generalisation, it may be particularly appropriate to change the information represented on the map according to the zoom level. This is the case, for example, when geovisualisation is used to show socio-economic data for exploratory purposes, in order to facilitate their understanding (Mericskay, 2016). This type of dynamic display can be found on existing geovisualisation interfaces (for example with a choropleth portrayal in Figure 2.12).

Finally, in the specific context of emergency management, we note that Löffler et al. (2007) also use changes in graphic rendering in their geovisualisation in reaction

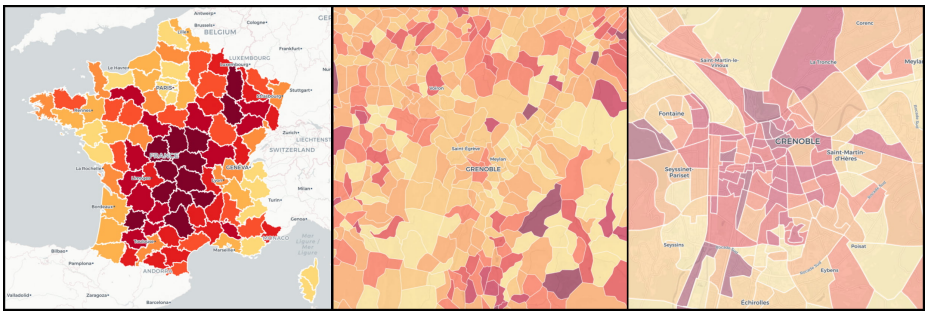

Figure 2.12: Geovisualisation of the share of vacant dwellings at different zoom levels. The territorial level represented varies according to the level of zoom (departmental, municipal, neighbourhood). Source: screenshots from http://map.datafrance.info - accessed on 20/05/2021.

to the user context (such as the zoom level). They consider that these elements contribute to decision support or even to confer some degree of intelligence to their geovisualisation interface.

2.2.4 Interaction with data and obtaining details

Another essential component of geovisualisation is the frequent possibility of interacting with the data underlying the map (e.g. to interact with their attributes or metadata), or even to integrate new data dynamically. This is the case, for example, when the graphical representation of the data cannot convey all the semantics or all the details associated with the object.

One of the most frequent functionalities in geovisualisation is the possibility of obtaining, when clicking on an entity, all the values of its attributes, for example in a popup (this can be seen on Figure 2.5 for example). It is also common to be able to display, when hovering over an entity with the cursor, the name of the entity and the value of interest (e.g. that of the variable on which the current representation is based). These basic interactions are essential in contrast to static cartography and play an important role as they will allow a good understanding of the data, enabling additional information to be accessed and knowledge to be derived from them, knowledge that can be used in reasoning prior to making a decision.

We have also seen previously that geovisualisation is also defined by the multitude of methods belonging to other disciplines that it can mobilise. In this respect, we note that the interactions with the data can take an original form. This is the case, for example, in the work of Löffler et al. (2007), which describes a geovisualisation system for emergency management in which users, rescuers, can ask questions in natural language to the underlying data infrastructure, for example to search for and integrate data filtered according to a particular characteristic (Löffler et al., 2007 take the example of a query such as "show me all landing places within 500m from my

position"). Such a feature that relies on Natural Language Processing techniques, can be essential to support the user's reasoning and in particular to fill in any gaps that he or she may have in handling geographic information. However, such features require significant development and a dedicated data infrastructure and are not available on a plug-and-play basis.

2.2.5 Design consideration: the balance between information and readability

Other elements have been put forward in the literature concerning geovisualisation for decision support. In their work, Wilkening and Fabrikant (2011) study how both the effect of the realism of the cartographic medium and the effect of the time constraint imposed impact the decisions that are made. The task asked to the participants of the experimentation is also close to our thematic since it consists in identifying a site suitable for landing a helicopter (both under varying time pressures, and on different types of maps). Looking specifically at the realism of the map medium, they found that their most detailed map type (with shaded and 3D-like relief) did not elicit faster or more accurate responses from participants than the less detailed map type (using contour lines). This is in line with findings by other authors, both in cartography and geovisualisation, which suggest that naive cartographers or users sometimes prefer media that convey the maximum amount of information, whereas some theorists, based on experimentations, tend to favour a sufficient level of information simplification to allow effective understanding.

In addition, in order not to hinder the readability of the map information, and in particular when creating map mash-ups, Huang et al. (2018) are interested in the synchronisation of the geometry of the base map objects and the overlaid ones. They identify the fact of obtaining this geometric synchronisation as a prerequisite for a proper geovisualisation.

2.3 Existing approaches for Search And Rescue

Prototypes of geovisualisation interfaces responding to various Search and Rescue (SAR) issues that are close to those of CHOUCAS project have been developed such as helping rescuers to locate lost persons or aircraft after an incident. We list them below, taking care to detail their purpose, their functionalities and their development methodology. On the one hand, the knowledge of their functionalities can be useful in the framework of the CHOUCAS project and on the other hand, the development methodology followed will allow us to discuss the choices of these proposals to position our work. We also point out that we are not interested in the organisational and operational aspects of the rescue (such as the assignment of personnel to an incident, the management of resources such as the availability of a helicopter or the monitoring of rescuers once they are on the ground) but in decision support for finding a location,

operated on a geovisual medium.

2.3.1 SAR prototypes in the literature

Work presented by Abi-Zeid et al. (2010) is part of a project whose ultimate goal is to develop a knowledge-based tool for SAR operations to support in determining the possible location of a missing aircraft. As in CHOUCAS, the aim is to determine the most likely location area on the ground, but here for aircrafts that have experienced an incident. The authors have formalised and organised the concepts involved in this task (e.g. terms such as *Hypothesis*, *Distress Case*, *Event*, etc. and *Possibility Area*, which is the area most likely to contain the object being sought). One of the founding elements of their approach is the formalisation of the workflow that they present to users in the form of a flowchart diagram. The authors then propose a prototype interface allowing to instantiate the different elements of the vocabularies and to support the reasoning described by the flowchart diagram. This work informs us mainly on the engineering method used, which seems to be effective and also adapted to the problems raised in the CHOUCAS project which are close to those of the authors.

MapSAR is probably the best-known approach, especially in North America. It is a solution that comes as a plugin to the ESRI ArcGIS desktop software and that is described by Durkee and Glynn-Linaris (2012). In particular, the authors identify the need for rescuers to have many layers of reference data that can be useful to them quickly (and as a prerequisite to handling the alert so as not to waste time searching for them when the time comes). Various elements of this proposal also concern the symbology including the fact that MapSAR is provided with a set of graphic symbols that can be used by rescuers and the fact that this tool offers default symbolisation of some elements related to location clues. This solution is still maintained and is now called SARGIS[7]. We note that the users of this solution, mainly North American, have been gathering since 2015 in the *Search and Rescue Working Group* from the National Alliance for Public Safety GIS (NAPSG) Foundation[8].

The approach proposed by Michal and Robert (2015) is named IGT4SAR and focuses on the modelling of lost persons behaviours from their last known position and their point of destination. The implementation is also made as an ESRI ArcGIS plugin, the IGT4SAR project being based on a fork of MapSAR. The authors provide us with various elements concerning the theoretical basis used for the modelling of lost persons which is carried out: they consider different behaviours models and explain how they can be combined with each other and with the terrain to produce a map of likely zones to be searched. However, their proposal does not focus at all on the visualisation of the results and the analysis of the media by the rescue worker, which takes place in the ArcGIS software interface.

[7] https://www.arcgis.com/apps/Cascade/index.html?appid=627dc87367f14618a9dd8164961415a8
[8] https://www.napsgfoundation.org/about/working-groups/

SHARON[9], captioned "Decision Support System for Search And Rescue of Missing People", is a product developed by Eledia Research Center[10]. While the product presentation page announces several issues similar to ours, it is not possible to get more information. The screenshots and the invitation to participate in the research effort raise questions about whether this is a fully integrated and functional interface or an ongoing project.

2.3.2 CHOUCAS project prototypes

The prototype interface described by Sreeves (2018) was developed for the CHOUCAS project as part of an internship. There are several strengths to this prototype interface (Figure 2.13), which is the first to be produced within the CHOUCAS project:

- the ability to dynamically retrieve OSM data about the part of the territory displayed on the screen,
- the possibility to convert these OSM features into location zones in order to materialise the location elements given by the caller (this is done using a buffer operation on a specific features or on a specific category of features),
- the possibility to create the intersection of these location zones in order to materialise the likely zone of the victim according to a set of clues,
- the possibility to manage several competing hypotheses (an hypothesis being a set a location zones).

Research issues in terms of geovisualisation with regard to the literature as well as additional proposals for this prototype are made in Viry et al. (2019). In this manuscript we notably note that there are limitations related to the import of data. The area in which the searches take place is not materialised on the map, leaving a doubt about the extent in which the objects have been downloaded (the concept of Initial Search Area, which has notably this role, is absent from this prototype and as been added later as explained below and in Chapter 7). Moreover this download is dependent on the overpass API which sometimes fails (depending on the size of the spatial extent requested). We also note that many symbology choices have been implemented in this interface. However, these choices only appear in the application code, which can be detrimental to the reuse of these choices without delving into the prototype application code.

One of our first works, presented in Viry and Villanova-Oliver (2019) and extended in Viry and Villanova-Oliver (2020) describes a second prototype interface produced as part of the CHOUCAS project: GASPAR, which stands for *Geovisual Application for Searching And Rescuing People*. Here our methodology consisted of identifying and formalising, in the form of a decision diagram and in a similar way to Abi-Zeid et al. (2010), the workflow of the rescuer in front of the desired interface

[9] https://www.search-and-rescue-dss.info/?lang=en
[10] http://www.eledia.org/

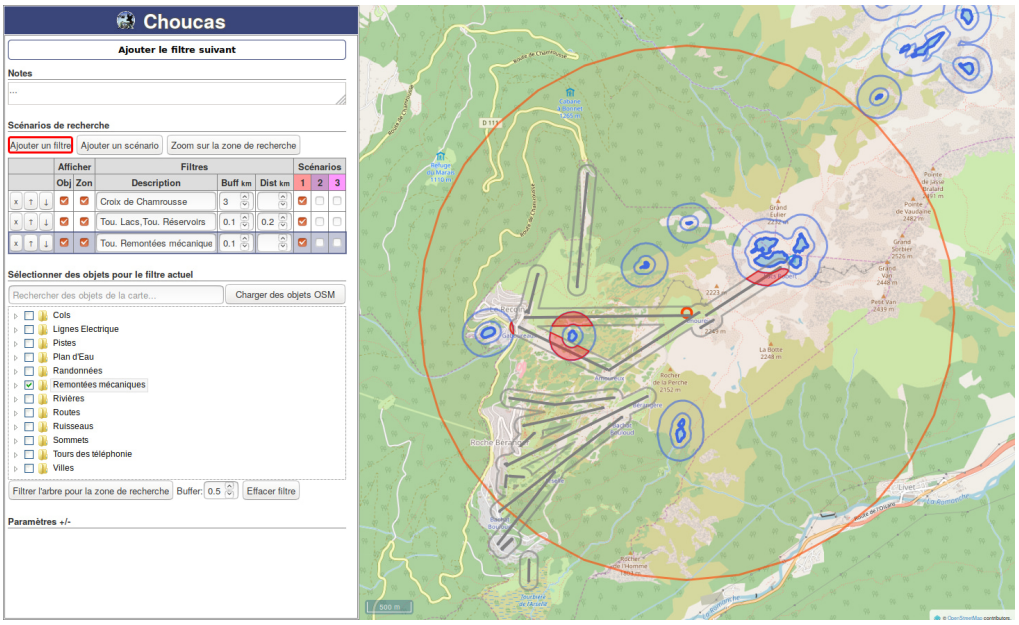

Figure 2.13: Screenshot of interface by Sreeves (2018).

(Figure 2.14). The second point of the method, and on the basis of this workflow, was to organise the various elements of the domain in an ontology, the Choucas Alert Ontology (latter called OAC, which stands for *Ontologie d'Alerte Choucas* in French) that contains concepts such as *Alert*, *Hypothesis*, *Initial Search Area*, *Victim*, etc. This ontology is also of importance within the whole CHOUCAS project as it serves as a shared vocabulary between the members of the project, both during exchanges and to allow interoperability between the works of the different members.

On the basis of these elements, a geovisualisation interface was created (Figure 2.15) This development methodology is flexible and has allowed us to explore various geovisualisation techniques such as joint visualisation on the map and on the globe with a digital elevation model, the use of a notepad to capture the clues given by the victim (Figure 2.16) and the integration of complex geomatic processes such as visibility and shadow zone calculations.

While model transformation methods have been used in order to generate part of the application's data model (in JSON Schema[11] formalism), based on the concepts and properties defined in the ontology, there is no formal link between the concepts of the ontology and the choices of visualisations and interface layout that have been made. This is particularly detrimental, in our view, to the transfer of knowledge about these design choices that could occur later in situations of reuse. To discuss this point in more detail, we will see in the following section how geovisualisation applications are created and how graphical choices are traditionally encoded.

[11] JSON Schema is a vocabulary that allows the annotation and the validation of JSON documents, the specification can be found on its website: https://json-schema.org/.

Figure 2.14: Reasoning model of the user.

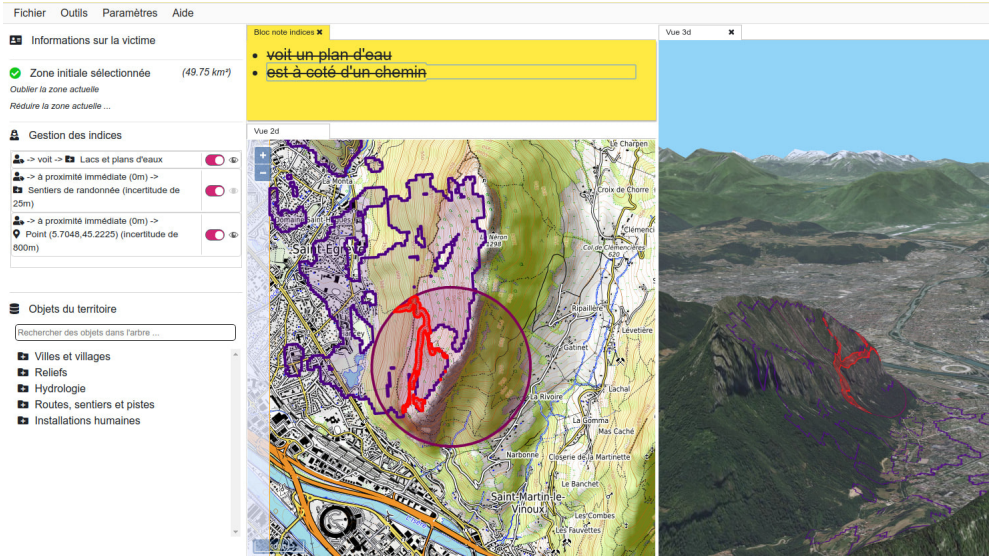

Figure 2.15: Screenshot of GASPAR interface.

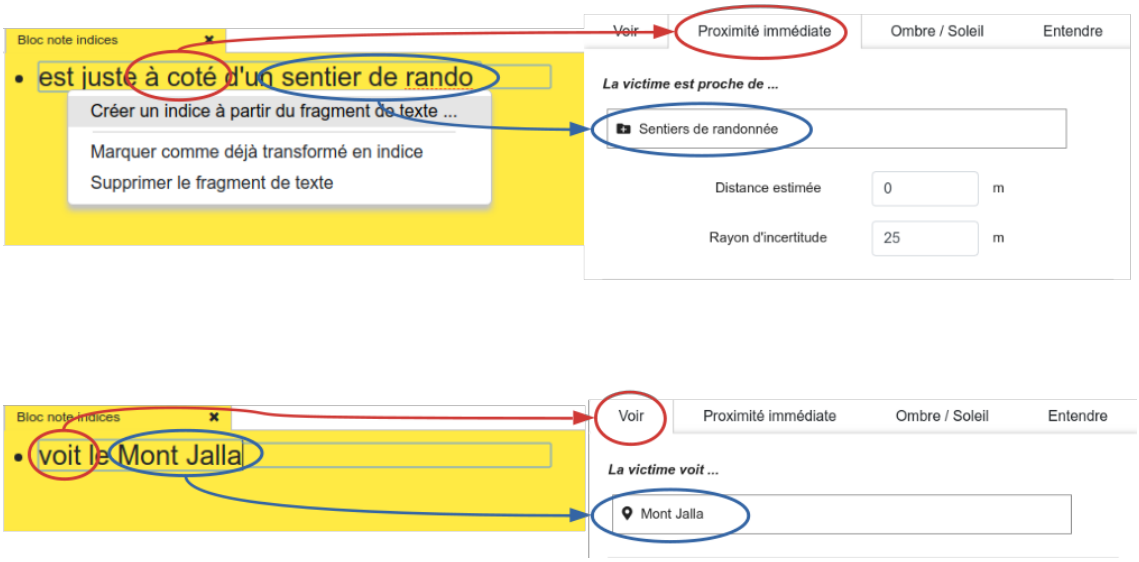

Figure 2.16: Screenshot of GASPAR interface: zoom-in on the notepad used to pre-fill the panel for transforming victim clues into location zones.

2.4 Making a geovisualisation

We have briefly discussed the development methodologies of some tools with similar aims to our issues within CHOUCAS project. Let's take a step back and look at the technical space of interactive map and geovisualisation creation. Here we are particularly interested in their implementation as Web applications. This choice can be explained by the large and dynamic ecosystem of open-source libraries available for the creation of interactive maps, resulting, since the 2010s, from "*a broad transition in client-side web mapping away from standalone, proprietary technologies (e.g., Adobe Flash) and towards open technologies that leverage the HTML, CSS, SVG, and XML web standards (the Open Web Platform) and the JavaScript programming language*" (Roth et al., 2015).

The choice of the Web is also particularly interesting for reasons of interoperability, allowing the developer to design applications that do not depend on any operating system but only require a Web browser. This trend, which is not specific to geovisualisation, has largely accelerated since the 2010s due to performance improvements made by Web browsers and their JavaScript engines[12]. This is part of a process that began in the early days of the Web and which was already identified as being able to improve user acceptance while avoiding the classic pitfalls of software distribution (Rice et al., 1996).

Beyond the choice of Web platform, it should be noted that the design and creation of a geovisualisation interface will in all cases require a combination of computer, cartographic and design skills (Smith, 2016). The design stage of a geovisualisation interface is thus a complex process, generally involving computer developers, domain experts and future users. It should be noted that the audience of future users can be difficult to capture because of its heterogeneous composition (with both general audience and domain specialists for example). These elements may cause tensions, or at least challenges to be addressed, in the design process (Smith, 2016).

In order to better understand the creation of geovisualisation, we discuss the design of geovisualisation from two angles, that of design from a conceptual point of view and that of design from a technical point of view. We then address the issues encountered that lead us to focus our work on the specification of geovisualisation interfaces.

2.4.1 Overall design and challenges

Various issues are associated with the design of geovisualisation interfaces. The main challenge generally concerns the design of an application adapted to the users' needs and it is generally the users' emerging needs that guide the choices that are

[12] JIT compilation was introduced in SpiderMonkey and V8 JavaScript engines as early as 2008 and reflect efforts to improve their performance.

made.

In their work, Robinson et al. (2005) present a user-centred design process for geovisualisation. As this concern is not new, they base their work on various works relating to user-centred approaches, whether in geovisualisation (Slocum et al., 2003) or not (Gabbard et al., 1999). They thus identify a sequence of stages, at each one of which the user is taken into account. These six stages are: *work domain analysis* (the initial communication between the client and the developers), *conceptual development* (the choice of the desired functionalities), *prototyping* (the creation of working models of the future application), *interaction/usability assessment* (the evaluation of the usability of the application as well of its interaction), *implementation* and *debugging*. We observe that in their approach, the conceptual development stage will feed, but also benefit from the feedback of the interaction/usability assessment and implementation stages (Figure 2.17)

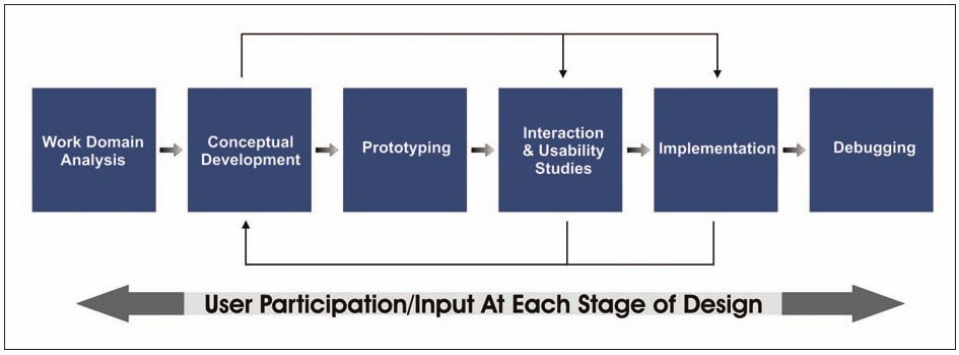

Figure 2.17: User-centered design process. Source: Image from Robinson et al. (2005).

As outlined by these steps, the process of creating the geovisualisation will lead to the choice and implementation of two elements that are the graphical charter and the interactions (which can apply, in both cases, for the geospatial information represented as well as for the interface as a whole). In the following subsection we will see what technical possibilities are available to implement these two crucial aspects of interactive maps.

2.4.2 How does it work ?

Various elements must be chosen or created when building a geovisualisation Web application. Since it is a Web application, these elements are based on the client-server model that is typical of Web applications. These elements can be listed as follows:

- the Web application itself (client-side executable code),
- one or more base maps (provided by the server),

- application-specific data layers (provided by the server)
- application-specific analysis or GIS functionalities (implemented on the client or server side)

Server side

The elements found on the server side mainly relates to the data infrastructure. A server can have several functionalities such as providing base maps, providing business layers or layers to be analysed, and providing GIS functionalities necessary for the operation of the application.

Specific applications, called *map servers*, make it possible to manage these three aspects: publication of tiled base maps (via the WMS protocol), publication of entities or layers (via the WFS protocol) and publication of geomatic processes (via the WPS protocol). In order to be able to find the geospatial data (necessary for the realisation of the base maps or simply to publish these raw data) these map servers enable to connect to various DataBase Management Systems. They thus play an important role in the interoperability of geospatial information. The two main softwares allowing this are MapServer[13] and GeoServer[14] that are both open-source.

In practice, it is quite possible to do without these solutions and it is not uncommon to use base-maps provided directly by the creators (such as those made with OpenStreetMap data and hosted on OpenStreetMap Foundation's servers), to use statically served datasets (in the form of a GeoJSON file, for example) and to implement the application's own geomatic processes on the client side.

Client side

The structure of a geovisualisation Web application is generally based on an HTML document integrating the interactive map and other interface elements. They are commonly designed using the single-page application (SPA) model. In this type of application, the presentation logic is done on the client side, the page is loaded once and does not need to be reloaded during navigation. Such applications are known for their benefits to the user experience in particular by offering a seamless navigation and a feeling close to that of native applications.

Several JavaScript libraries are dedicated to the creation of dynamic and interactive maps. The best known, among those open-source, are Leaflet[15], Openlayers[16],

[13] https://mapserver.org/
[14] http://geoserver.org/
[15] https://leafletjs.com/
[16] https://openlayers.org/

Maplibre[17] (a successor to Mapbox GL JS[18] which is no longer open-source) and Tangram[19]. These libraries have much in common. They are used to invoke, in JavaScript, the creation of a map on the HTML page. They then allow displaying base-maps (e.g. served via the WMS protocol), to add additional layers (whether the data comes from a service such as WFS or from a static file) and to implement and manage all actions related to interactions with the map. Other approaches that do not use a library dedicated to interactive mapping are also sometimes used to draw the maps. The d3[20] library, dedicated to linking data and HTML elements is sometimes used. It allows creating maps in SVG format or using an HTML canvas. PixiJS[21] is also a general-purpose library for drawing on a canvas and can therefore be used to create maps.

As seen in the previous subsection, interactions are essentials for the usability of the geovisualisation created. Although these frameworks support many interactions, the refinement of the proposed solutions will depend on the complexity of the application (interactions having effects on several related views, etc.) as well as the human time required to implement them.

Finally, there are JavaScript libraries which allow the various geomatic operations that are traditionally possible on the server side to be carried out directly in the Web browser. The best known is Turf.js[22]. Complemented by other libraries of the ecosystem (providing, for example, data structures such as R-Trees) it is thus possible to obtain all the functionalities necessary to satisfy the creation of a geovisualisation interface requiring geomatic functionalities for spatial analysis purposes.

We will not go into the details of the implementation of the rest of the components of the geovisualisation application (in particular the display of other resources linked to the map such as graphs and tables), as this can be done in the traditional way in HTML/CSS/JS and using the numerous existing libraries.

2.4.3 Reusable encoding of design choices

Choices that promote interoperability and that carry some semantics can be made when selecting a serialization and exchange format for data. This is particularly the case with the GML format[23], which is a standard from the Open Geospatial Consortium (OGC) and an ISO standard[24] using the XML language. This language is remarkably rich as it allows the encoding of reference coordinate systems as well as complex attribute types. This format is also particularly useful for creating models

[17] https://www.maplibre.org/
[18] https://www.mapbox.com/mapbox-gljs
[19] https://github.com/tangrams/tangram
[20] https://d3js.org/
[21] https://www.pixijs.com/
[22] https://turfjs.org/
[23] https://www.ogc.org/standards/gml
[24] ISO 19136: https://www.iso.org/standard/32554.html

that carry some semantics as it enables the creation of domain-specific application schemas that can latter be used to check data consistency. Thinking back to the SAR applications mentioned in section 2.3, it is possible in a GML application schema to define object types such as *initial search area* or *hiking trail*. These specific object types reference the traditional geospatial primitives on which the format is based. Probably one of the most famous examples of GML application schema is CityGML[25], a data model to store and exchange digital 3D city models.

Solutions that promote interoperability and reusability also exist for the representation of symbology choices. These solutions are based on a logic of separation of content and presentation, as for HTML and CSS for example. This is also the case of the CartoCSS language[26], a language whose syntax is based on CSS but dedicated to describe the style to apply to geospatial layers and their entities. This language offers the possibility to filter entities, for example to implement classifications (e.g. according to a type or a value in order make choropleth maps). However, this specification is not very commonly used, it is not usable with the Web frameworks presented above and is mainly aimed at describing the style of layers to be rendered as server-side tiles.

Two OGC specifications, often used together, are Symbology Encoding[27] (SE) and Styled Layer Descriptor[28] (SLD) (OGC®, 2007). The first describes an XML language for expressing style information for geographic features (vector or raster) and the second describes an XML language for describing how to assign these styles to layers in a WMS server context. These specifications are often referred to together, or sometimes SLD is even mentioned in place of SE, this is for a historical reason: until 2007, the SE specification was part of the SLD specification. SE makes it possible to use the various traditional symbolisation elements (by allowing, for example, the creation of *PointSymbolizer*, *LineSymbolizer*, *PolygonSymbolizer* and *TextSymbolizer*) and to implement rules making it possible, for example, to assign a style to entities meeting a particular condition (using operators such as *PropertyIsGreaterThanOrEqualTo*, for example), or to use transformation functions in order, for example, to map a continuous value to a discrete value, thus enabling to implement complete cartographic portrayals such as choropleth maps. SE and SLD specifications allow the description, in exchangeable and interoperable documents, of the choices made for graphic symbology. However, this specification is not commonly used directly with the Web frameworks presented above and is primarily intended to describe the style of server-side layers: the front-end libraries (such as Leaflet and Openlayers) do not accept SE or any common format for writing the style of the layers to be displayed. Furthermore, defining all the styles of a geovisualisation on the server side (thus not allowing the style to be changed dynamically, according to interactions or to the context) is not desirable and is generally only done for base maps.

[25] https://www.ogc.org/standards/citygml
[26] https://cartocss.readthedocs.io/en/latest/
[27] https://www.ogc.org/standards/symbol
[28] https://www.ogc.org/standards/sld

In this chapter we have reviewed the main concepts of geovisualisation, from its borrowings from cartography to the techniques specific to its interactive nature and to the knowledge support it allows. We have also seen that it is possible to mobilise technologies that favour the transmission of some kind of semantics with the data and that favour the reuse of symbology choices. However, these approaches have limitations in the way they can be used, especially in design workflow oriented on Web client technologies. In order to gain a better understanding of the technologies dedicated to formalising and exchanging knowledge, and because they are focused on the Web, we will look in the next chapter at the possibilities offered by Semantic Web technologies.

Chapter 3

The contribution of Semantic Web technologies for knowledge representation

We saw at the end of the previous chapter that different ways of representing knowledge related to geospatial data exist, in particular for the purpose of creating geovisualisations.

We focus in particular on Semantic Web technologies for several reasons. Indeed, this technological stack makes it possible to represent knowledge in formal ways and allows reasoning based on the Description Logic (DL). It makes it possible to define controlled vocabularies and even more precisely ontologies (representing varying degrees of expressiveness). Individuals belonging to these vocabularies can be created and stored in specific databases (later called *triplestores* or *RDF stores*). This knowledge can be distributed, notably through endpoints that allow query to be written in a dedicated language. Also, semantic rule mechanisms can sometimes be applied in order to produce new knowledge. Lastly, it should be noted that these technologies are widely documented and standardised by the World Wide Web Consortium (W3C).

Semantic Web technologies as defined by the W3C aim to offer a complete stack spanning from data representation (URI/IRI - data encoding is sometimes also represented at the base of this pyramid) to their semantic description to knowledge unification and proof logics enabling their use in applications (Figure 3.1). In this illustration, which presents a hierarchy of languages and standards, each layer uses the possibilities offered by the layer on the lower level. Some of these layers are not yet standardised (namely unifying logic, proof and trust). However, this does not prevent the use of Semantic Web technologies to describe complex knowledge that can be exploited in rich user interfaces using well-known and already standardised tech-

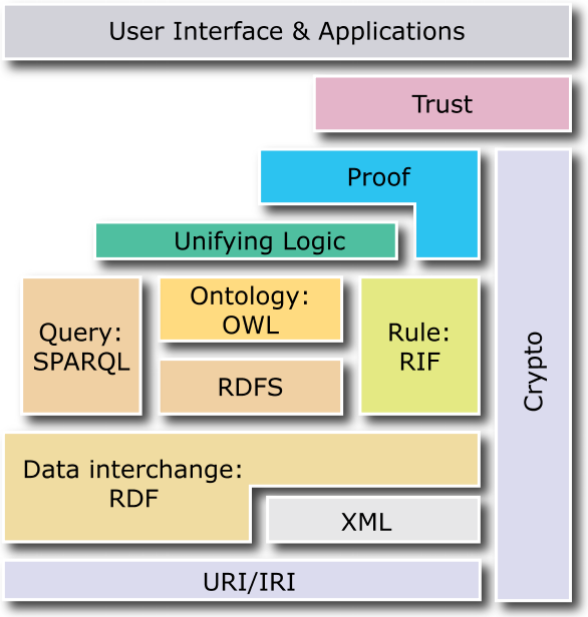

Figure 3.1: Semantic Web layer cake diagram (source: https://www.w3.org/2007/03/layerCake.png, available from https://www.w3.org/2001/sw/, consulted on February 15, 2021).

nologies (such as RDF, RDFS, OWL and SPARQL) as well as ad-hoc propositions coming from Working Groups of the W3C or from other contributors.

The *ontology* is one of the central concepts of this stack and this is also the one on which we focus. This term is borrowed from the discipline of philosophy. In computer science and according to one of the most widely used definitions, this term denotes "an explicit specification of a conceptualization" (Gruber, 1995) where conceptualization is heard as "an abstract, simplified view of the world that we wish to represent for some purpose" (Gruber, 1995). The authors still agree with this definition since recent publications define ontology as "a formal representation of knowledge that forms a shared conceptualisation of a given domain" (Hogan, 2020), emphasising the formal aspect of knowledge representation which will also be of particular interest to us. More precisely, the construction of an ontology makes it possible to identify the set of concepts and the essential properties that link these concepts in order to describe and categorise the various elements that make up a knowledge domain. The notion of "shared conceptualisation" is important to us. As stated in chapter 1, we are part of a research project including several laboratories and an operational operator: some of our proposals in terms of controlled vocabularies and ontologies are intended to be used by all the members of the project, reflecting a shared vision of the defined concepts.

This chapter aims to announce the foundations on which our approach is based. It also provides a comprehensive and up-to-date inventory of Semantic Web tech-

nologies, in order to guide our choices concerning the formal representation then the exploitation of knowledge relating to both the processing of a mountain victim alert (our case study) and geovisualisation.

3.1 Representing and querying knowledge

Work on the Semantic Web is part of an approach aimed at adding a layer of knowledge on top of the traditional web so that web applications can interpret data (as users do) in order to automatically perform some operations and to make interoperability and dialogue between applications possible.

The Resource Description Framework (RDF) standard is the basic building block of the Semantic Web stack proposed by the W3C (Lassila and Swick, 1999). It is a framework that aims to provide the Web with a more suitable data model having a graph structure.

This W3C recommendation from 1999 was updated in 2004 by a set of 6 documents covering the different aspects composing RDF 1.0: primer (Manola and Miller, 2004), concepts (Klyne and Carroll, 2004), syntax (Hayes, 2004), semantics (Beckett, 2004), vocabulary (Brickley and Guha, 2004) and test cases (Grant and Beckett, 2004). Updating work was undertaken and these recommendations were updated in 2014 giving birth to RDF 1.1 (Schreiber and Raimond, 2014; Cyganiak et al., 2014; Hayes and Patel-Schneider, 2014; Gandon and Schreiber, 2014; Brickley and Guha, 2014; Kellogg and Lanthaler, 2014). It is this latter version that we will refer to later when discussing the Resource Description Framework.

We also note that while RDF was initially intended as a data model for metadata of the web, its scope has broadened over time due to its power and versatility. It is now used for conceptual modelling or knowledge management applications for example.

3.1.1 A graph data model: the Resource Description Framework

RDF is a standard defining a graph-based data model with the purpose of exchanging data over the web. This data model is opposed to traditional models such as the relational model or the hierarchical model.

For the purposes of the following sections, we summarise the central and specific elements of this framework: the expression of information in RDF in the form of triples, the elements that can be used in these triples, the constitution of a graph from a set of triples and the serialisations that exist for RDF. These elements will allow us to detail several conventions that will be used later in this document.

Definition of facts as triples

RDF allows making statements about resources using a form known as *triples* which are the "core unit of data in RDF" (Hogan, 2020). The three components of the triple are `<subject> <predicate> <object>`. This simply expresses a relationship, through the predicate, between two resources, defined as the subject and the object. We note that the relationship is directed as it goes from the subject to the object.

Informally defined in pseudo-code, examples of facts that can be expressed are given in Listing 3.1. Later in this chapter, we will see how to actually define these assertions using RDF.

```
<Chaine de Belledone> <is a> <Massif>

<Chaine de Belledone> <is in> <France>

<Chaine de Belledone> <has culminating point> <Grand pic de Belledonne>

<Grand pic de Belledonne> <has height> <2977m>
```

Listing 3.1: Expressing facts about Belledone using triples (in pseudo-code).

As parts of a graph model, "subject" and "object" correspond to the nodes of the graph, while "predicate" corresponds to its arcs. We can therefore draw the graph corresponding to the above statements (Figure 3.2).

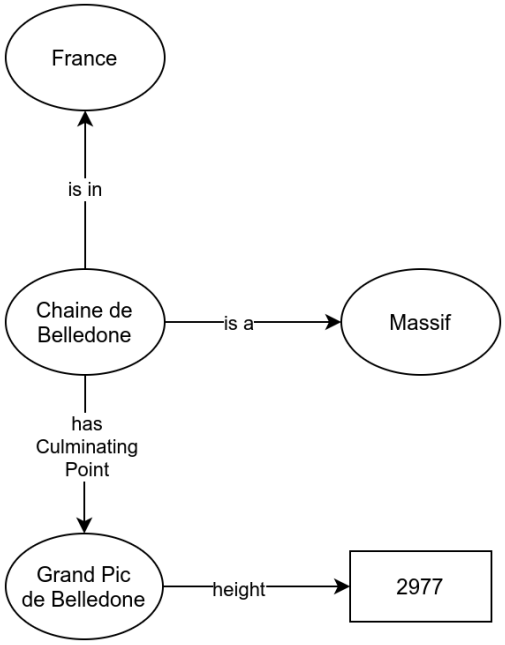

Figure 3.2: Graph resulting from the statements from the above listing.

We will see in what follows how to formally identify these resources.

Various types of RDF terms

Three RDF terms that allow the composition of triples:

- International Resource Identifier (IRI), that can be used to identify a subject, a predicate or an object,
- Literal, that can only be used as object of a statement,
- Blank node, that can be used as subject or as object of a statement.

International Resource Identifier (IRI) refers to the mechanism by which a resource is identified. This concept has been standardised outside the W3C Semantic Web stack by the Internet Engineering Task Force (IETF) in RFC3987[1]. IRIs are a generalization of the Uniform Resource Identifier (URI) concept, which is itself a generalisation of the Uniform Resource Locator (URL) concept[2].

```
1 http://example.com/mountains/Belledone
2
3 http://example.com/ns#Alice
4
5 http://example.com/ns#Bob
6
7 urn:issn:9780721409672
```

Listing 3.2: RDF IRIs.

IRIs can be written as in Listing 3.2. Although they do not reference all resources in the same way, they are all valid. However, it can be seen that some of them share a common part. In order to lighten IRIs writing, it is indeed possible to define and use what is called *prefixes*: they are shorthand corresponding to namespaces (Listing 3.3).

```
1 prefix ex: http://example.com/mountains/
2 prefix ns: http://example.com/ns#
3
4 ex:Belledone
5
6 ns:Alice
7
8 ns:Bob
```

Listing 3.3: RDF IRIs using prefixes.

Although the form taken by the definition of these prefixes changes a little depending on the RDF serialisation chosen (Section 3.1.1), this prefixed notation is commonly used in W3C documents as well as in works referring to the Semantic Web. We will also use this form throughout the text. Below (Table 3.1) is a list of commonly used prefixes. We will use them in the remainder of the text without further reference to the namespaces they designate.

[1] https://www.ietf.org/rfc/rfc3987.txt

[2] Knowing this, it should be noted that RDF 1.0 specification says it uses URIs while RDF 1.1 says it uses IRIs.

Prefix	Namespace
geo	http://www.opengis.net/ont/geosparql#
owl	http://www.w3.org/2002/07/owl#
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs	http://www.w3.org/2000/01/rdf-schema#
sf	http://www.opengis.net/ont/sf#
sh	http://www.w3.org/ns/shacl#
xsd	http://www.w3.org/2001/XMLSchema#
ex	http://example.com/ns#

Table 3.1: Commonly used prefixes in RDF.

The last line of the table (prefix `ex`) designates a namespace on the domain `http://example.com`. This domain is specifically intended to be used illustratively in documents and in our case it illustrates the fact that resources designated by IRIs are not necessarily available through the web, even when a URL is used. We will use this prefix in the examples that follow until the end of this chapter.

Literals allow the formal representation of human-readable information such as labels, comments, descriptions, dates and numeric values (to name only a few examples). These literals are strings that can represent dates, number, boolean and of course strings. They are composed by a lexical form (a unicode string) and optionally by one or two of the following elements : the IRI of a datatype and a language tag. All the terms defined in the following example (Listing 3.4) are valid literals in RDF.

```
"Richard"
"32"^^xsd:integer
"3.14"^^xsd:decimal
"true"^^xsd:boolean
"Chaine de Belledonne"@fr
"Mount Robson"@en
```

Listing 3.4: RDF literals.

A *blank node* enables the creation of resources with no names, i.e. not identified by an IRI but still indicating the existence of a thing. A blank node can also be called an anonymous resource and to put it another way, Hogan (2020) states that "blank nodes are locally-scoped RDF terms that indicate the existence of a resource without identifying it". It may seem strange at first, but we might want to make statements about a person living in France without naming the resource for that person. This is possible thanks to the blank nodes.

This type of construction has several interests from a technical point of view. For example, it will allow complex attributes to be represented when using GeoSPARQL and OWL-Time ontologies that we discuss below. It is also commonly used when using RDF containers (such as `rdf:Bag` or `rdf:Seq`) or when creating specific OWL classes such as Restrictions that we also discuss below.

Depending on the serialisation, blank nodes may be named by the RDF engine performing the serialisation (in case of Turtle serialisation, blank nodes are designated

by a an underscore as prefix, followed by a random identifier, such as `_:b23`), but this name is only valid locally within the RDF file in question.

RDF graph

An RDF graph is a set of triples using the formalisms described so far. This notion generally makes sense because it describes a coherent set of knowledge, on which queries will subsequently be performed.

Using the notions seen so far, we can now redraw the previous graph using the RDF formalism (Figure 3.3). It now uses IRIs (with prefixes), literals and we also added a new fact in order to also use the notion of blank nodes seen just above: we are expressing that `<Le grand Pic de Belledonne> <is seen by> [ <a Person> <living in > <France> ]` without naming the resource that designates this person.

We use several graphical forms here: ovals represent nodes designated by an IRI and empty ovals represent blank nodes, arrows represent properties also designated by an IRI and rectangles designate a literal value. These choices are quite conventional when displaying RDF graphs and we will use them again later in this manuscript.

The use of IRIs and literals to describe the elements of the graph has several advantages. Indeed, and unlike for example the relational paradigm, declarations made in an RDF graph can easily be exchanged ensuring their interoperability across different environments.

The other notion that is usually referred to when talking about RDF graphs is the notion of *RDF Dataset*. This notion covers a set of RDF graphs: a default graph and one or more named graphs, each one designated by a unique identifier.

Syntaxes

We have seen the basic elements for expressing information correctly in RDF. It is now time to look at how this information can be serialised. One of the six deliverables of RDF 1.x explicitly addresses the RDF/XML syntax which is the main normative syntax for RDF. However this syntax, based on XML, is particularly verbose. Other syntaxes specifically for RDF have been standardised.

Turtle, which stands for Terse RDF Triple Language, is a syntax of a language that has been specially designed for non-XML serialisation of RDF data. The Turtle specification is a W3C recommendation since RDF 1.1 (Beckett et al., 2014). Other standard syntaxes such as N-triples (firstly defined by Grant and Beckett (2004) in RDF 1.0 test-cases document) or JSON-LD (defined in Sporny et al. (2014) and updated in Sporny et al. (2020)) exist but they will not be discussed here. From now on, the RDF code listings that will be provided in this document will use the

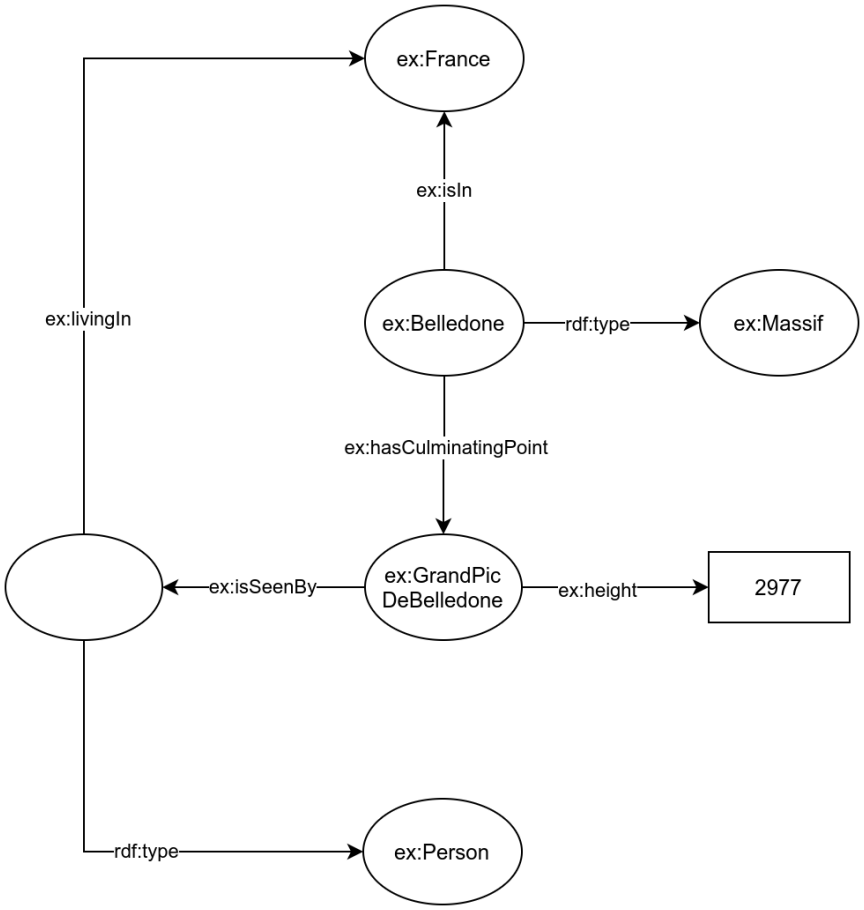

Figure 3.3: RDF graph resulting from the statements from the above listing.

Turtle syntax, as in Listing 3.5 which reuse the statements of the previous listing to express them in correct RDF 1.1, serialized in Turtle. Note that, in Turtle, using `a` is semantically equivalent to using the property `rdf:type`.

```
ex:Belledonne a ex:Massif .

ex:Belledonne ex:isIn ex:France .

ex:Belledonne ex:hasCulminatingPoint ex:GrandPicDeBelledonne .

ex:GrandPicDeBelledonne ex:hasHeight "2977"^^xsd:integer .

ex:GrandPicDeBelledonne ex:isSeenBy [
    a ex:Person ;
    ex:locatedIn ex:France ;
  ] .
```

Listing 3.5: Turtle example.

Other elements in RDF

The RDF specification also describes how to describe containers (ordered or unordered) collections.

RDF containers are used to represent a set of IRIs or literal values and are of several kinds: `rdf:Bag`, `rdf:Seq` and `rdf:Alt` which describe respectively an unordered container, an ordered container and a collection of alternatives.

Vocabulary elements are also provided to describe RDF collections as lists with `rdf:List`. We will not go into the details of the construction of these lists but simply note that they use a Lisp-like structure and thus require the introduction of the following vocabulary elements: `rdf:first`, `rdf:rest` and `rdf:nil`.

The other vocabulary elements provided as part of RDF 1.1 recommendation belong to a separate namespace: RDF Schema (RDFS) whose purpose is to provide lightweight constructs for defining ontologies. These elements are presented in what follows, alongside other powerful proposals for building even more expressive ontologies.

3.1.2 Defining vocabularies and ontologies: RDFS and OWL in action

RDFS

RDFS provides a schema and some basic ontological constructs: it will thus be possible to express the semantic characteristics of RDF data. This relies on the

possibility of defining "class" (`rdfs:Class`) and "property" (`rdfs:Property`), of creating classes and properties hierarchies and of specifying the "domain" (what can be used as a subject of a property) and the "range" (what can be used as an object of a property) of properties. We note that as such, class hierarchies support the inheritance of "domain" and "range" properties. RDFS also provides other important elements such as the `rdfs:Datatype` and `rdfs:Literal` classes which are used to describe datatypes and literals respectively, and the `rdfs:label` and `rdfs:comment` properties which allow to provide human-readable versions of the names and description of a resource. Note that, by convention, the first letter of class names is capitalized, while the first one of property names is not.

These constructs are associated with a system of logical consequences, also called entailments, which can be constructed in the graph with respect to the inferences that can be made about the data from the semantics of the model. Some inferences that can be drawn from RDFS entailment will be presented briefly below, alongside inferences specific to OWL in a dedicated section (3.1.4)

OWL

Web Ontology Language (OWL) is a knowledge representation language that allows the definition of ontologies with a high expressiveness by using constructs from Description Logic (DL).

This language was inspired by previous work on ontology languages: DAML, OIL, DAML+OIL, etc. The first version of OWL became a W3C recommendation in 2004 (Dean and Schreiber, 2004). The OWL 2 recommendation dates from 2009 and was revised in 2012 (Hitzler et al., 2012). When we refer to OWL in the remainder of this document, reference is made to the latest version, OWL 2, in its 2012 revision. Also we note that OWL 2 has been designed so that OWL 1 documents are also valid in OWL 2.

We note that the authors of the latest revision of OWL describe an ontology as being "a set of precise descriptive statements about some part of the world" (Hitzler et al., 2012). We emphasise the term *descriptive* because even though OWL could also be seen as a prescriptive solution, it allows to specify how to create new individuals and not only to describe existing ones, the authors insist on this point which could be misunderstood: "OWL 2 is not a schema language for syntax conformance. Unlike XML, OWL 2 does not provide elaborate means to prescribe how a document should be structured syntactically. In particular, there is no way to enforce that a certain piece of information (like the social security number of a person) has to be syntactically present. This should be kept in mind as OWL has some features that a user might misinterpret this way." (Hitzler et al., 2012).

The OWL 2 recommendation comes in 2 different flavours depending on the formal model-theoretic semantics used: *Direct Semantics* (Motik et al., 2012) which is

based on the semantics of DL $\mathcal{SROIQ}^{(\mathcal{D})}$ and *RDF-Based Semantics* (Schneider, 2012) which intend to apply to RDF graphs and is directly built on the top of RDF Semantics document (Hayes, 2004). In the latter one, this recommendation is compatible with RDFS and can be used as a complement to enhance the expressiveness of ontologies that can be described in RDF.

The vocabulary elements provided by OWL are indeed numerous. They allow the specification of classes (`owl:Class`) and properties of various types (such as `owl:ObjectProperty`, `owl:DatatypeProperty`, `owl:FunctionalProperty`, `owl:InverseFunctionalProperty` and `owl:SymmetricProperty`). Numerous ways are also offered to qualify these classes (e.g with `owl:disjointWith` and `owl:equivalentClass`) and properties (e.g. with `owl:propertyDisjointWith`, `owl:equivalentProperty` and `owl:inverseOf`) as well as to express equivalence or disjunction between individuals (`owl:sameAs` and `owl:differentFrom`). These elements can be of particular use in merging facts expressed in different models. Various other elements are provided, for example to specify anonymous classes resulting from the union or intersection of other classes, or restrictions (`owl:Restriction`) resulting from a constraint on a property.

By taking the examples seen so far and combining the modelling capabilities of RDFS and OWL to enrich these examples, we can define the classes that correspond to the elements to be instantiated and the properties that will be used to link them or to qualify them with literals (Listing 3.6).

```
ex:Massif a owl:Class .
ex:Peak a owl:Class .
ex:Country a owl:Class .

ex:isIn a owl:ObjectProperty ;
    rdfs:domain ex:Massif ;
    rdfs:range ex:Country .

ex:hasCulminatingPoint a owl:ObjectProperty ;
    rdfs:domain ex:Massif ;
    rdfs:range ex:Peak .

ex:hasHeight a owl:DatatypeProperty ;
    rdfs:domain ex:Peak ;
    rdfs:range xsd:integer .
```

Listing 3.6: Definition of a model using RDFS/OWL whose instantiation will make it possible to express the facts mentioned previously.

As an example, the graphical depiction of the ontology formed by the previous listing (Listing 3.6) can be done using a graphic notation (Figure 3.4). This figure is based on VisualOWL (VOWL), a visual language for the representation of ontologies proposed by Lohmann et al. (2016) and whose latest version of the specification can be found online[3]. The open-source tool WebVOWL[4] allows the creation of these graphs (both on the creator's website[5] and by installing it locally) from a VOWL-JSON

[3] http://purl.org/vowl/spec/
[4] http://vowl.visualdataweb.org/webvowl.html
[5] Until October 1, 2021.

format that can be obtained from an OWL file.

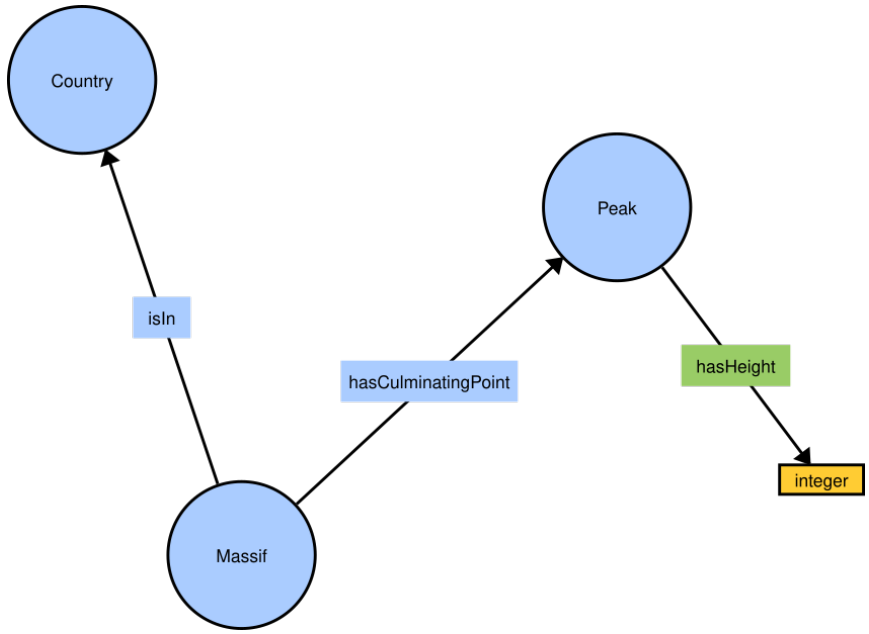

Figure 3.4: Graphical depiction of an ontology using the VisualOWL notation (made using WebVOWL).

We will use this formalism when we will be showing our OWL ontologies. However, this specification is not intended to display RDF graphs but OWL models. As such, individuals are not represented individually but only as a count at the location of the class to which they belong for example.

Choosing a level of expressiveness

The OWL 2 recommendation gives rise to two levels of expressiveness depending on the formal semantics used. In the case of direct semantics, ontologies are referred to as OWL 2 DL and in the case of RDF-based semantics they are referred to as OWL 2 Full. The expressiveness offered by OWL 2 Full makes the language undecidable but makes any RDF document compliant with OWL 2 Full.

The OWL 2 recommendation also defines usage profiles (Hitzler et al., 2012). Each of these profiles corresponds to a subset of constructs that can be used within that profile and allow either to implement more easily the entailment regime or to reason more efficiently on ontologies and data in OWL 2 that are limited to the use of this subset of constructs. These profiles do not represent a subset of each other and are threefold:

- OWL 2 EL, which is suitable for large class-oriented ontologies,

- OWL 2 QL, which corresponds to a subset of the language that can be realized using standard database technologies (such as SQL),
- OWL 2 RL, which corresponds to a subset of OWL 2 that can easily be implemented with a rule-based mechanism, making it usable for scalable reasoning.

Without going into the details of the constructs chosen for each of the profiles, it should be noted that at one extreme, OWL 2 Full allows the most expressive use of ontology design because its semantics encompasses RDF and RDFS vocabularies, but is undecidable. By contrast, the OWL 2 RL profile is supported by more tools (see 3.3) because it is easier to implement as a set of rules. These elements should be kept in mind when designing an ontology in OWL, depending on the type of usage that is intended. In addition, we note that there are tools, such as the OWL API profile checker[6] to check the conformity of an ontology to these different profiles.

Here we also present *RDFS-Plus* that can be seen as an extension of RDFS and as a subset of OWL. As defined by the authors who proposed this concept, it is "RDFS with a bit of OWL" (Allemang and Hendler, 2011). This is a specific set of RDFS and OWL constructs, sometimes presented as a "non-standard profile", that has been chosen based on an informal survey of practitioners regarding *pedagogical*, *practical* and *computational* criteria.

This concept benefits from recognition both in the industrial world, by being implemented as an inference regime in AnzoGraph and GraphDB products which will be presented briefly below, and in the academic world, notably in the context of research relating to the implementation of efficient reasoners for RDF, as in the proposal described by Subercaze et al. (2016).

Finally, this concept corresponds to an undeniable reality, with many authors facing the need to describe the expressiveness of their vocabulary as "RDFS plus some OWL constructs". Without referring to RDFS-Plus, the LOV platform advises to produce vocabularies with "Low formal constraints (basically RDFS and a fistful of OWL)"[7]. Furthermore, Glimm et al. (2012) jointly analyse subsets of RDFS/OWL, including RDFS-Plus, through the prism of the most used classes and properties on the web as well as through the prism of their support in various tools. This work is also of great help both in making the right choices when designing ontologies and in choosing the right tools.

In this document we will refer to *OWL 2 profiles* and *RDFS-Plus* concepts in order to contextualise the level of expressiveness of the proposed ontologies and the inference regime used on the data processed.

[6] https://github.com/stain/profilechecker
[7] https://lov.linkeddata.es/Recommendations_Vocabulary_Design.pdf

Practical examples of ontologies for spatio temporal applications

Several vocabularies and ontologies exist to define geospatial concepts. They aim to describe different elements such as a position, the various existing geometric primitives, the reference coordinate systems, the existing topological relationships between entities, etc. However, these vocabularies and ontologies do not describe all these aspects at once. Nor do we necessarily need to use all these aspects to describe data (in some application needs, the data is always *Point* data, or it is always in *WGS84*, etc.). These elements, as well as the natural evolution of vocabularies over iterations and possible divergences, explain the existence of a variety of practices to describe geospatial data and metadata.

In line with the elements presented so far, it seems normal to begin by mentioning the W3C Geo Vocabulary proposed in 2003[8]. This vocabulary[9] is described as "basic" and primarily enables the description of points with latitude, longitude and altitude components, expressed in WGS84 reference system. In parallel, work to include geographic coordinates in RSS feeds has resulted in the GeoRSS standard. Work has continued within the W3C Geospatial Incubator Group (GeoXG), resulting in the publication of a report (Lieberman et al., 2007) proposing an update to the W3C Geo Vocabulary, strongly inspired by the inclusion of GeoRSS concepts, as an ontology named Geo OWL[10].

GeoSPARQL 1.0 (OGC®, 2012) is an OGC standard which notably defines a RDFS/OWL ontology[11] for geographic information. It is probably the most widely used geospatial technology for the Semantic Web due to the richness of the proposition. Indeed, the standard does not only offer a vocabulary but also spatial functions allowing the computation of topological predicates between two entities and allowing to perform non-topological spatial operations (buffer, intersection, convexhull, etc.) which can be used in the SPARQL query language, presented in the next subsection. The Simple Features Geometry ontology[12] is also provided in the GeoSPARQL standard in order to represent a class hierarchy of geometric primitive types (such as Point, Polygon, etc.). Discussions and work on updating the standard are taking place in the GeoSPARQL Standards Working Group[13] of the OGC and should eventually lead to GeoSPARQL 1.1.

Other geospatial vocabularies exist and for example, the efforts of the NeoGeo community have resulted in the publication of a vocabulary allowing to describe spatial relations and geometries in 2009 which was updated in a new version in 2011, maintained on the GeoVocab website[14]. This vocabulary is designed as an extension

[8] https://www.w3.org/2003/01/geo/
[9] http://www.w3.org/2003/01/geo/wgs84_pos
[10] https://www.w3.org/2005/Incubator/geo/XGR-geo-20071023/W3C_XGR_Geo_files/geo_2007.owl
[11] http://schemas.opengis.net/geosparql/1.0/geosparql_vocab_all.rdf
[12] http://schemas.opengis.net/sf/1.0/simple_features_geometries.rdf
[13] https://github.com/opengeospatial/ogc-geosparql
[14] http://geovocab.org/doc/neogeo.html

to the W3C Geo Vocabulary in which it is possible to represent different types of geometry as RDF resources which includes all the nodes that make up a line or a polygon for example. This is a major difference from GeoSPARQL, in which the geometry of entities is given in the form of a literal (in WKT or GML for example).

In addition, ontologies concerning various aspects useful for the formalisation of geographic information are proposed by the IGN: an ontology of reference coordinate systems[15] and an ontology of geometric primitives[16]. These ontologies can also be useful for expressing metadata about geographical information.

Finally, if GeoSPARQL encodes usual topological relationships (using 3 families: simple features, Egenhofer and RCC8) between two entities, as traditionally expressed in geomatics, it does not allow the description of spatial relationships as they could be formulated by humans, in natural language. Vocabularies for encoding such spatial relationships are of particular interest in the CHOUCAS project. This aspect has been notably addressed by Bateman et al. (2010) and has been the subject of detailed work in the CHOUCAS project by Bunel (2020) which proposes an ontology of spatial relations, associated with methods allowing the calculation of corresponding fuzzy zones in raster format.

In addition to the spatial domain, temporal elements can be described in OWL. OWL-Time is a W3C recommendation (Cox and Little, 2017) describing a OWL ontology of temporal concepts and temporal properties. It makes it possible to express ordering information between instants and durations in a way that is compatible with many use cases. It can thus be used for example to express and solve scheduling decisions or even to represent and query geological timescales[17]. We will not go into further details about the possibilities of this ontology, but it should be noted that time positions and duration can be easily described using various other temporal reference system than the conventional calendar and clock (such as Unix-time).

The combination of GeoSPARQL and OWL-Time vocabularies alongside with the modelling capabilities of RDFS and OWL will allow us to express more complex elements than previously. In order to extend the examples presented in the previous listings, let us say that we now want i) to make explicit that our `ex:Peak` concept is also a geospatial entity, and ii) to define a concept named "location element" to describe a spatial relationship between two elements and the time or duration at which this spatial relationship is valid. For example, we want to encode the fact that a location element exists such as "John is near the Grand Pic de Belledonne on 12/04/2020 at 13:20".

```
# The new class we want to instantiate in this example
ex:LocationElement a owl:Class .

# We also redefine the 'Peak' class
ex:Peak a owl:Class ;
    rdfs:subClassOf geo:Feature .
```

[15] http://data.ign.fr/def/ignf/20160628.en.htm
[16] http://data.ign.fr/def/geometrie/20190212.en.htm
[17] http://resource.geosciml.org/vocabulary/timescale/gts2018

```
7
8  # Definition of some other classes
9  ex:LocationRelation a owl:Class .
10
11 ex:Person a owl:Class .
12
13 # Definition of individuals belonging to these classes
14 ex:John a ex:Person .
15
16 ex:isNearby a ex:LocationRelation .
17
18 # Definition of a Peak, which is also a GeoSPARQL Feature
19 ex:GrandPicDeBelledonne a ex:Peak ;
20     rdfs:label "Grand Pic de Belledonne"@fr ;
21     geo:hasGeometry [
22         a geo:Geometry ;
23         geo:asWKT "POINT(5.9912892 45.1709618)"^^geo:wktLiteral ;
24     ] .
25
26 # Defining an indivual belonging to LocationElement
27 ex:loc a ex:LocationElement ;
28     ex:hasRelation ex:isNearby ;
29     ex:hasSite ex:John ;
30     ex:hasTarget ex:GrandPicDeBelledonne ;
31     ex:hasInstantOrDuration [
32         a time:Instant ;
33         time:inXSDDateTimeStamp "2020-04-12T13:20:00Z"^^xsd:dateTimeStamp
34     ] .
```

Listing 3.7: Expressing spatio-temporal facts with GeoSPARQL and OWL-Time.

The choices made here have several advantages. For example, it can be seen that at no point have we asserted the fact that "John is nearby Grand Pic de Belledonne", we have only asserted the existence of an individual of type `ex:LocationElement`, mobilising the elements "John", "being nearby" and "Grand Pic de Belledonne" and expressed the time at which this information is valid. This could for example correspond to information collected by a mountain rescuer, which shows how such a formalism could be used to describe information of interest to us in the CHOUCAS project. This type of construction is also somewhat comparable to the constructions possible with the RDF reification vocabulary that allows the description of a RDF statement without asserting it in the graph.

On a more technical note, we can see that the use of blank nodes is the idiomatic way of expressing various kinds of information with both GeoSPARQL and OWL-Time: it is relevant not to name the resource corresponding to the geometry of an entity neither that the resource corresponding to an instant (while allowing them to be shared between statements when the knowledge is merged into a graph).

3.1.3 SPARQL, a query language dedicated to RDF

Querying is heard as technologies and protocols that can programmatically retrieve information from a knowledge base (as defined on W3C webpage for example[18]). In other database and storage paradigms, this refers to the SQL language for querying relational data, or to the XQuery language for querying hierarchical data.

SPARQL (SPARQL Protocol And RDF Query Language) is a language and a protocol dedicated to RDF that allows querying, updating, enriching or deleting data stored in RDF format. Like most of the bricks presented so far, this language is a W3C recommendation, published in 2008 for SPARQL 1.0 (Prud'hommeaux and Seaborne, 2008) and updated in 2013 for SPARQL 1.1 (W3C SPARQL WG, 2013).

To enable these query capabilities, SPARQL relies directly on the triple form that is constitutive of RDF and gives the possibility of querying graph patterns. The main features of SPARQL are notably the support for aggregation, negation, subqueries and creating values by expression. All of its features and syntax can be found in Harris and Seaborne (2013).

Back to the data we saw in the previous code listing (Listing 3.7), we might envisage to interrogate this knowledge in order to know, for example, which people in the context of a `ex:LocationElement` are near some object. This can be expressed by the following SPARQL query (Listing 3.8). Note that the syntax used is close to turtle's syntax (use of prefixed IRIs, writing of literals, etc.) and that common elements with SQL are present (keyword `SELECT ... WHERE` for a selection operation for example).

```
PREFIX ex: <http://example.com/ns#>

SELECT ?person ?target WHERE {
    ?locElem a ex:LocationElement ;
        ex:hasRelation ex:isNearby ;
        ex:hasSite ?person ;
        ex:hasTarget ?target .
}
```

Listing 3.8: Example SPARQL query

Only one row, containing 2 columns, is returned by this query (Table 3.2). They contain the IRI's of the resources that correspond to the query expressed.

?person	?target
http://example.com/ns#John	http://example.com/ns#GrandPicDeBelledonne

Table 3.2: Row returned by the query from Listing 3.8.

The power of SPARQL also lies in its ability to perform federated queries that use the knowledge from another SPARQL endpoint (Prud'hommeaux and Buil-Aranda,

[18] https://www.w3.org/standards/semanticweb/query

2013). This is done by using the SERVICE clause that informs the query processor to execute a portion of the query against a particular SPARQL endpoint. The results are combined with the results of the rest of the query and then returned. This type of mechanism is a means of consuming linked data without the need to copy it first.

It is thus possible to extend the previous query by connecting to services such as the SPARQL endpoint of Wikidata to retrieve information about all the things that have coordinates and are registered in Wikidata near the Grand Pic de Belledonne (such as in Listing 3.9 and its corresponding result, Table 3.3) or to do a text search to find a Wikidata entity that would also be named "Grand Pic de Belledonne" and get more information about it.

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX ex: <http://example.com/ns#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wikibase: <http://wikiba.se/ontology#>
PREFIX bd: <http://www.bigdata.com/rdf#>

SELECT ?placeLabel ?typeLabel ?place WHERE {
    ?locElem a ex:LocationElement ;
             ex:hasRelation ex:isNearby ;
             ex:hasSite ?person ;
             ex:hasTarget ?target .

    ?target geo:hasGeometry [ geo:asWKT ?geom ] .

    SERVICE <https://query.wikidata.org/sparql> {
        SERVICE wikibase:around {
            # Places with coordinates
            ?place wdt:P625 ?location .
            # That are in a given circle aroung a point
            bd:serviceParam wikibase:center ?geom  .
            # Where the circle has a radius of 5km
            bd:serviceParam wikibase:radius "5"^^xsd:integer .
            bd:serviceParam wikibase:distance ?distance .
        }

        ?place wdt:P31 ?type .
        ?place rdfs:label ?placeLabel .
        ?type rdfs:label ?typeLabel .
        FILTER(lang(?placeLabel) = "fr") .
        FILTER(lang(?typeLabel) = "fr") .
    }
}
```

Listing 3.9: Example of nested federated SPARQL query

Without going into the details of the possible integration of the knowledge obtained in this way, we can see how such possibilities can be interesting to enrich information obtained during a victim search in the CHOUCAS project.

Obtaining a set of results in the form of a table, as shown here with the SELECT

?placeLabel	?typeLabel	?place
Eau d'Olle	rivière	http://www.wikidata.org/entity/Q3046680
Croix de Belledonne	montagne	http://www.wikidata.org/entity/Q3003298
Grand Charnier d'Allemond	montagne	http://www.wikidata.org/entity/Q3113554
Grand Pic de Belledonne	montagne	http://www.wikidata.org/entity/Q3113702
Grande Lauzière	montagne	http://www.wikidata.org/entity/Q3114954
Grande Lance de Domène	montagne	http://www.wikidata.org/entity/Q3114955
Pic du Grand Doménon	montagne	http://www.wikidata.org/entity/Q3382427
Mont Saint-Mury	montagne	http://www.wikidata.org/entity/Q31630964
La Grande Lance	montagne	http://www.wikidata.org/entity/Q31641152
La Croix de Belledone	montagne	http://www.wikidata.org/entity/Q31662824
lac de Belledonne	lac	http://www.wikidata.org/entity/Q3215362
lac du Crozet	lac	http://www.wikidata.org/entity/Q62648470
glacier de Freydane	glacier	http://www.wikidata.org/entity/Q3108334
Le Grand Replomb	sommet	http://www.wikidata.org/entity/Q31639917
rocher de l'Homme	sommet	http://www.wikidata.org/entity/Q31641809
sommet Colomb	sommet	http://www.wikidata.org/entity/Q31647104
sommet de Belledonne	sommet	http://www.wikidata.org/entity/Q31651287
pic Belledonne	sommet	http://www.wikidata.org/entity/Q31651303
chaîne de Belledonne	chaîne de montagnes	http://www.wikidata.org/entity/Q792472
lacs du Domènon	ensemble de lacs	http://www.wikidata.org/entity/Q3215892

Table 3.3: Rows returned by the query from Listing 3.9.

clause, is not the only feature of SPARQL. Indeed, it is also possible to use the `ASK` clause to obtain a result in the form of "true" or "false" and to use the `CONSTRUCT` clause to obtain a result in the form of an RDF graph. In addition, SPARQL also enables the removal, the insertion or the update of values in a graph. This is possible with the use of `DELETE` and `INSERT` clauses, sometimes combined within the same query. The use of the `CONSTRUCT` clause will be of particular interest to us. Indeed, this kind of query allows to return an RDF graph from a graph template expressed in the query (that can contain, for example, new individuals created on the fly according to the matches found in the queried graph).

3.1.4 Drawing inferences

Ontology languages such as RDFS and OWL define constructs (properties) whose underlying semantics can be used for reasoning. For example and among others, RDFS defines the semantics of `rdfs:domain` and `rdfs:range` and OWL defines the semantics of `owl:ObjetProperty` and `owl:FunctionalProperty`. The application of the inference regime specific to RDFS or OWL may have consequences on the knowledge expressed. For example, considering the RDFS/OWL model defined earlier (Listing 3.6) and expressing the fact that `:Abc ex:isIn :Xyz` then, by the semantics of RDFS, a reasoner will deduce that `:Abc a ex:Massif` and that `:Xyz a ex:Country`. Similarly, if we express that `:SomeThing ex:isIn "some literal"` an OWL reasoner must be able to tell that the ontology is not consistent (i.e. there is no model that captures this knowledge).

Various reasoning tasks exist and can generally be implemented by reasoners such as realisation (or materialisation as the computation of all implied class assertions for named classes and individual names appearing in the ontology), classification (which

is the computation of all implied class subsumptions between named classes in the ontology), consistency checking (whether the ontology is consistent) and entailment checking. Classification and realisation tasks will lead to the creation of new knowledge, in a similar way to the example given at the beginning of this sub-section. The knowledge present before the reasoning stage is said to be *asserted*, whereas the knowledge created at this stage is said to be *inferred*.

However, the new knowledge that can be inferred from this reasoning is limited to the semantics of the various constructs available in RDFS and OWL ; they are said to be monotonic languages which means, for example, that they do not support default reasoning where some truth is true unless it is contradicted by some other information. It is therefore not possible with them to define IF-THEN-ELSE rules neither rules enabling the calculation of a value based on the characteristics of an individual (a typical example would be calculating the area of a rectangle using its `height` and `width` properties)

After discussing how data in RDF can be validated, we will discuss solutions that exist for describing more complex rules and for generating new knowledge or modifying existing knowledge.

3.1.5 Graph shape validation

In the example seen earlier (Listing 3.6), we have defined classes and properties using RDFS and OWL, but there is no guarantee that the data we instantiate from these classes respects the model in question. Indeed, as stated before, this is not the purpose of languages like RDFS and OWL. Nevertheless, it may seem natural to check if the instances of `ex:Peak` and `ex:Massif` present in our graph conform to a precise form. Proposals in the W3C stack specifically validate the correct shape of RDF constructs in a graph.

Shape Constraint Language (SHACL) is a W3C recommendation (Knublauch and Kontokostas, 2017) resulting from the efforts of the RDF Data Shapes Working Group[19]. The recommendation describes SHACL Core, a RDF vocabulary allowing the construction of constraints to validate RDF data and SHACL-SPARQL, an extension mechanism for describing additional constraint components using SPARQL.

Two central concepts of SHACL are node shapes and property shapes. These are respectively a mechanism for declaring constraints directly on a node and a mechanism for declaring constraints on a node through the values associated with it through a path. Finally, the shape graph is the graph that contains all the shape definitions to be used. This is the graph that will be given as input to the SHACL validation engine, alongside the RDF graph to be validated. The SHACL validation engine returns a validation report. In the case of a conformance, a simple validation report (containing `sh:conforms true`) is returned. In the opposite case, a validation report

[19] https://www.w3.org/2014/data-shapes/wiki/Main_Page

listing the errors and metadata about their cause is returned. Note that the individuals conforming to shapes defined for validation might be used in SHACL - Advanced Features as targets for the application of rules (see 3.2.2).

In the continuity of Listing 3.6 we might want to express that an individual of type `ex:Massif` must be linked to 1 or more individuals of type `ex:Country` by the `ex:isIn` property. This can be done using a SHACL shape (Listing 3.10) and when this mechanism is applied to the RDF graph, a validation error is reported for each individual that does not satisfy the expressed constraint.

```
ex:MassifInOneOrMoreCountry
    a sh:NodeShape ;
    sh:targetClass ex:Massif ;
    sh:property [
        sh:path ex:isIn ;
        sh:class ex:Country
        sh:minCount "1"^^xsd:integer ;
        sh:message "Each individual of type ex:Massif is expected to be linked to
    one or more ex:Country using the ex:isIn property."@en ;
    ] .
```

Listing 3.10: Example of SHACL shape

The interest in the expressive power of SHACL and the various uses that can be made of it can be found in numerous themes, ranging for example from the modelling of construction scheduling constraints (Soman, 2019) to the Publication Office of the European Union[20] that sees the advantage of SHACL to implement its application profiles in order to both perform data validation and generate the appropriate documentation. The Publication Office of the European Union, like national-level agencies for example, is known to be a heavy user of Semantic Web technologies for metadata management, and as such, their feedback is particularly valuable. In this regard, it is worth noting that they point out "the problem of segregation between documentation and implementation" and the fact that they wish to "precisely describe, for both humans and machines, what the data ought to be"[21]. These points echo both our desire to formalise in a reusable way the elements that make up a mountain victim search alert and the declarative approach to geovisualisation that we mentioned at the beginning of this manuscript and that we will try to tackle in the remainder of it.

Another proposal similar to SHACL, sharing many purposes, is Shape Expressions Language (often called ShEx). This W3C Member submission, published in 2014 in its 1.0 version (Solbrig and Prud'hommeaux, 2014), includes a primer and a semantics specification. This proposal, which also benefits from an important interest in the Semantic Web community, continues to evolve and to be updated, reaching today its version 2.1 (Prud'hommeaux et al., 2019). ShEx is used to describe and validate graph structures. It has several serializations including a compact syntax (ShExC) that is often used for its ease of reading for humans (being close to Turtle

[20] https://www.w3.org/2016/11/sdsvoc/SDSVoc16_paper_23
[21] https://www.w3.org/2016/11/sdsvoc/willem

and SPARQL). Similarly to SHACL, ShEx (under its RDF representation) can be described and validated in ShEx.

3.2 Drawing more complex inferences and setting up a system of rules

We focused on proposals describing a language of semantic rules that can be used with information represented in RDF. Ideally, these solutions should allow new knowledge to be merged into the initial graph, so that it can be interrogated using query mechanisms such as SPARQL. We are also looking for a solution in which the rules are written in the same formalism as the information to be processed, in RDF, thus allowing to add semantics to a model in the form of rules, and which must be able to be executed in several environments (i.e. that may be used in various triplestore and not only depend on a specific rule engine). Finally, we would like the rules that can be written to be rich and complex, allowing for example to create new individuals qualified by properties, all in one rule.

One of the first technologies that comes to mind when talking about semantic rules, applied to semantic Web and RDFS/OWL ontologies, is probably Semantic Web Rule Language (SWRL). The proposal for SWRL was made to W3C by Horrocks et al. (2004). It is, in the words of the W3C, a "member submission" and not a "recommendation", indicating that the proposal has not gone through the standardization process. This proposal is particularly interesting because it directly targets the OWL language but one of its weak points is the fact that it predates OWL 2. Discussions have taken place to update SWRL to support OWL 2, but this has not resulted in a new version of the proposal. Finally, few tools support this proposal.

Additionnaly, a W3C recommendation concerning the exchange of semantic rules: it is called RIF (Rule Interchange Format). RIF is often visible on the cake layer representations of the Semantic Web (Figure 3.1), acting as a logical layer of rules. However, in the end this is essentially a specification for writing and exchanging rules in the Semantic Web, not for executing rules at the application level. For this reason, we have not gone into further detail about this solution. Another existing solution with some recognition in the community is Notation 3 (N3) Logic (Berners-Lee et al., 2008). However, this proposal has the disadvantage of being strongly linked to the N3 serialization format. Moreover, neither RIF nor N3 Logic are integrated by default in the triplestores and frameworks for the Semantic Web that we will present later. It should be noted that discussions are still taking place in W3C working groups concerning this lack of a semantic rules language and that SPIN/SHACL-AF rules are mentioned as possible solutions[22].

We have considered SPIN and SHACL-AF that are two recent proposals that benefit from implementations and some recognition in the community. These two

[22]See https://github.com/w3c/EasierRDF/issues/27 for example.

proposals are focused on RDF and not on OWL like SWRL, but this does not prevent them from being used on RDF graphs containing RDFS/OWL information. Finally, these two proposals, by their nature, allow for the encoding of more complex rules than those mentioned above.

3.2.1 SPARQL Inference Notation

SPARQL Inference Notation (SPIN) is a W3C member submission by Knublauch et al. (2011) allowing the expression of business rules using SPARQL. SPIN enables notably to express how to compute the value of a property based on other properties and under which conditions a set of rules has to be executed.

The SPIN proposal seems to have aroused real interest in the community and one can find various scientific works that use this proposal as an inference engine. As a matter of fact, this proposal is implemented in certain products such as AllegroGraph or TopBraid (we will come back to these points below in a dedicated section). Another indicator of the interest in SPIN is the ecosystem that has been developed around it[23] with proposals such as SPARQLMotion (a declarative language that is used to define data processing pipelines), SPINx (SPIN JavaScript Functions) and SWP (which is used to link domain concepts with user interface descriptions).

We do not go into much more detail about the functionality of SPIN as SHACL and SHACL-AF, presented in the next subsection, is often seen as a successor to SPIN and makes broadly the same actions possible[24] while being built on the vocabulary of the SHACL recommendation.

3.2.2 SHACL Advanced Features and SHACL JavaScript Extensions

The RDF Data Shapes Working Group has also elaborated proposals allowing the definition of rules, both in RDF (with a dedicated vocabulary and based on SHACL-SPARQL) and in JavaScript. These are SHACL Advanced Features (Knublauch et al., 2017) and SHACL JavaScript Extensions (Knublauch and Maria, 2017), respectively referred to in their abbreviated form as SHACL-AF and SHACL-JS. Both are W3C working group notes published at the same time as SHACL recommendation and that are based on its principles and vocabulary.

SHACL-AF extends SHACL specification by defining RDF vocabularies that allow new functionalities such as the definition of *SHACL functions*, which are reusable blocks containing complex operations described in SPARQL, and the definitions of *SHACL rules*, which enables creating new triples from existing ones. In this way, the

[23] See https://spinrdf.org/spinstack.html for further information.
[24] As explained on https://spinrdf.org/spin-shacl.html for example.

inference mechanism that can be described is very rich and allows, for example, the creation of new individuals and the calculation of values based on the characteristics of an individual.

Back to the example model presented in Listing 3.6, let us assume that classifying the individuals of type `ex:Peak` according to their height is needed. This action is possible with a SHACL rule of type `sh:TripleRule` that allows expressing the whole rule in the form of triple. This is also possible using `sh:SPARQLRule` that enables more expressive rules to be written with SPARQL queries using the `CONSTRUCT` clause (Listing 3.11). Using `sh:SPARQLRule`, multiple triples can be created at once, as well as new individuals.

```
ex:ClassifyPeak
    a sh:NodeShape ;
    sh:targetClass ex:Peak ;
    sh:rule [
        a sh:SPARQLRule ;
        sh:construct """
      CONSTRUCT {
        $this a ?typePeak .
      } WHERE {
        $this ex:hasHeight ?value .
        BIND(IF(?value > 1500, ex:HighPeak, ex:LowPeak) AS ?typePeak).
      }
        """ ;
    ] .
```

Listing 3.11: Example of SHACL-AF rule

SHACL-JS also extends the SHACL specification but this time by defining an extension mechanism based on the JavaScript language. It is thus possible to define, as JavaScript functions, the various features of SHACL such as constraints, targets and rules. These definitions are, just like with SHACL-AF, linked in a shape graph to be used by a SHACL-JS compliant engine. An interesting aspect is that these functions can be located in local files or in files that can be resolved from the web.

As with SPIN, the adoption of SHACL Core in many tools (see Section 3.3) and the progressive adoption of SHACL-AF and SHACL-JS in some tools demonstrates the interest from the community and the fact that these proposals strongly fulfil a user need.

In the context of the work we are carrying out here, we opt for SHACL-AF, in particular because it has the advantage of only requiring the use of Semantic Web technologies for the specification of rules (which can therefore be in RDF or in the form of SPARQL queries).

3.3 Existing implementations

The choices of languages we made for our approach have also been based on implementations that exist for the previously mentioned Semantic Web technologies. We present here a review of the main technologies concerning the three central points of the Semantic Web stack. The implementation of these technologies usually relies on what is known as a *RDF store* or *triplestore* (3.3.1). This is a type of database specially designed to store information in the form of a triple and to retrieve this information through a query mechanism, typically using the SPARQL language presented earlier. *Frameworks for Semantic Web programming* (3.3.2) help to build powerful Semantic Web applications. They can be used to manage the parsing of RDF documents, their loading or merging into a graph, sometimes through a connection to one of the triplestores mentioned above. They make knowledge interrogation mechanisms available and allow the implementation of more advanced techniques such as actions triggered by changes in the graph. They also usually propose some kind of reasoners or rule engines (3.3.3) to materialise the knowledge that results from the different levels of inference presented previously (RFDS and OWL in its various flavours). In addition other inferences can be drawn using a rule engine. Both reasoner and rule engine can be included in the choosen triple-store solution or can be used independently (on the command line, via a programmatic API). We also review them alongside the triplestores that implement them or independently depending on their nature. Another important point that is usually raised is that of *editors* (such as Protege software[25] or TopBraid Composer[26]), which we have deliberately chosen not to address.

For these three particular points (triplestores, frameworks for Semantic Web programming and reasoners/rules engines), a first selection has been made to select tools to be discussed, guided by the following expectations :

- being a well-known, freely available solution that implements some of the standards mentioned above (notably RDFS, OWL and SPARQL)
- offering a programmatic API in a popular language (allowing for example to implement a home-made logic at the level of the inference engine or to trigger specific actions during query or update)
- giving access to spatial functionalities (such as GeoSPARQL)
- allowing the use of a rule engine (ideally with one of the mechanisms presented above)

Part of our criteria is shared with existing work (such as Hebeler, 2009; Garbis et al., 2013; Huang et al., 2019). However, the comparison we provide here is not based on these works, which more specifically evaluate the performance of geospatial RDF stores (Garbis et al., 2013; Huang et al., 2019) or aim at identifying a suitable framework for Semantic Web programming (Hebeler, 2009) but without requiring geospatial capabilities. Furthermore, Allemang and Hendler (2011) discuss

[25] https://protege.stanford.edu/
[26] https://www.topquadrant.com/products/topbraid-composer/

the essential elements that frameworks for the Semantic Web must have, but do not mention any of them. We adopt some of their criteria for our analysis, such as the capabilities offered by the RDF store and by the RDF query engine, the presence of a parser/serialiser supporting many formats and the ability to interact programmatically with the RDF store or directly with the models.

We also discuss later why the programmatic API criterion is essential for our current work, but also why we believe that this point could become irrelevant in the future.

3.3.1 Triplestores

The triplestore is the core element of a system based on Semantic Web technologies since it is the one that stores knowledge in the form of RDF graphs and provides access to this knowledge through one or several query mechanisms such as SPARQL. All of those listed here meet the above criteria (such as supporting spatial functionnalities through GeoSPARQL), however there are important differences in terms of their licenses and the level of control natively possible on these triplestores (such as the use of a rule mechanism). The elements of this sub-section are summarised in Table 3.4.

GraphDB [27] is also a graph database and triplestore, developed since 2000. It is also released under a proprietary licence but it is freely available. It offers different profiles of reasoning such as RDFS-Plus, OWL 2 QL and OWL 2 RL.

Apache Jena is a complete open-source Java framework for building Semantic Web application. Without using its programmatic API (which will be detailed in the next section), it is possible to use its Fuseki[28] component that will be used to create a SPARQL endpoint and a web administration interface for data stored in using TDB (a Jena component offering persistent storage). It supports RDFS and various flavours of OWL 1.1 as well as OWL 2 using Openllet reasoner (see below).

StarDog[29] is a graph database and a triplestore released under a proprietary licence but also freely available. It supports reasoning in RDFS and in OWL 2 (DL, EL, QL, RL) and also partially supports GeoSPARQL since recently[30]. We can also note that Stardog supports the writing of custom SPARQL functions in Java as well as various interesting functionalities, not necessarily related to our problems, such as the support of GraphQL.

Strabon[31] is an open-source triplestore. One of the specificities of Strabon is that it primarily supports the stSPARQL query language which is based on the rep-

[27] https://www.ontotext.com/products/graphdb/
[28] https://jena.apache.org/documentation/fuseki2/
[29] https://www.stardog.com/
[30] https://www.stardog.com/blog/geospatial-a-primer/
[31] http://strabon.di.uoa.gr

resentation language stRDF, which both are specifically designed for spatiotemporal data. It also supports, partially and since recently, GeoSPARQL. However, to our knowledge, it does not support a ready-to-use RDFS/OWL inference solution.

Virtuoso Open-Source Edition[32] is an open-source triplestore developed as a parallel to the commercial product Virtuoso[33] (we did not include it in the comparison because its free use is limited in time). Virtuoso Open-Source Edition supports RDFS as well as some OWL constructs (basically RDFS-Plus) but does not offer full support for any OWL 1.1 or OWL 2 profile.

GraphDB, StarDog, Strabon and Virtuoso Open-Source Edition are complete triplestores products. They are not directly aimed at providing an API for Semantic Web programming (although some of them can be used though Jena or RDF4J APIs) ; they are rather intended to be configured by configuration files and administered via a graphical interface or via HTTP. In contrast, while Jena Fuseki is also intended to be parameterised by configuration files, it is also easily accessible from the Jena programmatic framework. The other difference relates to the licensing and distribution of these products. GraphDB and StarDog are proprietary products and are distributed rather monolithically. Because it is an open-source framework, Jena offers a more modular structure.

Other popular and reputedly high-performance solutions are not mentioned here because they do not support GeoSPARQL or because they are not available freely and without time restrictions (such as RDFox[34] or AnzoGraph[35]).

	Standards implemented	Programmatic API	Rule engine	Licence
GraphDB	RDFS, OWL, SPARQL	Java (Jena, RDF4J)	Native (custom notation)	Proprietary, free to use
Jena Fuseki	RDFS, OWL, SPARQL	Java (Jena)	Native (custom notation)	Open-source (Apache 2.0)
StarDog	RDFS, OWL, SPARQL	Java (Jena, RDF4J)	Native (custom notation) and SWRL	Proprietary, free to use
Strabon	SPARQL	Java (RDF4J)		Open-source (Mozilla Public License 2.0)
Virtuoso	RDFS, OWL, SPARQL	Java (Jena, RDF4J)	SPIN	Open-source (GNU GPL-2.0)

Table 3.4: Triplestores comparison.

The elements presented in Table 3.4 show that several choices are possible if one wishes to store and query spatial information and apply the deductions of a model

[32] https://github.com/openlink/virtuoso-opensource/
[33] https://virtuoso.openlinksw.com/
[34] https://www.oxfordsemantic.tech/
[35] https://www.cambridgesemantics.com/anzograph/

written in OWL. However, for the implementation of our approach, we also need to consider a framework for Semantic Web programming. Indeed, we will need its functionality for tasks such as loading models in various serialisation, integrating RDF data from the web and implementing a mechanism of semantic rules with a fine-grained control over it.

3.3.2 Frameworks for Semantic Web programming

The frameworks (here heard as library of computer code) we present here are all available under open-source licenses. In general, they allow the parsing and the serialisation of many formats defined for the Semantic Web, the merging of several schemas or knowledge bases, as well as the programmatic expression of statements. They also generally allow interaction with the SPARQL engine used, for example to define custom SPARQL functions that can then be used in queries. Here we observe their differences: the language they use, if they permit to use one of the triplestores mentioned above as a storage solution or if a storage solution is included, if they need to add a brick for reasoning, etc. We also remark that most of these APIs are centred on RDF technology (making users manipulate concepts such as *Resource*, *RDF Graph*, etc.) while some others are centred on the OWL language (making users manipulate concepts such as *ontology*, *axiom*, etc.). The elements of this sub-section are summarised in Table 3.5.

	Standards implemented	Programming language	Spatial functionalities	Validation	Native rule engine
dotNetRDF	RDFS, SPARQL	C#	GeoSPARQL	SHACL	SPIN
Jena	RDFS, OWL, SPARQL	Java	GeoSPARQL	SHACL	Custom notation
OWL-API	OWL	Java			SWRL
Owlready	OWL	Python			SWRL
RDF4j	RDFS, SPARQL	Java	GeoSPARQL	SHACL	SPIN
RDFLib	RDFS, SPARQL	Python		SHACL	Partially SHACL-AF
Redland	SPARQL	C (with bindings in C#, Java, Perl, PHP, Python, Ruby)			

Table 3.5: Semantic web framework comparison.

dotNetRDF [36] is an open-source C# framework providing a RDF-centric API. It does not have GeoSPARQL among its native functionalities but allows the use of

[36] https://github.com/dotnetrdf/dotnetrdf/

triplestores that do (like GraphDB). Its implementation of SPARQL 1.1 is complete and allows the writing of custom functions in C#.

RDF4j [37] is an open-source Java framework for building Semantic Web and linked data applications. As dotNetRDF, it does not have GeoSPARQL among its native functionalities but allows the use of triplestores that do (like Strabon or Stardog). Indeed, RDF4J is probably the framework that allows to use the largest variety of triplestores. It also has a dedicated reasoning interface and provides an RDFS reasoner but not a ready-made OWL reasoner (but again, it allows the use of a storage layer that has one, such as GraphDB). It supports writing custom SPARQL functions in Java.

RDFLib[38] is an open-source Python library for working with RDF. It supports many serialisation formats and provides an implementation of SPARQL 1.1 for which it is possible to write custom functions in Python. It is part of a collection of volunteer-maintained libraries adding various building blocks to it, making it a real framework for the Semantic Web ; such building blocks are for example, a reasoner supporting RDFS and OWL 2 RL and a validator implementing SHACL and we note that the implementation of SHACL-AF is in progress. However, no GeoSPARQL implementation is available at the time of writing.

Jena [39] is an open-source Java framework for building Semantic Web and linked data applications. It supports many serialisation formats, provides various storage solutions and has an implementation of SPARQL 1.1. Although it has an RDF-centric API, it provides an API for ontology manipulation and an API for RDF graph validation and reasoning, with several reasoners included and a rule engine using a custom notation. Jena does not natively propose OWL 2 reasoners but only OWL 1.1 compatible reasoners. This consideration, that is the main point against Jena, is not necessarily a limitation. On the one hand the use of the non-standard RDFS-Plus profile is possible with Jena's reasoners and on the other hand some OWL 2 documents only use constructs that also belong to OWL 1.1, making them directly usable. Finally there are several possibilities to support OWL 2, like using a reasoner such as Openllet that supports all profiles of OWL 2 or implementing OWL 2 RL through a set of rules[40].

Redland librdf[41] is an open-source C framework providing support for RDF. It offers the various useful blocks such as serialisation / deserialisation of many formats, a storage layer based on a choice of various solutions, a query module supporting SPARQL (however, not all of SPARQL 1.1 seems to be covered). In addition and to our knowledge it does not provide a plug-and-play reasoner. The library is stable but, unlike the others frameworks here that are all actively maintained, it does not appear

[37] https://rdf4j.org/

[38] https://github.com/RDFLib/rdflib

[39] https://jena.apache.org/

[40] Such an implementation can for example be found on the web: https://github.com/william-vw/owl2rl-jena.

[41] https://github.com/dajobe/librdf

to be undergoing any recent development (since 2015). Furthermore, it does not seem to allow the writing of custom SPARQL functions in a programming language.

OWL-API [42] is a Java library for creating, manipulating and serialising OWL 2 ontologies, with reasoner interfaces for working with all the various level of expressivity (Full, DL, EL, QL, RL). We mention it here because, although it is not really a framework for the Semantic Web, it is the most complete Java tool for manipulating OWL ontologies and it is also the first OWL-centric library that we present here. Moreover, this library can be used with ONT-API[43], a RDF-centric implementation of OWL-API based on Jena framework. This makes it possible to combine the programming interfaces of OWL-API and Jena to interact directly with ontological graphs.

In the same line as OWL-API, we can also mention Owlready[44]. It is an OWL-centric Python library to implement what is described as ontology-oriented programming (classical Python constructs are used to manipulate ontologies, e.g. Python classes represent OWL classes, class attributes represent OWL properties, etc.). It also allows some interaction with RDFLib, making it possible to manipulate OWL models with Owlready and RDF data with RDFLib while remaining in the same technical environment.

This sub-section make it easier to identify what spectrum of actions frameworks for Semantic Web programming can cover. We also notice that the adoption of some frameworks will still require choices to be made about the reasoner to be used (for RDFS/OWL inferences) and about a possible semantic rule system. Indeed, several of the libraries presented here (OWL-API, Owlready and Jena to some extent) offer by default to choose among several pluggable and provided reasoners. Furthermore, while several of these libraries offer native support for SPIN, none of them offer full support for SHACL-AF. We present the choices offered by the ecosystem in the following subsection.

3.3.3 Reasoners and rules engine

We have seen that solutions allowing the application of RDFS and OWL entailments already exist within triplestores or within programming frameworks for the Semantic Web. However, they are not necessarily complete, especially with regard to OWL. There are other reasoning solutions that can either connect to the aforementioned frameworks or be used in other ways (as a Command-Line Interface (CLI) tool, as a web service, etc.). First we present several OWL reasoners and then several semantic rule systems.

[42] https://github.com/owlcs/owlapi
[43] https://github.com/owlcs/ont-api/
[44] https://bitbucket.org/jibalamy/owlready2

ELK[45] is a Java reasoner supporting the OWL 2 EL profile. It can be used from OWL-API library or as a CLI tool. This library is still actively maintained.

Openllet[46] is an opensource reasoner with a strong reputation. It supports all OWL 2 features and profiles as well as SWRL rules. Openllet is community-maintained and build on top of the Pellet reasoner (which is available in the Protege tool for example). It can be used with Jena (both in the Fuseki triplestore and in the Java framework) and with OWL-API for example.

Konclude[47] also handles reasoning using the OWL 2 DL profile, but it is intended to be used mostly by CLI or using OWLink specification[48], which nevertheless makes it usable with OWL-API without having a proper OWL-API interface. However, it cannot be used directly from Jena. Konclude is maintained and under development and there is a special focus on the performance it offers. It reuses blocks from the Redland librdf framework presented earlier.

Other frequently mentioned reasoners are those that can be used directly with OWL-API. For example Fact++[49] is an open-source reasoner for OWL written in CPP and JFact[50] is its Java port. HermiT[51] is another open-source reasoner. All three can be used in the OWL-API library and support all OWL 2 features and profiles.

We have cited several existing OWL reasoners that may be useful. This is a broad field and selecting a reasoner that is suitable for the needs of a Semantic Web application can be challenging. This is exemplified by the work of Tai et al. (2011), which is specifically dedicated to developing a computer-aided approach to assist in the selection of an appropriate reasoner. The authors note that this task is essentially based on discussions between developers and experts and is the object of an overhead communication. We also hope to contribute to the knowledge on this subject through the practical and up-to-date review we provide here.

Concerning semantic rule systems, we have previously mentioned SPIN and SHACL-AF. The reference implementation of these two technologies exists in the form of a CLI tool and of a Jena library named SHACL[52], developed and maintained by TopQuadrant. This library implements all SHACL specification documents.

Specific reasoning engines that do not depend on Semantic Web technologies and that are freely available also exist. Some of them can be easily used with knowledge represented in RDF. This is for example the case of Drools[53]. It is a Business Rules

[45] https://github.com/liveontologies/elk-reasoner
[46] https://github.com/Galigator/openllet/
[47] https://github.com/konclude/Konclude
[48] https://www.w3.org/Submission/2010/SUBM-owllink-structural-specification-20100701/
[49] http://owl.man.ac.uk/factplusplus/
[50] https://github.com/owlcs/jfact/
[51] http://www.hermit-reasoner.com/
[52] https://github.com/TopQuadrant/shacl/
[53] https://www.drools.org/

Management System (BRMS) with a forward and backward chaining inference based rules engine. However, this kind of solution is not based on well-known standard proposals, such as SWRL or SHACL-AF and the rules written for this system can only be used by the Drools engine. Moreover, it does not allow adding semantics, in the same formalism as the one used to represent knowledge, RDF, together with the targeted domain knowledge. We therefore did not consider such solutions.

3.3.4 Focus on Jena capacities and specifics

The observations made so far lead us to focus more specifically on the Apache Jena framework. It has a large number of major advantages: it is released under a free and permissive licence, it has quality documentation and above all it easily allows the use of two elements that we have chosen so far, GeoSPARQL and SHACL-AF (through its reference implementation presented above).

Indeed, while RDF4J shares many of its strengths with Jena, it should be noted that on the one hand it does not support OWL natively and on the other hand it does not provide an implementation of SHACL-AF. Alternatively, a solution based on Owlready and RDFLib could have been also particularly attractive because of the friendliness of the APIs offered by these two libraries and because of the strength of Python ecosystem in enabling rapid prototyping and application development. However, this solution does not support two of our requirements, namely GeoSPARQL and SHACL-AF support.

Echoing the point mentioned above, we can note that Jena allows :

- to choose a way to store data (using TDB storage, connection to other triplestores such as GraphDB or StarDog),
- to choose a reasoner among those proposed by the library (allowing to use the RDFS-Plus subset) or coming from a third party library (such as Openllet for better support of OWL 2),
- to validate RDF data using SHACL,
- to choose a rule mechanism among those proposed by the library, those supported by reasoners (such as SWRL which is supported by Openllet) or by third-party libraries (such as SPIN and SHACL-AF),
- to use GeoSPARQL (spatial functions, spatial predicates and RDFS entailment among other features),
- to be used with ONT-API which gives a better access to OWL 2 features while stile having the RDF-centric API of Jena.

Furthermore, the choice of a framework offering a strong programmatic API coupled with the storage solution has essential interests for us compared to choosing only a triplestore. Indeed, we will need to perform transformations on the models when they are loaded, to implement a custom logic for merging the graph containing the SHACL-AF inferences, and to implement actions triggered by events on the RDF

graph. All these actions are possible with Jena.

Other points, which have not been mentioned previously, also play in favour of this framework. Indeed, it provides an easy mean of exposing data graphs as SPARQL endpoints and it also make it possible to define personalised functions that can be called in SPARQL queries. These personalised functions can be of various types (filter functions, aggregate functions, etc.) and they can integrate a geospatial dimension (for example by using JTS[54], a core library for vector-based geomatics), and therefore be perfectly integrated into a workflow using GeoSPARQL.

Beyond these aspects, the library is constantly improved and offers access to technologies or standard proposals that we have not mentioned so far but which could be of interest later in the context of our proposal (Jena supports RDF Star[55] since its version 4.0.0, dated 27 March 2021). Moreover, Jena is used under the hood by several commercial products (such as TopBraid EDG and TopBraid Composer), which demonstrates its reliability.

Thus, Jena makes it possible to completely build applications for the Semantic Web using the technologies we need and giving fine control over them. This point is also made in the literature where Jena is also sometimes identified as a first choice framework for implementing Semantic Web applications (Hebeler, 2009). The performance of the GeoSPARQL implementation for Jena also looks respectable in published benchmarks comparing it to other products (Huang et al., 2019).

All the elements presented in this chapter have given us a good understanding of the technical space of the Semantic Web and allow us to make an informed choice regarding the different elements of an architecture for these technologies. In the next chapter, we will discuss the scientific works that have proposed to formalise the knowledge of the geovisualisation domain as well as those that propose to exploit the data of the Semantic Web to create these geovisualisations.

[54] https://github.com/locationtech/jts

[55] Of which the last published draft can be found online: https://w3c.github.io/rdf-star/cg-spec/

Chapter 4

Geovisualisation generation from semantic models: a requirements analysis

4.1 Introduction

In Chapter 2 we showed that geovisualisation is a first-choice approach when it comes to support a user's reasoning when exploring geographical information or when solving a spatial problem. We have also identified the important components to be implemented in order to create a geovisualisation tool that meets these needs, and more specifically our needs within the CHOUCAS project.

In Chapter 3 we detailed the foundations of Semantic Web technologies. Indeed, more and more data relevant for geovisual analysis is structured and is published in the Web, sometimes data is even directly described using the RDF formalism and made available through SPARQL endpoints. We have also seen in the previous chapter that RDF offers a powerful framework for knowledge management, beyond the description of Web data.

The trend mentioned above requires that the methods, techniques and tools of geovisualisation be able to use natively these data expressed in the languages of the Semantic Web. Thus, our work is situated at the intersection of these two fields (Figure 4.1).

In this chapter, we explore approaches that can be used to automate the geovisualisation of data located in the Semantic Web or of data described by Semantic Web models, which already are contributions to Semantic Web-based Geovisualisation.

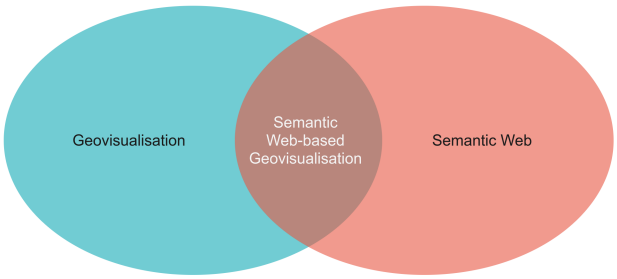

Figure 4.1: Semantic Web-based Geovisualisation.

During our review of the literature on this subject, we found numerous studies aiming at evaluating the contribution of Semantic Web technologies in order to encode and to exploit knowledge with a geospatial component. Among them, researches focus on:

- formalisation of such a geospatial knowledge (for example by Abadie et al., 2010; Janowicz et al., 2013; Zhang et al., 2015; Scheider et al., 2019),
- its querying (Zhang et al. cover this issue extensively in their book Zhang et al., 2015),
- its integration (on which Maué and Schade, 2009, Zhang et al., 2010 and Prudhomme et al., 2020 offer interesting findings for example)

Regarding more specifically issues dealing with visualisation, challenges were identified rather early (e.g. by Fabrikant, 2001) but the use of Semantic Web technologies for the specific purpose of building geovisualisation is rarely addressed when geospatial semantics issues are tackled (e.g. by Janowicz et al., 2013 and by Hu, 2018).

By taking an interest in formal approaches aimed at using ontologies to specify both the task to be carried out in the geovisualisation and the graphical elements that can be displayed, all this being instrumented by a mechanism of semantic rules, we are particularly in the spirit of the conceptual proposals made by Sara Fabrikant which notably states that "*an ontological approach to visualization seems particularly adequate for formalizing semantic and semiotic transformation components of web-enabled geovisualization*" (Fabrikant, 2001) in what can be seen as a position paper.

In the next sections, we present a bibliographical review of the various works that have addressed the issues of cartography and geovisualisation of data formalised or published using Semantic Web technologies, as well as approaches that have used ontologies to formalise the essential elements of a geovisualisation, in order to identify the limitations of these works but also the concrete elements that can be reused in our approach and what they teach us in terms of design choices.

4.2 Geovisualisation generation approaches for Semantic Web models or data

In the literature, various approaches aim to support or to automate the creation of maps or geovisualisations and to adapt their contents to the user or to some kind of context using semantics or rule-based approaches. Some of them are based on the exploitation of linked data, either to enrich the content of maps or to directly exploit a remote dataset to be portrayed.

4.2.1 Approaches to assist in the choice of an appropriate cartographic portrayal

Iosifescu-Enescu and Hurni (2007) identify the lack of mechanisms for formalising cartographic rules as one of the elements that can impact on the cartographic quality of the maps produced. They have an approach based on two OWL ontologies, the *Cartographic domain ontology* (Figure 4.2) which describes the vocabulary needed to describe maps, and the *cartographic task ontology* which describes the cartographic rules to be implemented (these rules are only presented in natural language in the article). They also describe a generic architecture of a cartographic library that can be used for a symbolisation process based on these ontologies. The advantages they see in this include the possibility of formalising mapping rules independently of any specific mapping software. Software based on these ontologies would not need to write additional code when adding new representation functionalities but would simply exploit the updated ontologies in question. It is also worth noting that they wished to separate the vocabulary proposals relating to mapping from the rules that use them. This makes it possible to reuse these vocabularies in another context, and to have different sets of cartographic rules depending on the desired purposes. Finally, as a significant limitation to this work, we note that their ontologies do not seem to be published on the Web and that the generic architecture of a cartographic library that is described is not yet implemented.

The work of Smith (2010) proposes ontologies for the construction and the successful execution of an expert system dedicated to thematic cartography. The ontologies are described in detail on the paper and reuse portions of those proposed by Iosifescu-Enescu and Hurni (2007). The result is a *Map Ontology* (Figure 4.3) in which he identifies and organises the following concepts: *Attribute*, *Graphic*, *Layout Element*, *Map*, *Map Projection*, *Production Medium*, *Spatial Phenomenon* and *Visual Variable*. We note that the relations necessary to create a well-formed map (sometimes in the form of "has exactly one" restrictions for example) are encoded in its ontology. This formalisation is achieved in OWL and, as before, the lack of dependence on specific software is emphasised in this proposal. One of the main limitations of this work is the absence of an example of a map description using this vocabulary as well as the absence of a software implementation based on this vocabulary.

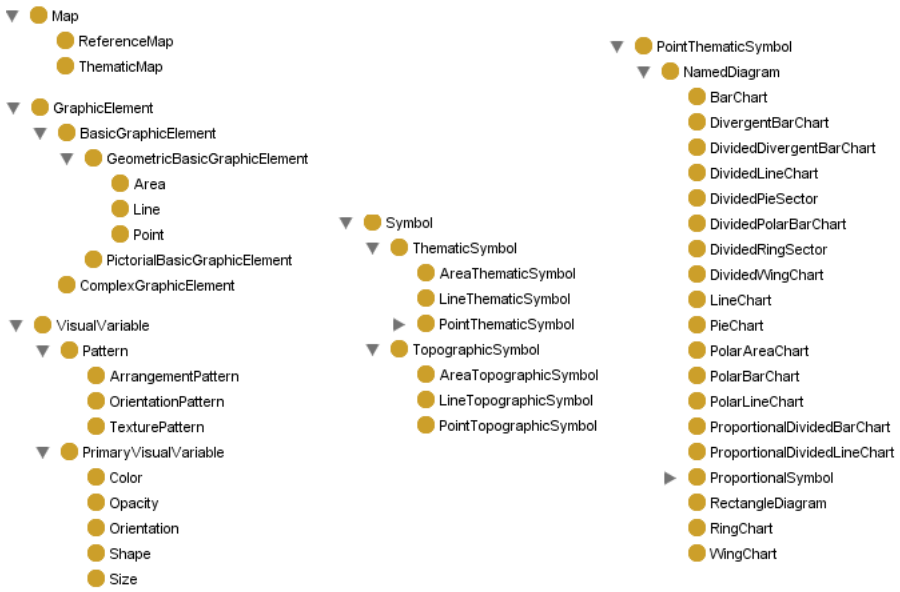

Figure 4.2: Details of the *Cartographic domain ontology* proposed by Iosifescu-Enescu and Hurni (2007). Source: Image from Iosifescu-Enescu and Hurni (2007).

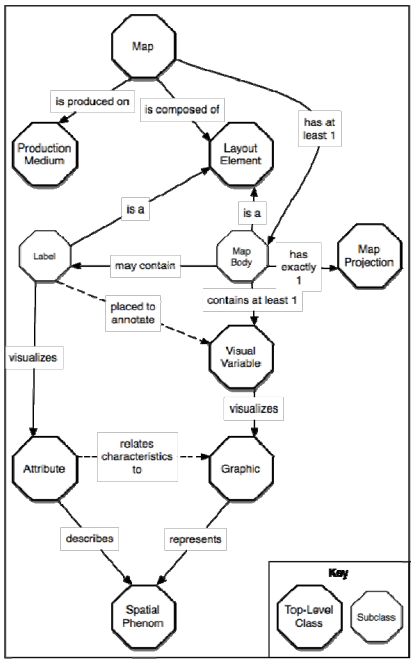

Figure 4.3: Details of the *Map Ontology* proposed by Smith (2010). Source: Image from Smith (2010).

Brus et al. (2010) are interested in designing an intelligent system for cartography. According to them, this intelligent system could assist users in creating correct maps. Indeed, the authors start from the fact that map making, which is the appanage of cartographers who know the rules, has become more democratic and that cartographers are not the only ones to produce maps. However, they believe that the production of correct maps is also a goal for non-cartographers. In order to meet these challenges, they assume that one possible solution is to store the knowledge specific to mapping in a knowledge base and to build an intelligent system based on this knowledge base. To this end, the authors first propose a cartographic ontology (Figure 4.4) based on the knowledge acquired from experts in the field: cartographers. The authors are then interested in the use of this knowledge in a system of rules: they choose Drools (mentioned in Section 3.3.3) and can thus define rules specific to cartography, using terms from their ontologies (one of the rules encoded in this way is visible as "when there exists PointFeature then insert PointSymbol"). We think that the approach is interesting, in particular the double hierarchy: that of data types (with the three subtypes *AttributeDate*, *DataTypeDomain* and *GraphicData*) on the one hand, and that of representation types (*CartographyMethod* on the other). Nevertheless, elements are missing to concretely qualify the different cartographic methods present in the hierarchy and no concrete implementation based on the ontology and derived Drools rules is shown. Moreover, we consider it is a drawback to encode the simple rules they present in a dedicated formalism that does not belong to the Semantic Web ecosystem.

Ruzicka et al. (2013) and Penaz et al. (2014) (who notably share a co-authorship) shed light on the possibilities of formalising declarative knowledge related to cartography in order to integrate it into an intelligent application based on a rules system. Both proposals use OWL as well as the Drools rule engine (mentioned in section 3.3.3) when it comes to integrating declarative knowledge with procedural knowledge. Penaz et al. (2014)[1] offer precise insights on modelling cartographic knowledge by adapting the use of the main OWL functionalities (property restrictions, inverse properties, functional properties) to their case study on thematic cartography. The goal of Penaz et al. (2014) is to be able to express as much information as possible about the maps to be created and in particular to express good cartographic practices in their ontological model (such as "the map page contains at least one element of map composition which is title and subtitle"). As their proposal is dedicated to thematic mapping (governed by the rules presented in Section 2.1) they also encode types of portrayals adapted to the type of data (so that they obtain a set of *QualitativeThematicMapMethod* and a set of *QuantitativeThematicMapMethod* for example, cf. Figure 4.5). The paper by Ruzicka et al. (2013) starts from the same observation as Brus et al. (2010) mentioned above: the fact that non-mapmakers also produce maps, without having knowledge of all the rules, unlike mapmakers. The description of the results of Ruzicka et al. (2013) is less detailed and focuses more specifically on the stage of integration of declarative knowledge into procedural knowledge, using Drools. The independence to a specific system is again mentioned as a wish but also as a limitation encountered by the authors: "*The first idea was to include as*

[1]This work is part of the same research project as the work of Brus et al. (2010)

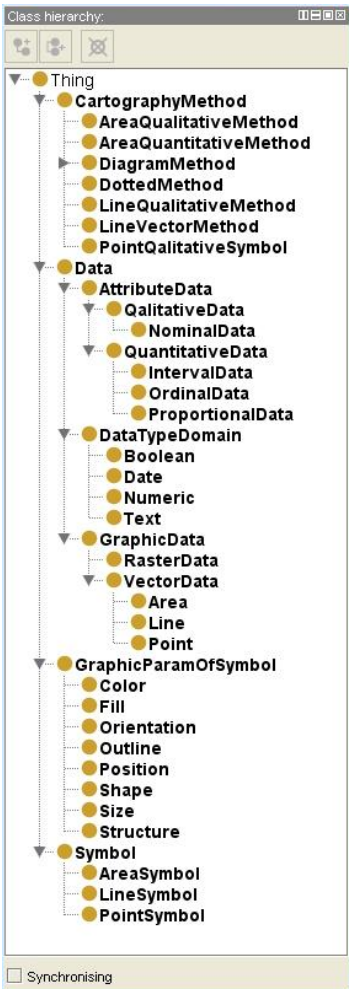

Figure 4.4: Details of the cartographic ontology Brus et al. (2010). Source: Image from Brus et al. (2010).

much as possible rules in the ontology. That approach results in a general solution that is independent on an expert system and re-usable in other projects that deal with cartography. Unfortunately there are several limits of the tools that allow to export an ontology." (Ruzicka et al., 2013). We note that the limitations they encounter seem to be mainly technical and related to Drools. However, we find the approach noteworthy for its willingness to encode rules alongside ontologies.

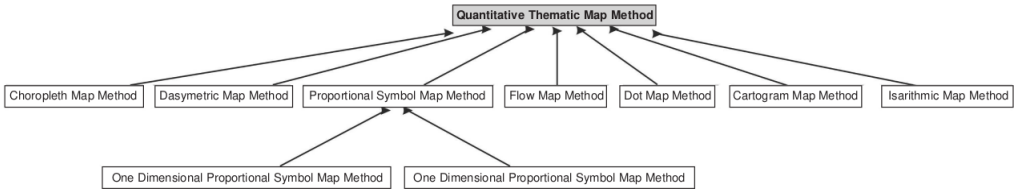

Figure 4.5: Classes hierarchy for quantitative thematic map methods in the experimental ontology of Penaz et al. (2014). Source: Image from Penaz et al. (2014).

Finally, the proposal of Vatin (2014) seeks to describe methods of geovisual portrayal in order to integrate them into an intelligent process that supports users in their use of geospatial information. The geovisual portrayals they address are complex and the author intends to propose useful and adequate geovisualisation techniques to the user, depending on the data to be represented, the task to be solved and the context of use. In this proposal, ontologies are used to indicate which portrayal can be suitable but they are not used to create and implement the map that host the suggested portrayals. The rules allowing these suggestions to be made are written using SWRL (see Section 3.2). Like Iosifescu-Enescu and Hurni (2007) and Smith (2010) for cartography, the author identifies that many of the models that formalise fields related to geovisualisation are tied to specific tools or languages, and that the proposal of ontologies and semantic rules overcomes these limitations.

4.2.2 Approaches aimed at adapting maps to the user context

Dong et al. (2008) propose an architecture conceived to produce a web-mapping application adapted to the user and to his or her visualisation context. The particularity of this work is to target the cartographic generalisation to highlight or not the elements of interest on the map. The authors use entity-relationship modelling to capture the personalisation of the map according to user requirements by considering their location, time, task, feature class (the type of information to be visualised), media (such as computer or mobile device) and other information about users' personal profile (language, interest, etc.). Based on this information, the architecture they propose is able to take into account geometric and semantic information of objects as well as the user requirements in every simplification task. The approach is said to be ontology-based but it is not clear if they are deployed during the use of the system or only for its preparation. However, we note that the authors also express

their desire to create a map that is tailored to support a particular task as well as the originality of the approach, which is clearly user-oriented and where most of the information related to geovisualisation is formalised at the user level.

Huang et al. (2018) tackles a problem encountered during the realisation of cartographic mash-ups, when the geometrical representations between the base-map and overlaying thematic data are not synchronised (Figure 4.6). This problem resonates strongly with difficulties that may be encountered in numerous geovisualisation projects requiring the integration of data from heterogeneous sources and their presentation to the user. Precisely, we believe that the synchronisation of overlaid elements with the background map, as presented by the authors, or even an appropriate selection of overlaid elements according to their presence or not on the background map, that we think is somewhat in the spirit of Dong et al. (2008) work. Huang et al. (2018) propose a relative positioning approach based on shared geometries between datasets to synchronise geometric representations for map mashups. To formalise and implement the relative positioning, the authors use ontologies that are based on the GeoSPARQL vocabulary and exploits linked data in order to organise the necessary knowledge. This approach is particularly interesting both for demonstrating the power of ontologies, and in particular GeoSPARQL, to solve complex tasks, and for providing a technical answer to a problem that can be encountered when creating geovisualisations.

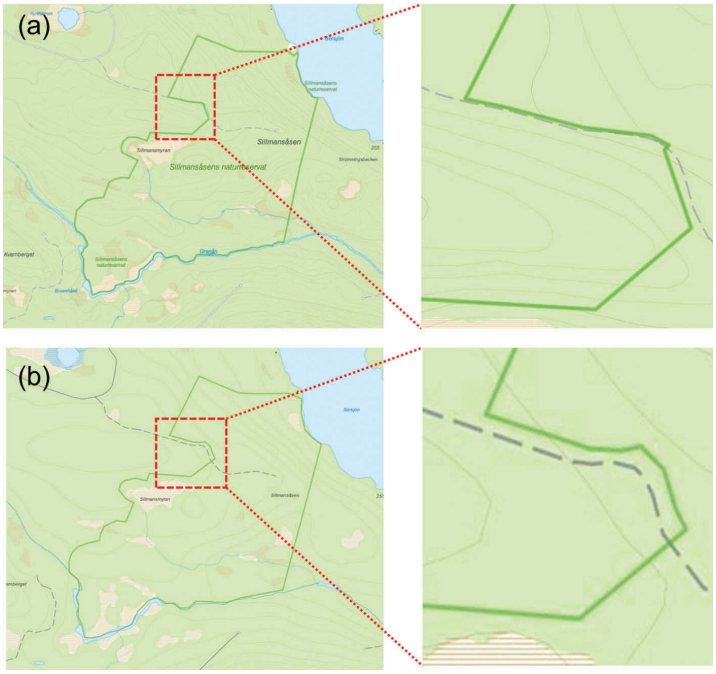

Figure 4.6: Lack of synchronisation of geometries between the features of the base-map and the overlaid feature. Source: Image from Huang et al. (2018).

4.2.3 Research for the publication of cartographic style information

In 2017, an OGC report that aims to specify semantic information models and REST APIs for Semantic Registry, Semantic Mediation and Semantic Portrayal Services is published (Fellah, 2017). This work is intended to be used as an integration solution to federate and unify the information produced by different OGC catalog services. For the part that interests us, this work has given rise to the formalisation of various semantic information models allowing the description of style information as well as the composition of graphic symbolizers. To this end, Fellah (2017) proposes:

- a *Style Ontology*, that defines the concept of Style and Portrayal Rules,
- a *Symbol Ontology* that defines the concepts of SymbolSet and Symbol (Figure 4.7),
- a *Symbolizer Ontology* (Figure 4.8) that defines the concept of Symbolizer (and its subclasses PointSymbolizer, LineSymbolizer, PolygonSymbolizer, TextSymbolizer, RasterSymbolizer and CompositeSymbolizer),
- a *Graphic Ontology* (Figure 4.9) that defines the different graphical elements that allow to create symbolizers.

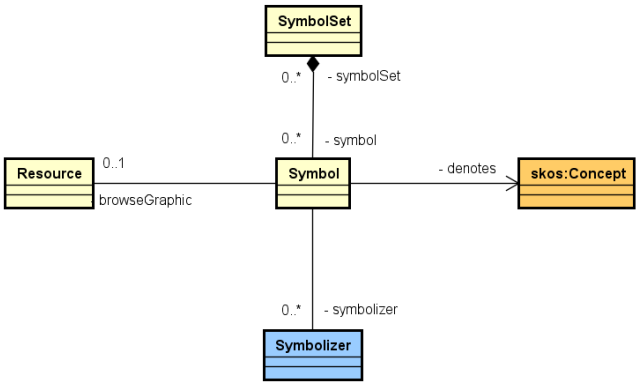

Figure 4.7: Overview of the *Symbol Ontology*. Source: Image from Fellah (2017).

Although these formalisations are based on the study of different specifications (such as the Scalable Vector Graphics format, MapCSS, CartoCSS and SLD/SE to mention only the best known and those mentioned in Chapter 2), it is possible to find many similarities in vocabulary and conceptual organisation between Fellah (2017) proposal and the SLD/SE specifications. The essential differences are in the formalism used by Fellah (2017), that of the Semantic Web, unlike SLD/SE which uses the hierarchical XML format, as well as the introduction of the concept of Symbol, absent from SLD/SE vocabularies. The concept of Symbol is defined to promote the reuse of symbols between different styles and allows the collection of symbols that are intended to be used together in symbol sets.

The ontologies described in this OGC Testbed are however not available directly with the report in a computer-readable form.

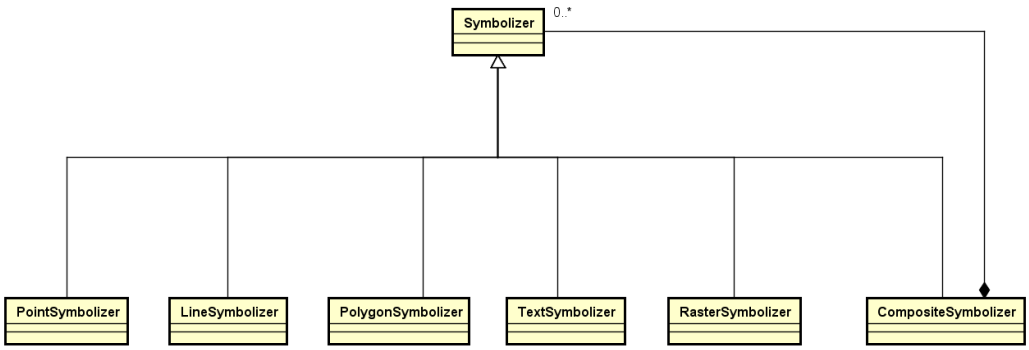

Figure 4.8: Overview of the *Symbolizer Ontology* hierarchy. Source: Image from Fellah (2017).

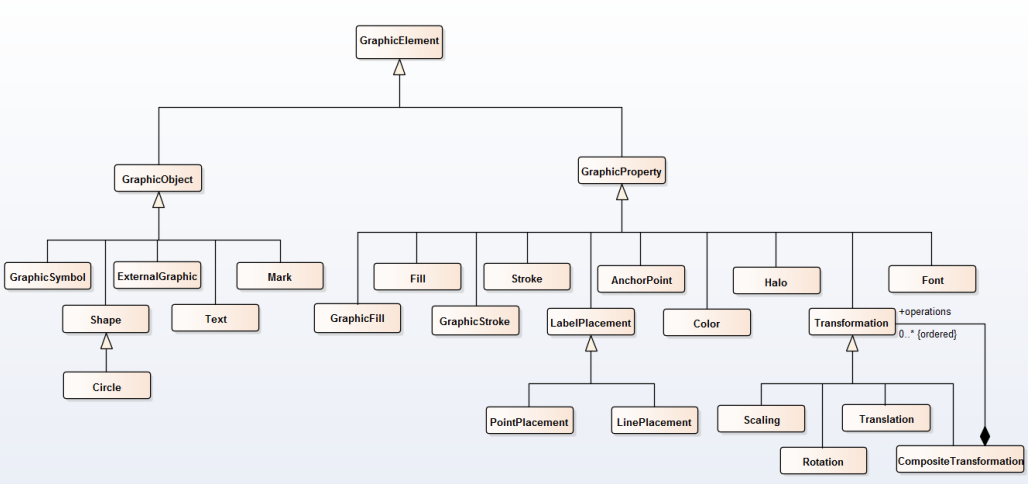

Figure 4.9: Overview of the *Graphic Ontology* hierarchy. Source: Image from Fellah (2017).

4.2.4 Approaches seeking to exploit the semantics of the data or the model

Based on the observation that existing vocabularies such as the W3C Geo Vocabulary and GeoSPARQL (cf. Chapter 3) provide the building blocks for publishing geospatial data in the Semantic Web, León et al. (2012) identified the need for a tool to explore and visualise RDF geospatial datasets. They thus propose *Map4RDF*. This proposal is notably aimed at organisations publishing geospatial data and proposes that they set up an instance of *Map4RDF* and connect it to their triplestore in order to provide a faceted browser to their users (Figure 4.10). The resulting geospatial browser can display the data grouped under different categories, here called *facets*, and interact with their entities (accessing detailed information, sharing the resource on Twitter, editing of the resource, etc.). We note that from the point of view of the administrator of *Map4RDF*, the definition of these facets is done in RDF, in a declarative fashion. The *Map4RDF* proposal makes it possible to use data described with GeoSPARQL and thus to display the different types of geometries supported by this vocabulary. Potential limitations include the fact that it does not offer advanced options for defining the selection of data according to some characteristics and that it does not allow defining styles using properties other than the colour. The code for this proposal is freely available on a Git repository[2] and there are running instances on the Web[3].

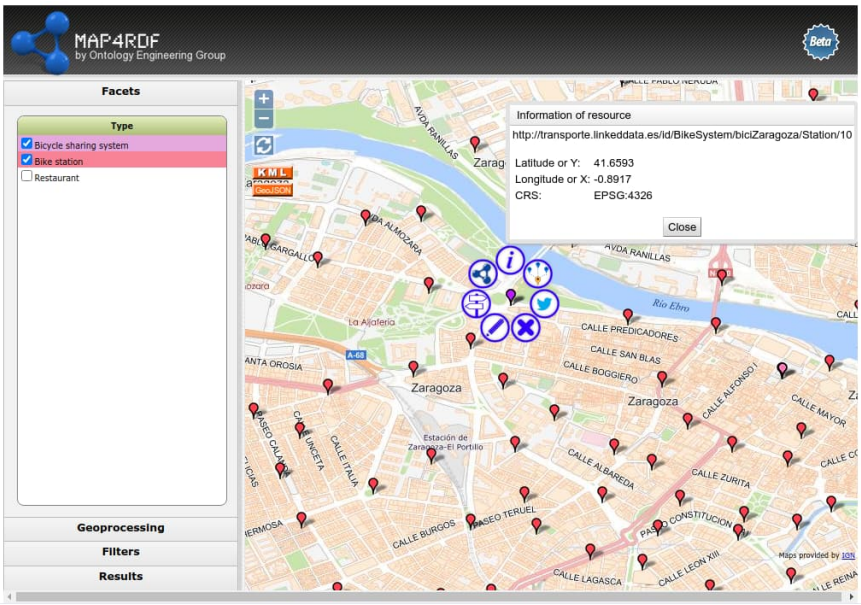

Figure 4.10: Example of interaction with an entity in *Map4RDF* (screenshot from http://transporte.linkeddata.es/browser.html - accessed on October 10, 2021).

[2] https://github.com/oeg-upm/map4rdf

[3] Such as http://certidatos.ign.es/map, http://transporte.linkeddata.es/browser.html and http://webenemasuno.linkeddata.es/browser.html.

Trillos Ujueta et al. (2018) note that no library seems to be able to process and display RDF data on a map. They therefore propose a JavaScript library, *RDF2Map*[4], which aims to display RDF individuals on a Leaflet map. RDF2Map offers an interesting level of abstraction since it allows to retrieve, from a turtle file loaded beforehand or from DBpedia, RDF individuals in the form of JavaScript objects directly usable by Leaflet. However, the filtering options are limited to the geometric type of the data. It is not possible to filter according to an RDFS/OWL class or other more complex conditions (value of a property, etc.), thus not exploiting the potentially rich semantics of the RDF data to be depicted. Furthermore, the library only targets geospatial data described by the NeoGeo vocabulary, whereas one might have expected GeoSPARQL to be targeted as well. While the authors mention perspectives such as the possibility of using data from other endpoints, this is not the case and the library has not been modified since the publication of the accompanying article. Finally, by its very nature, RDF2Map does not aim at and does not allow for the publication of symbology choices in RDF which can be a disadvantage for sharing and reusing symbology choices in the Semantic Web.

Varanka and Usery (2018) make a particularly interesting proposal from a conceptual point of view. They propose to design and implement what they call *the map as a knowledge base* so that both humans and machines can interpret it. The authors discuss the technical and architectural challenges facing this approach. The proofs of concept they implement allow them to conclude that advances in Semantic Web technologies and linked data make them viable for mapping, allowing for a map product in which RDF triples are accessible behind the visual vocabulary shown to the user. The authors show two ways of using the map as a knowledge base (Figure 4.11). The first involves using SPARQL queries to fetch the objects of interest to be mapped according to some constraints (they thus exploit a graph from the U.S. Geological Survey to ask, in SPARQL, the following data: "EPA pollution sites within 5 km of the Pittsburg, Missouri Volunteer Fire Station"). The second is using what they call the "browsable graph approach" (also described as a "follow-your-nose" approach by the authors) where new data is retrieved by selecting a resource from data already present on the map.

The technical choices made by Varanka and Usery (2018) encourage us on the direction we have chosen (with the use of SPARQL for example) as well as to go further (with an architecture that we believe could be simplified). While this approach is not directly aimed at exploiting the semantics of an existing data model, we can see how it can contribute to the creation of an intelligent geovisualisation infrastructure. Furthermore, we consider that seeing *the map as a knowledge base* is a strong and appealing concept and we recognise ourselves in this formulation, in the implications it underlies as well as in its ultimate purpose which is close to the Semantic Web: to allow interpretation by humans and machines. The main limitation we see is the lack of discussion about vocabularies to describe the graphical representation of objects shown on the map. This point is according to us indispensable in order to fully concretise the concept of *map as a knowledge base*.

[4] https://github.com/atrillos/Rdf2Map_library

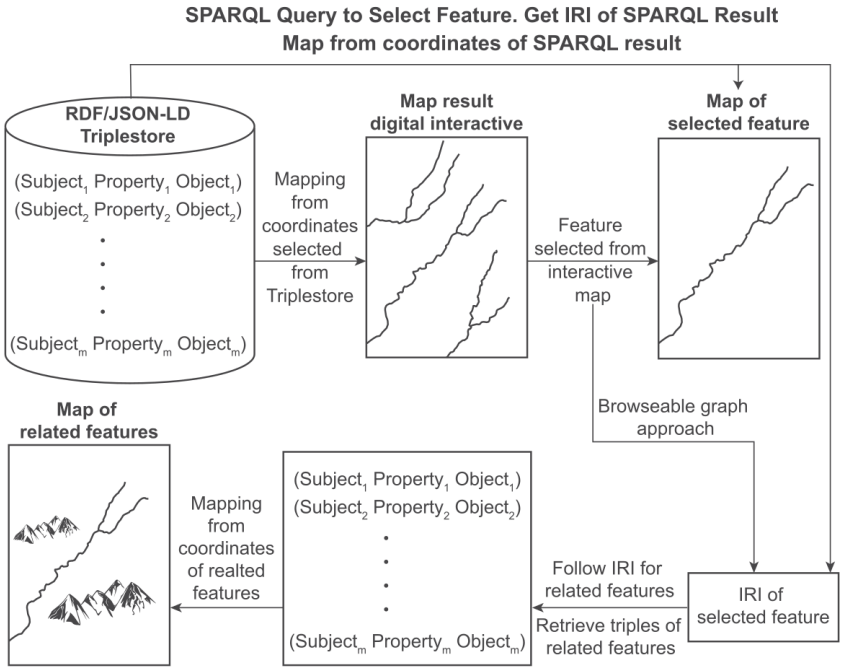

Figure 4.11: Illustration of the concept of *map as a knowledge base*, showing the two approaches to data retrieval. Source: Image from Varanka and Usery (2018).

These research perspectives have been pursued and have been the subject of another publication Wagner et al. (2020) in which the authors describe the implementation of a system aimed at exploring linked data that semantically integrate geospatial data from various sources. This provides more detailed information on the implementation of the system described by Varanka and Usery (2018) and shows it in case studies. While there is a strong emphasis on integrating data into their system, some interesting points are raised regarding methods of symbolization that are suitable with a cartographic user interface design that supports integrating linked data (in particular the grouped display of objects in clusters depending on the zoom level).

Huang and Harrie (2020) propose a system that allows formal representation of geovisual knowledge in a machine-readable way. They propose several ontologies covering the *cartographic scale*, the *data portrayal* and the *geometry source*. The vocabulary proposed for the scale is particularly interesting and is based on the work of Carral et al. (2013) and Huang et al. (2018) but differs from them by allowing the implementation of a conditional portrayal in which the symbolization depends on the visualisation scale and attribute/geometric data associated with the feature. The vocabularies proposed for data portrayals (Figure 4.12) are notably based on the work of the OGC Testbed for Semantic Portrayal (Fellah, 2017) presented previously.

In the approach of Huang and Harrie (2020) the definition of the graphical el-

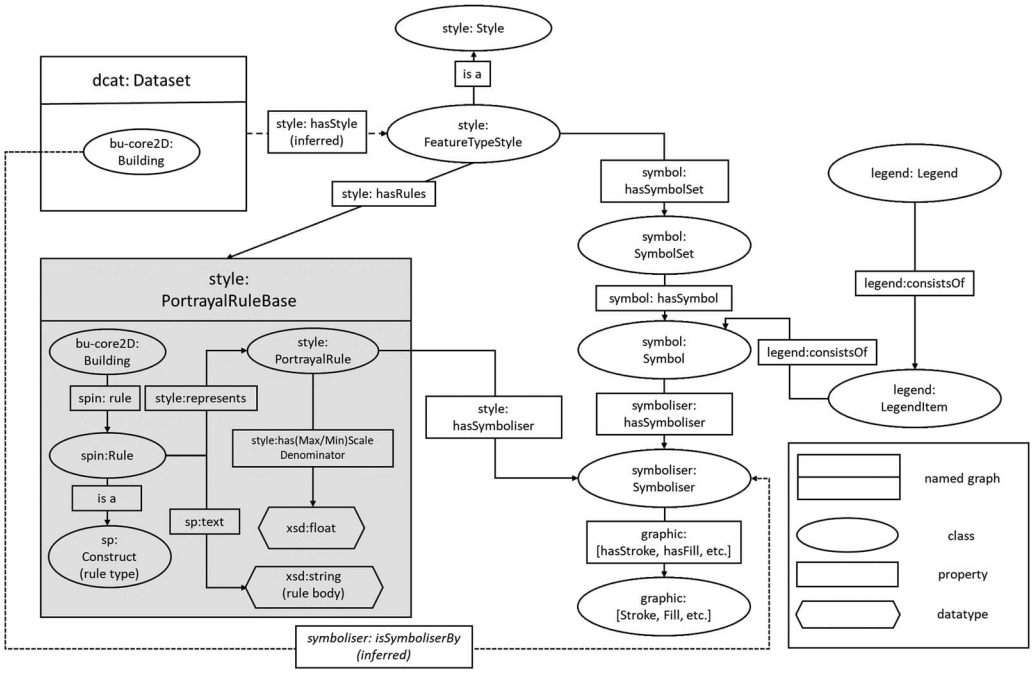

Figure 4.12: Ontologies for *data portrayal*. Source: Image from Huang and Harrie (2020).

ements is therefore carried out in RDF; then the semantic rules that will allow to link individuals and these graphical elements are written using SPIN. They thus allow conditional portrayal (i.e. filter the entities to which they apply) through these rules. Nevertheless, these rules are not generic in the sense that they must be written in a specific way depending on the geovisualisation to be created (i.e. adding new graphical elements does not only require the user of the approach to write them in RDF but also to write new SPIN rules so they can be used). In their case study, they visualise buildings coloured according to their age. One of the subtleties of their approach is also that the geometry of the entities to be represented is chosen according to the zoom level as well as according to the availability of geometries in the exposed datasets (some buildings do not have detailed geometry) as shown in Figure 4.13. This is solved by the SPIN rules that they call *geometry source* rules. One of the main limitations we see concretely (illustrating our previous point) is the lack of genericity of the approach as well as the quantity of SPIN rules (we can count twelve *portrayals rules* and two *geometry source rules* specific to their use-case[5]) to be written to obtain a choropleth portrayal for which there are well-defined codes (such as the mapping of continuous values to classes represented by a colour). We note, however, that their ontologies and rules are readily available online[6], which facilitates rapid iteration on this work, although the data is not published (for licensing reasons) nor the source code of the implementation.

[5]All their rules are visible at the following URL: https://github.com/RightBank/Knowledge-based-geovisualisation/tree/master/rules

[6]https://github.com/RightBank/Knowledge-based-geovisualisation

Figure 4.13: Knowledge-based geovisualisation of heritage building in central Stockholm. Source: Image from Huang and Harrie (2020).

In the case study presented by Huang et al. (2020), the authors use the portrayal vocabularies proposed by Huang and Harrie (2020) and introduced just above with a specific focus on the geovisualisation of urban bicycling suitability. In this work, the approach is still based on the SPIN inference mechanism but SHACL constraints are also added to validate the integration of the data to be exploited by the geovisualisation. We note that the authors also identify the lack of genericity of the Huang and Harrie (2020) approach and that they try to remedy it by using new concepts formalising high-level cartographic knowledge (Figure 4.14) and reusing part of the work of Brus et al. (2010). In particular, the authors formalise the types of data that can be mapped (using the *AttributeData* and *GraphicData* concepts and their hierarchies unchanged from Brus et al., 2010) and link these types to continuous colour progressions (they propose vocabulary elements for this purpose in their ontology). Depending on the number of classes required by the data encountered by their system, a rule interpolates the colour progression into the correct number of colours.

Despite this, we believe that the lack of genericity of the approach remains the main limitation: as with previous approach, it requires writing SPIN rules specific to the data to be exploited in order to link individuals and their symbolizers (especially as their quantitative data already has 4 pre-calculated categories). As before, the

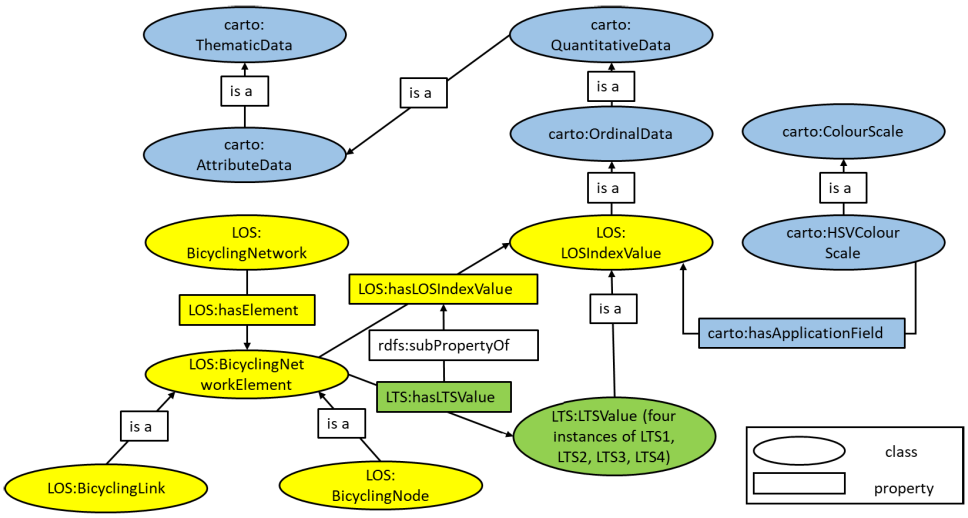

Figure 4.14: Core concepts and relations in the *cartographic ontology* of Huang et al. (2020) and their relations with the data to be depicted. Source: Image from Huang et al. (2020).

ontologies used are available online, and this time, together with the source code of the implementation[7].

4.3 Semantic Web models that could benefit from additional geovisual semantics

Works of the literature, including especially thesis recently defended within the STeamer team[8] use Semantic Web technologies to formalise fundamentally geospatial domains. Some of these works do not go as far as using the semantics of their models for the creation of the geovisualisation. Some others rely on an approach composed of many steps, leading to the duplication or to the loss of the semantics of the original model.

For example, the work of Saint-Marc (2017) aims to understand the spatial dynamics of flood management and its impacts on the railway network. This work led to the creation of an ontology, IDISFER, describing relationships between natural flood phenomena and incidents on the railway system. The author also proposes a graphic semiology to depict the different types of incidents on the rail network but there is no approach to formally link this symbology with the data described by IDISFER. Besides, the graphic semiology proposed by Saint-Marc (2017) is revived by Villanova-Oliver (2018) who draws the contours of an approach based on

[7] https://github.com/RightBank/Knowledge-based-integration-and-visualisation

[8] STeamer is a research group at LIG (Grenoble Computer Science Laboratory) in which this PhD has been conducted.

ontologies for organising at a semantic level the concepts exploited in a process of an automatised creation of a geovisualisation. She starts from the observation that the knowledge contained in an ontology (whether application, business, or domain) cannot be directly transposed into a geovisualisation interface and identifies the challenge of formalising the correspondence between a concept (from an application, business, or domain ontology) and the geovisual portrayal concepts that can be associated with it. In this proposal, this correspondence is formalised by means of a matching ontology, *GVR Matching*, which makes it possible to establish the link between the data and their symbolisation in the geovisualisation using the *GEOVIS* ontology (Figure 4.15). This proposal does not result in the formalisation of ontologies that can be directly exploited by a machine but is explained on paper with the help of diagrams and Description Logics. This approach, which sets some design choices that can be made, is an important building block of the approach advocated in this manuscript and is completely in line with the ideas of Fabrikant (2001).

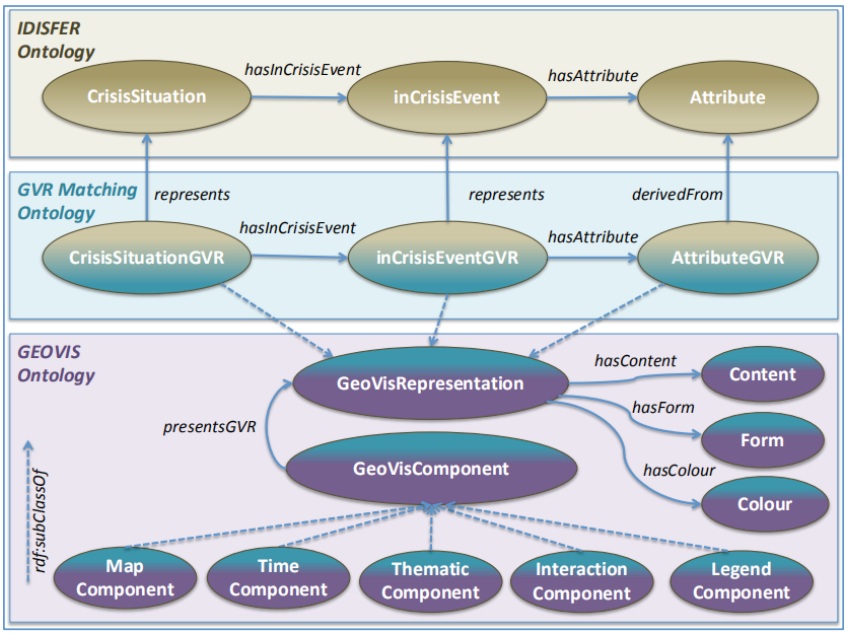

Figure 4.15: Diagram of the *GVR Matching* approach of Villanova-Oliver (2018). Source: Image from Villanova-Oliver (2018).

Bernard (2019) proposes ontologies allowing the description of administrative entities (the TSN-Ontology, for Territorial Statistical Nomenclature) and the changes they undergo over time (the TSN-Change Ontology). These ontologies aim at the constitution of RDF graphs containing the knowledge related to the changes that these administrative entities have experienced. The concepts defined in these ontologies are fundamentally spatio-temporal and some of them are quite graphical. Indeed, the TSN-Change ontology defines several types of *geometry restructurations* such as *fusion*, *scission* or *integration* to only mention a few. We believe that the RDF graphs constituted from this approach and immersed in the Web of data could be exploited through a geovisualisation description and generation approach that would directly

exploit the semantics of TSN-Change to convert it into a graphical semiology and allow its use in a geovisualisation application. However there is currently no approach to do this directly. The author herself points out the great potential of these ontologies for geovisualisation, but does not propose any tools to exploit them (the author even provides a proposal for a green-orange-red symbology as well as examples of corresponding queries on pp.176-179).

Other recently proposed models have both an obvious spatial component but no proposed geovisualisation interface to exploit them. This includes, for example, the ontology of reference objects (OOR) developed in the course of the CHOUCAS project (Olteanu-Raimond et al., 2020). It could be exploited so that its data can be portrayed to build WMS tiles for example. We keep these works in mind, ensuring that our proposal can also offer a solution to simply derive a geovisualisation from them.

4.4 Looking back at our research problem and overview of the contributions

We announced in Chapter 1 that our research problem was to facilitate or even to automate the creation of rich geovisualisation that exploit the semantics of a data model, in particular for the Semantic Web. We have learned, in the first two chapters of the state of the art, some interesting elements, respectively regarding the contributions of geovisualisation to effectively support spatial reasoning and regarding the contributions of Semantic Web technologies to formally represent knowledge. In this chapter we have seen that approaches already exist that have tried to capture the knowledge of the cartography domain as well as approaches to use Semantic Web technologies or consume Semantic Web data in order to create maps. Despite this, we believe that none of these approaches can answer our research problem, but that they constitute an interesting foundation on which to base our proposal. We thus discuss the interesting and reusable elements found in the literature as well as the ideas of open issues suggested in the literature that we believe could be solved by our proposal.

4.4.1 Lessons learned and foundations for our approach

All the works cited in this chapter supports the fact that it is possible to use declarative knowledge and make it usable for mapping and geovisualisation. We have seen, however, that the approaches of the different authors are of various nature and that some proposals are partially outdated by others. We have also seen that not all of them have the same level of development (from abstract proposals to fully implemented systems). Here we point out the main elements that we retain and on which we base our work (these elements are summarised in the Table 4.1 - only works whose elements we reuse appear in it).

Proposal	Why the approach does not fit	Elements that we reuse and how
Iosifescu-Enescu and Hurni (2007)	Dedicated to thematic cartography. Vocabularies only available on paper.	Some vocabulary elements (and their relationships). Conceptual notions on software design for rule-based symbolisation.
Smith (2010)	Dedicated to thematic cartography. Vocabularies only available on paper.	Some vocabulary element (and their relationships)
Brus et al. (2010)	Approach specific to cartography and not to geovisualisation.	Vocabulary elements related to cartographic representations and mapable data types.
León et al. (2012)	Approach rather intended to present data catalogue. Limitations in the methods of data selection and styles to be applied.	The idea of the generic approach allowing the consumption of RDF data and the declarative approach for the selection of the categories of data to be represented.
Ruzicka et al. (2013)	Approach specific to cartography and not to geovisualisation.	Advices on using OWL to represent mapping-specific knowledge.
Penaz et al. (2014)	Approach specific to cartography and not to geovisualisation.	Advices on using OWL to represent mapping-specific knowledge.
Fellah (2017)	Intended for the publication and mediation of information related to cartographic representations of OGC catalogue data.	The symbol, symbolizer and graphic vocabularies (although concretely Huang and Harrie, 2020 have done the work of writing it in RDF and we rely on this version)
Varanka and Usery (2018)	Not a declarative approach to visualise data from an existing model. No readily available implementation.	Conceptual notions on the formalisation of the map as a knowledge base
Villanova-Oliver (2018)	Not implemented.	The high-level vision of geovisualisation, in particular as being made up of multiple components. Some vocabulary elements (and their relationships).
Huang and Harrie (2020)	Lack of modelling at a high level. Need to write specific rules to link data and symboliser. Vocabularies available but not the full implementation.	The symbol, symbolizer and graphic vocabularies. Conceptual notions on software design for rule-based symbolisation.

Table 4.1: Summary of reusable elements of the state of the art on semantic models for geovisualisation.

The architecture of the cartographic library for rule-based symbolisation proposed by Iosifescu-Enescu and Hurni (2007) is a theoretical proposition. It still seems to be adapted to current practices and we recognise ourselves in it. Indeed, as they mention, we wish to propose several techniques for exploiting the knowledge base for the geovisualisation that we are going to create (WMS server, vector feature map, generation of documentation, etc.). They also express the desire to separate mapping vocabularies from the rules that use them. This is an interesting point and we also propose vocabularies that can be used independently of the rules that use them.

Ruzicka et al. (2013) also express their wish for independence from a specific system, but they mention technical in nature limitations they encountered. The technical choices we are doing are different and benefit from advances in Semantic Web technologies. This is why we are confident in the adoption of a declarative approach that is as independent as possible from a specific system.

Details on the graphical properties of visual variables made by Iosifescu-Enescu and Hurni (2007) are, in the end, reflected in Huang and Harrie (2020) proposal of portrayal ontologies. Indeed, we particularly value Huang and Harrie (2020) and Huang et al. (2020) approaches and notably their ontologies for data portrayal that are based on the work of Fellah (2017). These ontologies (namely *scale*, *symbol*, *symbolizer* and *graphic* ontologies) are updated and used in our proposal.

However, we believe that the link must be created between these ontologies for data portrayal and ontologies describing at a high level i) what a geovisualisation is, and ii) what its visualisation context is. The vocabulary and organisational proposals at a high level made by Iosifescu-Enescu and Hurni (2007) are partially taken up by Smith (2010) in his *Map Ontology*. In our opinion, they need to be adapted to our geovisualisation challenges and also to the new graphic vocabularies that we are going to use. Similarly, the proposal of Villanova-Oliver (2018) for a high-level organisation of the geovisualisation domain, called *GEOVIS Ontology*, is interesting but lacks some concepts and lacks operationalisation. We however reuse some of them such as *GeoVisComponent* and an adapted version of *GeoVisRepresentation*.

The proposal of León et al. (2012) is interesting because it already targets the visualisation of Semantic Web data and aims at some degree of genericity in the consumption of data, but it is limited in the way the data to be represented are selected and in the way these data can be represented. These elements are partially solved by Huang and Harrie (2020) and Huang et al. (2020), but they require the user of their approach to write SPIN rules themselves. We think this is an important pitfall and it is in line with the issue identified by Villanova-Oliver (2018) in relation to the formal linking of the concepts or individuals to be used geovisually and their graphic properties. We believe that it is possible to perform the task of describing the geovisualisation to be created and to create it in a completely declarative way by the user of our approach. In particular, we believe that rather than having a "specific" (in that it contains terms specific to the domain ontology being exploited) matching ontology such as Villanova-Oliver (2018), we can have a generic vocabulary, allowing

to describe how the graphical symbolisation concepts should be linked to the data and allowing to be expanded into an RDF graph containing these links. As we are interested in producing useful geovisualisations, this approach must embody some form of intelligence in that it allows the semantics of the data to be exploited or even transformed. We also think that this declarative approach allows for an even better implementation of the concept of *map as a knowledge base* stated by Varanka and Usery (2018).

4.4.2 Our contributions

Beyond exploiting Semantic Web data, we claim that *Semantic Web-based Geovisualisation* is a field that also offers an opportunity to make methods, techniques and tools of geovisualisation evolve so they fully exploit the possibilities offered by the Semantic Web technologies stack. To do this, we first propose to include the concepts of the geovisualisation field into the Semantic Web through new vocabularies to ensure their interoperability with the existing bricks. As such, by building on the proposals, whether architectural, conceptual or concrete, of most of the authors cited in this chapter (namely Iosifescu-Enescu and Hurni, 2007; Brus et al., 2010; Smith, 2010; Fellah, 2017; Varanka and Usery, 2018; Villanova-Oliver, 2018; Huang and Harrie, 2020; Huang et al., 2020) we propose a stack of compatible ontologies for geovisualisation we describe in the next chapter (Chapter 5). The ontologies created are a *geovisualisation ontology*, a *context ontology*, an *interaction ontology*, a *colour palette ontology* and an ontology of *cartographic solutions*. We also modify the *symbol*, *symbolizer* and *scale* ontologies and reuse the *graphic ontology* from Huang and Harrie (2020). Together these ontologies form a vocabularies ecosystem enabling the description of the main components of a geovisualisation. This constitutes a first contribution in terms of knowledge representation for the field of Semantic Web-based Geovisualisation. As a step further, in Chapter 6, we propose a model of RDF specification document allowing describing the geovisualisation to be created *a priori*, meaning before any implementation, and declaratively, thanks to the previous vocabularies we define. We afterward present a framework allowing transforming this specification into a well-shaped RDF graph ready to be exploited by a geovisualisation interface. This generic geovisualisation interface is also part of our contribution as a ready-to-use functional template.

Our proposals, which are detailed in the following chapters and implemented in several case studies, allow us to solve both problems mentioned in the literature previously (notably by Iosifescu-Enescu and Hurni, 2007; Varanka and Usery, 2018; Villanova-Oliver, 2018) as well as those we have asked ourselves in Chapter 1 regarding the elaboration of an intelligent geovisualisation.

Part II

Proposals for the geovisualisation designer

Chapter 5

An ontology ecosystem to describe geovisualisation

5.1 Introduction

Our readings, in particular those mentioned in Chapter 2, have enabled us to identify the conceptual elements that make up a useful geovisualisation interface for the user as well as the various types of graphical representations to be implemented in order to make semiotically correct depiction of phenomena and to support the reading of the map.

We propose in this chapter a formal description of the various elements necessary for the creation of a geovisualisation, as well as the relationships that link them, focusing our approach on its central element: the interactive map. This description is provided in the form of several ontological vocabularies using RDFS/OWL. From the most general to the most particular, they are as follow:

- *covikoa-geoviz* to describe the interface of a geovisualisation application (Section 5.2.1),
- *covikoa-interaction* to describe interactions that can happen in the application (Section 5.2.2),
- *covikoa-context* to describe the initial state of the application and the initial visualisation context (Section 5.2.3),
- *covikoa-derivation* to specify filters and rules for selecting spatial entities that correspond to a portrayal (Section 5.3.2),
- three vocabularies to describe the composition of graphic objects to be displayed on the map, the *symbol*, *symbolizer* and *graphic* ontologies that are taken from Fellah, 2017 and Huang and Harrie, 2020, and discussed in Chapter 4 (Section 5.3.1),

- *scale*, a vocabulary to describe the visualisation scale at which a portrayal is valid that is taken from Huang and Harrie, 2020, and previously discussed in Chapter 4 (Section 5.3.3).

Finally, and to return to high level concepts, we propose an instantiated vocabulary for describing discrete colour palettes, *dicopal* (Section 5.4.1). We then propose *carto* (Section 5.4.2), a vocabulary to describe common cartographic solutions and the data types that are adapted to them. The latter is compatible and can make use of palettes described in *dicopal*.

It is worth mentioning that these vocabularies are compatible with each other and can be used in a composable way, using only a subset of them or all depending on the need. Furthermore, they can be used independently of the system that is proposed in the next chapter. Indeed, in accordance with what we identified in Chapter 3 as favouring the reuse of ontologies for the Semantic Web, the vocabularies we propose here have only weak formal constraints (making them valid in RDFS-Plus and OWL 2 RL for example). In the same logic to facilitate the reuse of our ontologies in another context, we choose that the requirements for their use in the framework presented in the following chapter are specified by SHACL shapes *from the framework side*, keeping ontologies independent from it.

We also note that *covikoa-derivation* vocabulary is not only used to describe filters and rules for selecting entities but also various elements relating to the materialisation of data in order to enable the passage from a specification document to a well-formed graph mobilising all the covikoa ontologies: this is discussed in Chapter 6.

Since the new vocabularies introduced are numerous and in order to facilitate the reading, we provide below a table with their full name as well as the prefixes and namespaces introduced in this chapter (Table 5.1).

Name	Prefix	Namespace
Covikoa-geoviz	gviz	http://lig-tdcge.imag.fr/steamer/covikoa/geoviz#
Covikoa-interaction	ion	http://lig-tdcge.imag.fr/steamer/covikoa/interaction#
Covikoa-context	cvkc	http://lig-tdcge.imag.fr/steamer/covikoa/context#
Covikoa-derivation	cvkd	http://lig-tdcge.imag.fr/steamer/covikoa/derivation#
Symbol	symb	https://gis.lu.se/ont/data_portrayal/symbol#
Symbolizer	symblzr	https://gis.lu.se/ont/data_portrayal/symbolizer#
Graphic	graphic	https://gis.lu.se/ont/data_portrayal/graphic#
Scale	scale	https://gis.lu.se/ont/visualisation_scale#
Carto	carto	http://lig-tdcge.imag.fr/steamer/covikoa/carto#
Dicopal	dcpal	http://purl.org/dicopal#

Table 5.1: Prefixes and namespaces introduced in this chapter.

Finally, all the element (ontology, rules, code, data for case studies) which constitute the contribution of this thesis are available on a Git repository[1] and we sometimes refer to files of this repository by giving their URLs directly.

[1] https://github.com/mthh/CoViKoa

5.2 Geovisualisation application

5.2.1 Covikoa-geoviz vocabulary: a generic, extensible and high-level description of a geovisualisation interface

This vocabulary describes at a high-level the concepts used in the construction of a geovisualisation interface[2]. It allows to describe the various components of the interface and the way in which some RDF data should appear on these components. Figure 5.1[3] gives an overview of the concepts and their relations and we present through the text subsets of this figure for more clarity about the concepts that are introduced.

We first briefly introduce the notion of Semantic Data Model (SDM). This is the term we use to designate the RDFS/OWL ontology that describes the data that have to be geovisualised. It has to be understood as the data model of any application domain. For instance in the context of administrative divisions, concepts of 'country', 'region' and 'municipality' could be the data to geovisualise. Likewise, in a land-planning context it may define the 'road network' with a hierarchy of road types, the various categories of 'buildings' that may be encountered, etc. The two expectations for this model depicted in Figure 5.1 (middle left, delimited by a dashed border) are:

- that the different types of things to be represented are structured using the notion of class (whether RDFS or OWL),
- that it structures and qualifies geospatial information, describing some of its elements using the GeoSPARQL vocabulary (cf. Chapter 3).

We explain later the way this SDM is linked to the concepts of *covikoa-geoviz* we present now.

Geovisualisation application

`gviz:GeoVisualApplication` (Figure 5.1, bottom left) is the concept that ties together the various components of a specific geovisualisation interface. While it could have been implicit (for a given graph containing only the knowledge of a single interface for example), we have chosen to make this concept explicit. This can allow, depending on the organisation of the data chosen by a user of this ontology, to store the description of several geovisualisations in the same graph. Moreover, we believe that this is necessary to offer a complete approach, from the geovisualisation

[2]The *covikoa-geoviz* vocabulary can be found in its turtle serialisation at the url https://github.com/mthh/covikoa/blob/master/rdf/voc-geoviz.ttl.

[3]We use the visual notation from the VOWL specification (http://purl.org/vowl/spec/v2/) that was presented in Chapter 3 for all ontology images in this chapter with two notable differences: we use ellipses instead of circles to facilitate readability and we include the prefixes of the represented classes.

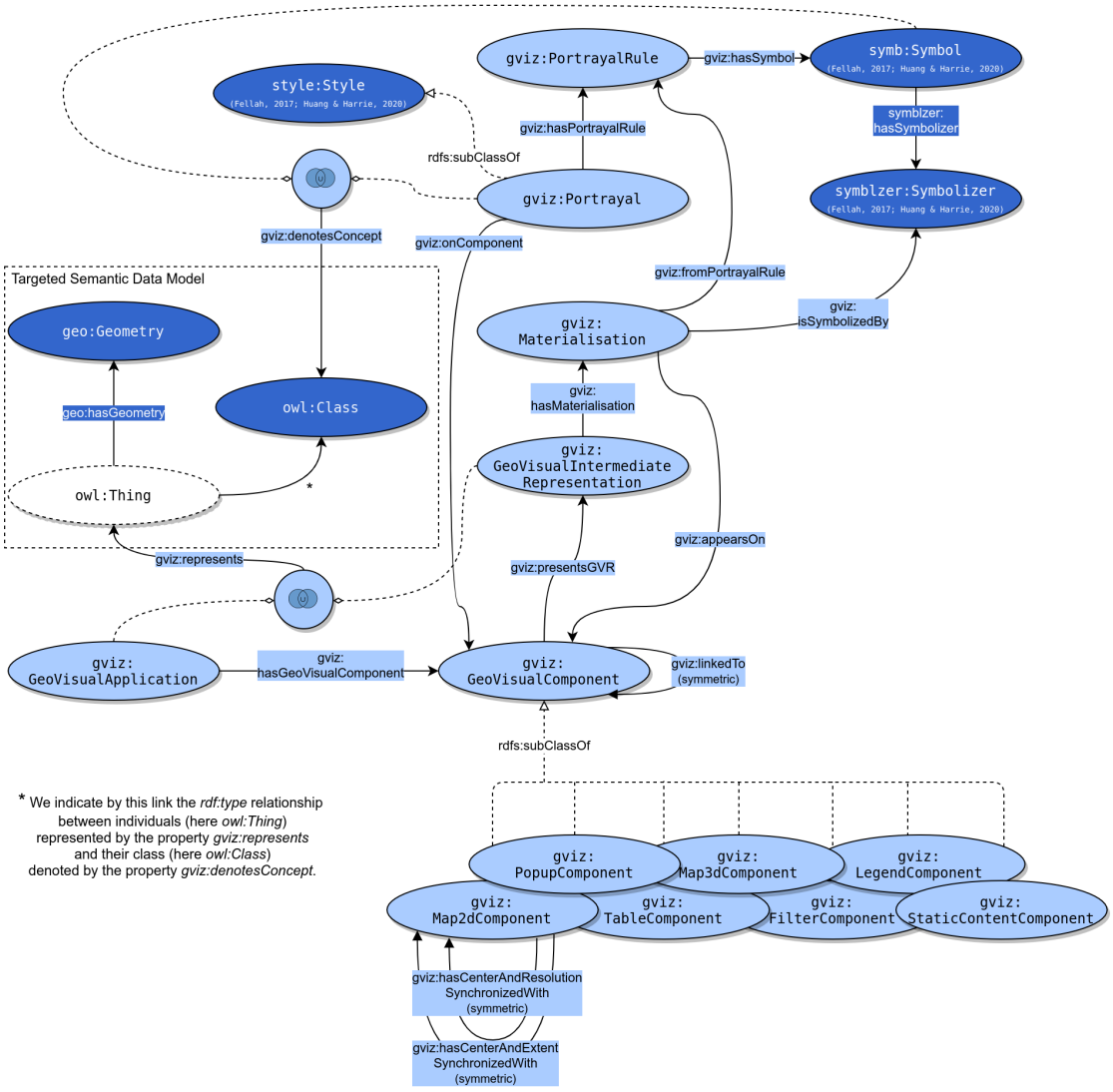

Figure 5.1: Overview of *covikoa-geoviz* ontology.

application to its components and the way in which individuals appear in it.

In order to add additional semantics, we encourage the user of this ontology to subclass `gviz:GeoVisualApplication` to describe its own type of application and its further instantiation.

Components for geovisualisation

We also define a class representing the components (`gviz:GeoVisualComponent`) of a geovisualisation application. In our approach a `gviz:GeoVisualApplication` is linked to one or more `gviz:GeoVisualComponent` (with the property `gviz:hasGeoVisualComponent`).

Formalising the notion of component makes it possible to describe complex geovisualisations such as the screenshots seen in Chapter 2: several maps possibly synchronised (Figure 2.5), terrain view (Figure 2.6), table, etc. The concept of `gviz:GeoVisualComponent` we propose is specialised by the following subclass (Figure 5.1, bottom):

- `gviz:Map2dComponent` (an interactive map),
- `gviz:Map3dComponent` (an interactive terrain view),
- `gviz:TableComponent` (a data table),
- `gviz:FilterComponent` (a component allowing to filter the entity displayed one or more other components),
- `gviz:StaticContentComponent` (a component allowing to display a static content specified in HTML),
- `gviz:LegendComponent` (a component displaying the legend of portrayals that appear on one or more map),
- `gviz:PopupComponent` (a component that displays a pop-up window at the location of a click, typically to display detailed information about an entity).

We distinguish ourselves here from the proposal of Huang and Harrie (2020) presented in Chapter 4 in which the interactive map is not formalised (it is thus considered as existing by default and being unique) and in which however the legend component is formalised. On the other hand, we are closer to the approach described by Villanova-Oliver (2018) in which the notion of component appears, including map and legend, or, to a lesser extent, to the ontological formalisation of Smith (2010) which describes the map element as having *at least one Map Body* (Figure 4.3), encompassing the fact that it may have several map bodies.

We believe that our components cover an interesting (and easily demonstrable in a generic implementation) part of geovisualisation interfaces; however, we are aware that this does not cover all the components that may be encountered. A user of this ontology could be led to define himself sub-classes that would be necessary (other components identified in Chapter 2 are typically diagrams or temporal components for example).

Furthermore, if the purpose of *covikoa-geoviz* is to describe the geovisualisation interface through its components and the data that appears in them, other elements related to the geovisualisation components could be formalised in order to take into account the various aspects of a geovisualisation interface (such as the presence of a basemap or the size and position of the components). We propose in Subsection 5.2.3 a vocabulary dedicated to the initial context of visualisation that covers these elements.

Any `gviz:GeoVisualComponent` is intended to have a content: this relies on the notion of `gviz:Materialisation` which is presented below.

Materialisation of individuals in components

Two central concepts of our approach are `gviz:GeoVisualIntermediateRepresentation` and `gviz:Materialisation` (Figure 5.1, middle). The approach we take here responds to the issues identified in Chapter 4 concerning the linking of the domain knowledge (the individuals to be represented) with the knowledge specific to geovisualisation and is inspired by Villanova-Oliver (2018), who proposed an intermediate representation in the form of a dedicated matching ontology which is intended to facilitate the provision of information in a suitable form for the implementation of the geovisualisation.

We define `gviz:GeoVisualIntermediateRepresentation` as the class representing new individuals created to represent, in the graph, individuals of the SDM to be materialised in a geovisual component. Each instance of this class is linked, by the property `gviz:represents`, to the individual of the SDM it represents. The individuals of type `gviz:GeoVisualIntermediateRepresentation` thus establish the link between domain knowledge and the knowledge related to geovisualisation. More precisely, they are representations of SDM individuals whose properties are exploited to later decide on the portrayals to apply according to rules presented in the next subsection. While the SDM is in a sense raw data, these IRs make it possible to have an elaborated version of the individuals, and these IRs can thus be the subject of properties aimed at qualifying the individuals to be represented for the sole purpose of geovisualisation. This concept thus offers the freedom to obtain variations on the SDM data without modifying them. As such, we note that in this approach no links are created from the SDM data but only to their destinations.

We then define `gviz:Materialisation` as the concrete materialisation, on a component, of an individual represented by a `gviz:GeoVisualIntermediateRepresentation` (with the property `gviz:hasMaterialisation`). In other words, this materialisation on a component means that the individual of the SDM appears on it (property `gviz:appearsOn`). This materialisation makes the links with the knowledge related to other geovisualisation concepts such as information necessary to visually render the individual on the component. Figure 5.1 (upper right) shows a `gviz:Materialisation` can be linked through the `gviz:isSymbolizedBy` property to a `symblzr:Symbolizer`, notably useful in

the case of a gviz:Map2dComponent or gviz:Map3dComponent.

In accordance with the definitions in the previous two paragraphs, there is one gviz:GeoVisualIntermediateRepresentation per individual appearing in the application and as many gviz:Materialisation per individual as it appears on the application's components. For example, a specific building that would appear twice on a map (by the polygon representing it and by its label visible at high zoom levels) and once in a table, would have a total of three materialisations.

As previously for gviz:GeoVisualApplication, we encourage (this is in fact the default approach in what we present in the following chapter) the user of this ontology to subclass gviz:GeoVisualIntermediateRepresentation in order to describe its own types of gviz:GeoVisualIntermediateRepresentation (for example for each SDM data class that is represented).

Portrayals and portrayal rules

We then define the concept of gviz:Portrayal (Figure 5.1, top middle) as the concept to describe a complete representation, through one or more rules (gviz:PortrayalRules) to describe how to link the data to the graphical element of its gviz:Materialisation. This class denotes one or more SDM concepts through gviz:denotesConcept property (we note that, as depicted thanks to the VOWL graphical notation for representing OWL unions, the symb:Symbol class can also be used as the subject of this property, this is discussed in Section 5.3.1). Finally, it is possible to qualify on which component a gviz:Portrayal appears (through the gviz:onComponent property).

We made the gviz:Portrayal class a subclass of style:Style which is defined as "(a) central concept (that) associates symbol sets with portrayal rule sets, which define the mapping of feature types to symbols" (Fellah, 2017) and is also reused in Huang and Harrie (2020). In Fellah (2017) proposal, style:FeatureTypeStyle, which is a familiar element to users of the SLD specification, is also a subclass of style:Style for example. Since our proposal is not intended to describe catalogues of styles but to describe the concrete portrayals that are applied to data and the links with this data (stored in a RDF graph), we create the concept of gviz:Portrayal which is closer, in our definition, to the approaches used in geovisualisation and cartography.

In order to allow for the expression of how individuals are depicted concretely, each gviz:Portrayal is linked to one or more gviz:PortrayalRules. A gviz:PortrayalRule concretely describes which individuals of the concept denoted by the portrayal are related to which symblzr:Symbolizer, possibly in compliance with the satisfaction of a cvkd:PropertyConstraint or a cvkd:SpatialConstraint (these elements belong to the *covikoa-derivation* vocabulary and are presented further in the text). Each gviz:Materialisation is thus also linked (by the gviz:fromPortrayalRule property) to a gviz:PortrayalRule. This explains more thinly the direct link (by the gviz:isSymbolizedBy

property) between `gviz:Materialsation` and a specific `symblzr:Symbolizer` we mentioned above.

In practice, it is this organisation that allows us to describe a cartographic representation that applies to entities that are not intended to be represented in the same way. For example, a choropleth representation (defined as a `gviz:Portrayal`) of a set of entities requires several `gviz:PortrayalRules`, one for each class of data (expressed in our case by a `cvkd:PropertyConstraint`) and linked to the appropriate `symblzr:Symbolizer` for the class of data. This example will be discussed in more detail in Section 5.3.2 and through Figure 5.4.

We also note that the concepts of `gviz:Portrayal` and `gviz:PortrayalRule` are deliberately generic over any `gviz:Component` in our approach. Their qualification by a symbol specialises them quite naturally into `gviz:PortrayalRule` which can be used on `gviz:Map2dComponent` and `gviz:Map3dComponent`. We discuss in the next chapters how to use these `gviz:PortrayalRule` for a `gviz:TableComponent` but they could also serve to describe the content of other components such as diagrams.

5.2.2 Covikoa-interaction vocabulary: describing interactions with geovisual components

In Chapter 2, we discussed the interactions usually encountered with the cartographic component of a geovisualisation. The findings of Roth (2012) (cf. Figure 2.4) allow us to postulate that it is possible to describe an interaction by formalising the following three elements:

- the user's intent (which comes down to the question: *what does the user want to do?*)
- the trigger for the interaction (*how the user can do it?*)
- the outcome of the interaction (*where and how does the interaction impact?*)

Our objective is to demonstrate the feasibility of taking into account interaction features in our approach. It appears challenging to formalise the whole field of interaction so we choose to focus on one specific user's intent only. Indeed, since we envision geovisualisation as a tool to support the construction of knowledge and decision making, we consider this is in line with an analytical purpose. We call this concept the `ion:AnalyticalPurpose` of an interaction.

Based on the work of Roth (2012), and in particular on the objective-based primitives that he identifies in the literature, we retain different kinds of analytical purposes: *identify*, *compare*, *filter* as well as *order / sort*. We believe that these four analytical purposes (the first two are cited by many authors such as Wehrend and Lewis, 1990; Maceachren et al., 1999; Andrienko et al., 2003) cover the part most often implemented in geovisualisation interfaces as well as the part that makes geovisualisation unique. We also think that they allow us to demonstrate our point

as stated by the implementation we propose for them in the next chapters.

Figure 5.2 gives an overview of the vocabulary we build[4]. The `ion:Interaction` class is central in this ontology and ties all the other elements together. An instance of this concept is linked to each `gviz:PortrayalRule` or each `gviz:Portrayal` whose materialisations (`gviz:Materialisation`) allow the interaction. This makes it possible to express that any materialisation associated with a `gviz:PortrayalRule`, or with the `gviz:PortrayalRules` of a `gviz:Portrayal`, can be the trigger or recipient of the interaction. This allows flexibility to declare an interaction only once when it applies to all materialisations of each portrayal rule of a portrayal (e.g. in the case of a choropleth representation that requires several portrayal rules). This flexibility is also granted when declaring the outcome of the interaction through the `ion:targetsEntitiesFrom` property that is linked to a `ion:SelectionStrategy` (see below).

It is then necessary to look at the element that triggers the interaction: we call this concept the `ion:Event`. The aim is to describe the various events that can be triggered by the user of the interface. We instantiate this concept with the individuals `ion:singleClick` (a single click on the materialisation of a spatial entity), `ion:mouseOver` (the action of hovering the cursor over the materialisation of a spatial entity), `ion:contextMenu` (although this action does not necessarily trigger the opening of a contextual menu, we use the JavaScript terminology for the click action corresponding to the context-menu, typically the right click for a right-hand desktop pointing device) and `ion:doubleClick` (a double click on the materialisation of a spatial entity).

The next element we have chosen for qualifying an interaction is its result: we call this concept the `ion:Outcome` of the interaction. This concept makes the most links with other knowledge. It is indeed possible to describe with this `ion:Outcome` (typically instantiated as a RDF blank node, cf. Listings 5.1 and 5.2):

- the `gviz:Component` on which the interaction has an effect through the property `ion:onComponent`,
- the possibility to use either a new `symblzr:Symbolizer` or a `ion:SymbolizerModifier` which is a mechanism allowing to point to a property to be modified during the interaction and to provide its new value (the semantics we associate with using a new `symblzr:Symbolizer` is that, during the interaction, a new graphical element is added to the map -it matters if it uses a fill colour with transparency for example- whereas using `ion:SymbolizerModifier` corresponds to modifying the graphical element already present),
- the possibility to define a selection strategy (`gviz:SelectionStrategy`), which allows to describe on which materialisations the interaction has an effect through two types of strategies:
  - `ion:FollowPropertyPath` which allows to describe that it is necessary to follow a property path, starting from the individual corresponding to the materialisation with which the interaction takes place to obtain one or

[4]The *covikoa-interaction* vocabulary can be found in its turtle serialisation at the url https://github.com/mthh/covikoa/blob/master/rdf/voc-interaction.ttl.

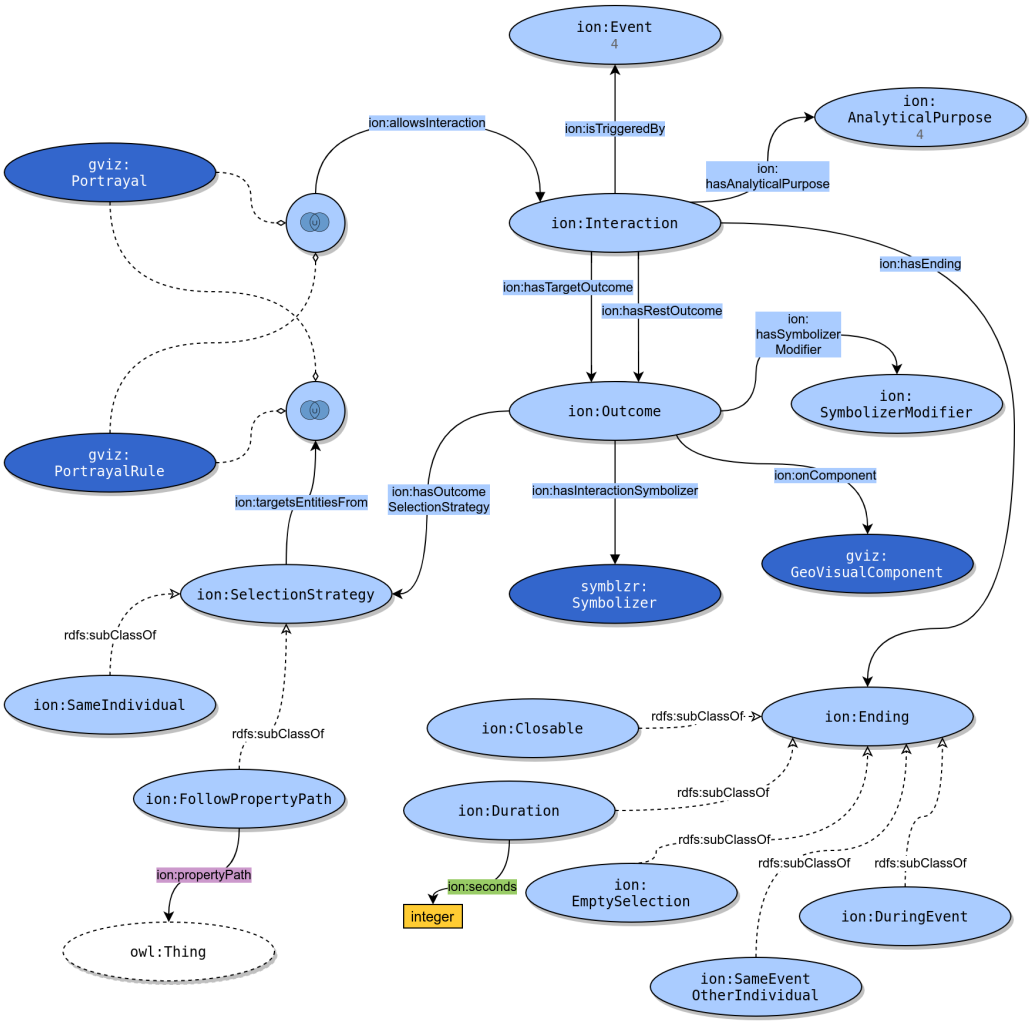

Figure 5.2: Overview of *covikoa-interaction* ontology.

more individuals: the interaction applies then to their materialisations on the `gviz:Portrayal` or the `gviz:PortrayalRule` linked by the property `ion:targetsEntitiesFrom`,
  - `ion:SameIndividual` which allows to describe that the interaction applies to the materialisation corresponding to the entity with which the interaction takes place but on the `gviz:Portrayal` or the `gviz:PortrayalRule` linked by the property `ion:targetsEntitiesFrom`
- the possibility of reversing the effect of the selection strategy (which was explained just above if property `ion:hasTargetOutcome` was used) by using property `ion:hasRestOutcome`, which means that it is not the materialisations pointed out by the selection strategy but all the other materialisations of the `gviz:Portrayal` or of the `gviz:PortrayalRule` linked by the property `ion:targetsEntitiesFrom` that are candidate to receive the outcome of the interaction.

All the properties with `ion:Outcome` as their domain are not intended to be used at the same time: we assume that an `ion:Interaction` has a global effect on a component (property `ion:onComponent`) or an effect on the materialisations issued by a `gviz:PortrayalRule` or by a `gviz:Portrayal` through the definition a `ion:SelectionStrategy`. In doing so, we recognise the diversity of possible outcomes for interactions, while believing that the combination of `ion:AnalyticalPurpose` type and `ion:Outcome` information allows for the full description of various classic geovisualisation interactions.

In addition, we have identified that not all interaction outcomes end for the same reason. Indeed, some may give rise to a transient effect of a defined duration (temporary highlighting for example), others may end as soon as the event that triggers it ends (this is sometimes the case for the display of information on cursor hovering), others may be explicitly closeable (for example for an information popup obtained following a click) or may be closed following an event taking place outside the entities that can receive the interaction. This is why we propose the class `ion:Ending` which is linked to `ion:Interaction` and expresses the reason for the termination of its outcome(s). For now, we define five subclasses of this concept corresponding to the examples we have just mentioned. We have chosen not to model them as instances but as subclasses to allow the user to instantiate them and to qualify them with possible properties (such as the `ion:seconds` property in the case of a `ion:Duration` instance). We also believe that it offers the possibility of describing selections of groups of entities (sometimes called *brushing* in geovisualisation) in the context of selection interactions that would only end when an empty selection (`ion:EmptySelection`) is made.

All these concepts make it possible to define a coherent and extensible vocabulary for describing interactions (Figure 5.2). Indeed, we believe that it is extensible because of two aspects:

- new `ion:AnalyticalPurpose` can be defined (as long as they are documented and fit into our model),
- the `ion:Outcome` concept can support the addition of new properties, which would be for example specific to components (like `symblzr:Symbolizer` which is specific to `gviz:Map2dComponent` and `gviz:Map3dComponent` among the components we de-

fine).

We illustrate this modelling with the instantiation of two classical interactions. For example, our model allows us to describe an interaction, triggered by a simple click (`ion:singleClick`) on an entity of the map, in order to display the detailed information of the entity on a component of type `gviz:PopupComponent` (Listing 5.1). In this case we qualify the interaction with the analytical purpose `ion:identify` and the outcome with the property `ion:onComponent` which points to an instance of `gviz:PopupComponent`. All this information contains all the semantics necessary to trigger the display of a popup containing all the attribute information of the clicked entity (as it is common in geovisualisation) and this is due to the well-known geovisualisation meaning of the *identify* purpose coupled with the fact that the outcome occurs on popup component.

```
:identifyToPopup
    a ion:Interaction ;
    ion:isTriggeredBy ion:singleClick ;
    ion:hasAnalyticalPurpose ion:identify ;
    ion:hasEnding [ a ion:Closable ] ;
    ion:hasTargetOutcome [ a ion:Outcome ;
        ion:onComponent :nutsIdentifyPopup ; # a gviz:PopupComponent
      ] .
```

Listing 5.1: Example of instantiation of `ion:Interaction` (1).

The second example (Listing 5.2) is that of an interaction that would, in an application with a map and a data table, highlight on the map the entity clicked on in the table (this linking technique that propagates the information about the selected data to other views or components is described by various authors in the literature such as Dykes et al., 2007; Roth, 2012). An outcome is defined for the entity targeted by the interaction (through the `ion:hasTargetOutcome` property and the `gviz:Portrayal` which is targeted) and an outcome is defined for a set of other entities (through the `ion:hasRestOutcome` property and the `gviz:Portrayal` which is targeted). This makes it possible to apply both a specific symbology to the targeted entity (i.e. to its correspondence in another component) and a different symbology to all the other entities (i.e. to their correspondence in another component). Such an interaction result can be seen as an implementation of the highlighting techniques proposed by Robinson (2011) and mentioned in Chapter 2. Finally, it is possible to specify, optionally, a duration during which the outcome of the interaction takes effect.

```
:highligthOnMap
    a ion:Interaction ;
    ion:hasAnalyticalPurpose ion:identify ;
    ion:isTriggeredBy ion:singleClick ;
    ion:hasEnding [ a ion:Duration ;
        ion:seconds 10 ;
      ] ;
    ion:hasTargetOutcome [ a ion:Outcome ;
        ion:hasInteractionSymbolizer  [ a symblzr:PolygonSymbolizer ;
            graphic:hasStroke [ a graphic:Stroke ;
                graphic:strokeColor "rgba(99,99,99,1)"^^graphic:cssColorLiteral ;
              ] ;
```

```
            graphic:hasFill [ a graphic:Fill ;
                graphic:fillColor "rgba(255,255,0,1)"^^graphic:cssColorLiteral ;
              ] ;
          ] ;
        ion:hasOutcomeSelectionStrategy [
            a ion:SameIndividual ;
            ion:targetsEntitiesFrom :somePortrayal ;
          ] ;
      ] ;
    ion:hasRestOutcome [ a ion:Outcome ;
        ion:hasInteractionSymbolizer  [ a symblzr:PolygonSymbolizer ;
            graphic:hasStroke [ a graphic:Stroke ;
                graphic:strokeColor "rgba(99,99,99,1)"^^graphic:cssColorLiteral ;
              ] ;
            graphic:hasFill [ a graphic:Fill ;
                graphic:fillColor "rgba(210,210,210,0.6)"^^graphic:cssColorLiteral ;
              ] ;
          ] ;
        ion:hasOutcomeSelectionStrategy [
            a ion:SameIndividual ;
            ion:targetsEntitiesFrom :somePortrayal ;
          ] ;
      ] .
```

Listing 5.2: Example of instanciation of `ion:Interaction` (2).

Finally, the notion of interaction as presented here describes interactions between component entities or between components. It does not, however, allow the description of interactions on (or with) a component, independently of its data. These interactions with a component, which make it possible to change the state of the component but not necessarily to interact with its data, are frequent in cartography: for example, the actions of *zooming*, *panning* and *rotating*. We explain in the following subsection how we approach this topic through the initial visualisation context.

5.2.3 Covikoa-context vocabulary: describing the initial visualisation context

Since the purpose of our ontologies is to describe a geovisualisation interface, it is useful to have a vocabulary to describe the state of its various components through the initial visualisation context.

The *covikoa-context* vocabulary has this vocation[5]. It is built around the concept of application (`cvkc:Application` which is made equivalent to `gviz:GeoVisualApplication` with the property `owl:equivalentClass`) and components (`cvkc:Component`, which is made equivalent to `gviz:GeoVisualComponent`) and their respective sub-classes[6]. We have chosen to redefine these component classes in order to allow the reuse of this

[5]The *covikoa-context* vocabulary can be found in its turtle serialisation at the url https://github.com/mthh/covikoa/blob/master/rdf/voc-context.ttl.

[6]Class representation with a double border and the IRI of the equivalent class(es) under the main IRI is what allows to depict equivalent classes in VOWL.

vocabulary independently of *covikoa-geoviz*. The names of the components are also slightly different (Figure 5.3), which is explained by the fact that this vocabulary is clearly oriented towards *the implementation of the interface* of the geovisualisation application. As shown in Figure 5.3, an equivalence relationship has also been established between all the subclasses of `cvkc:Component` and `gviz:GeoVisualComponent`.

This vocabulary is structured mainly around a few utilitarian concepts as well as numerous datatype properties that allow to concretely describe additional aspects of a geovisualisation through the qualification of its components. At the components level, this vocabulary makes it possible to describe, for each of them, their desired size and position in the interface as well as the possibility to close them, to resize them or to move them which are possibilities frequently encountered in dashboard-like geovisualisations (none of these datatype properties allowing to qualify the components are depicted in Figure 5.3 for the sake of readability).

Concerning more specifically the map and terrain components (here `cvkc:Map2d` and `cvkc:Map3d`), *covikoa-context* thus makes it possible to express the presence or absence of a background map (such as those presented in Chapter 2, Figure 2.3). This is an element that we consider essential to formalise in order to describe the cartographic components of a geovisualisation interface. While present in the proposal of Iosifescu-Enescu and Hurni (2007) (which formalises the *ReferenceMap* and *ThematicMap* concepts), this element was absent from the proposals of Varanka and Usery (2018) and Huang and Harrie (2020) when they respectively describe interactive maps and geovisualisations. We propose several individuals of type `cvkc:BaseMap` containing both the reference url to use them and the attributions to put on the map. It is also possible to describe the webmapping library used (`cvkc:MappingLibrary`) by the component, the desired presence of a graticule as well as its initial map extent.

In addition, we said in the previous subsection that some interactions are inherent to the cartographic components of a geovisualisation (zoom, pan and sometimes rotation). These interactions, implemented by default by webmapping libraries, can sometimes be voluntarily deactivated in a geovisualisation component (for example to obtain a map with a fixed extent, either as the main map because the scale requires it, or used as a reference map, like a carton in paper maps). The disabling of these interactions is thus explicit in our model, as it is in the webmapping libraries, and we define the concept of `cvkc:DeactivableMappingInteraction` (for which we instantiate three individuals corresponding to the three interactions mentioned above). A last element that we define in relation to `cvkc:Map2d` is a constrained scale range (`cvkc:hasConstrainedScaleRange`), meaning that zooming is possible only between the min- and max-values defined by the `scale:Scale` that is linked to it.

Furthermore, it is possible to describe, at the level of the application itself (via the `cvkc:VisualisationContext`) various elements specific to the user's context (`cvkc:UserContext`) such as: the fact that he or she has a colour vision deficiency, the device that he or she is using, and elements specific to his or her interface preferences (`cvkc:InterfacePreferences`), such as the choice of a "light" or "dark" theme (these datatype properties are not shown in Figure 5.3).

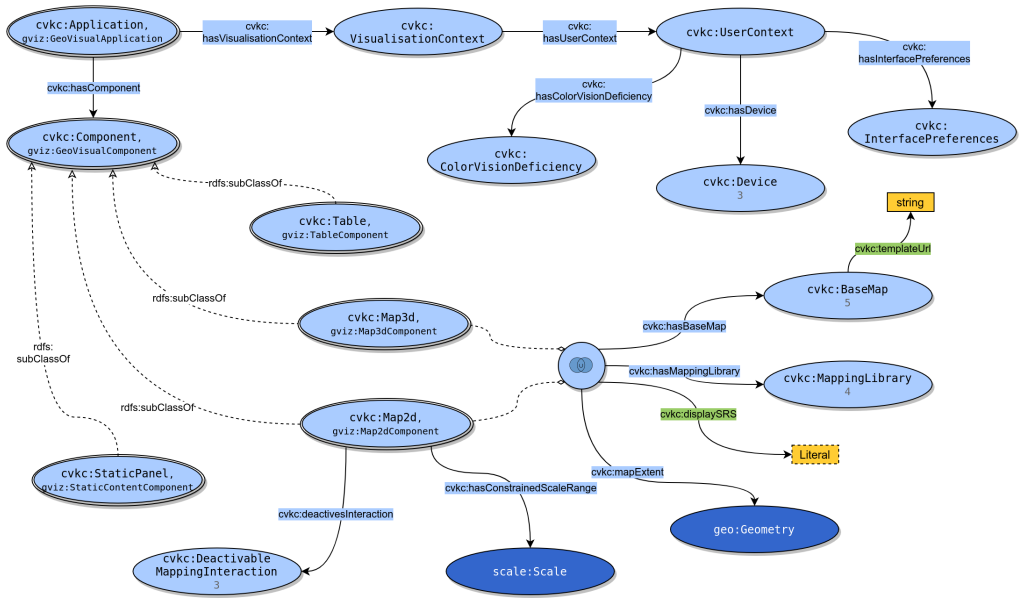

Figure 5.3: Overview of *covikoa-context* ontology.

Although it captures classic elements of the implementation of the user interface of a geovisualisation, we recognise that *covikoa-context* is intimately related to both the *covikoa-geoviz* vocabulary and the purpose of the framework presented in the next chapter (and we discuss at this point how the information described by *covikoa-context* can be used in practice).

5.3 Description of the symbols involved in portrayal rules

5.3.1 Graphical properties of symbolisers

We reuse here three vocabularies described by Fellah (2017) and made available in RDF by Huang and Harrie (2020): the *Symbol ontology*, the *Symbolizer ontology* and the *Graphic ontology* that have been presented in Chapter 4. However, we apply to them slight modifications which are described below.

Concerning *Symbol ontology*[7], we no longer use the `symb:denotes` property to express that the symbol denotes a specific concept in order not to distort the original semantics given by the authors. Instead, we now use the `gviz:denotesConcept` property whose domain is both `gviz:Portrayal` and `symb:Symbol`. Indeed we think that offering this flexibility is relevant depending on the type of representation to be implemented and on how the data can be organised in RDF. A classical example to justify this

[7]The *Symbol ontology* can be found in its turtle serialisation at the url https://github.com/mthh/covikoa/blob/master/rdf/voc-huang-symbol.ttl.

choice can be that of a choropleth representation in which the `gviz:Portrayal` denotes a particular concept but the different symbols that compose it (of different colours according to the data class of the entity to be represented) do not denote any particular concept. Conversely, one can imagine a symbolisation by pictograms in which each `symb:Symbol` denotes a specific concept of the SDM (this is the case in the example given by Fellah, 2017). Otherwise, we do not modify the classes it contains nor its link to the *Symbolizer ontology*.

In *Symbolizer ontology*[8] we define a new annotation property which is used on the classes `symblzr:PointSymbolizer`, `symblzr:LineSymbolizer` and `symblzr:PolygonSymbolizer` to specify that the default geometry types which correspond to them are respectively (Multi-)Point, (Multi-)LineString and (Multi-)Polygon. We use the *Simple Features Geometry ontology* which is part of the GeoSPARQL proposal (presented in Chapter 3). This annotation is used later, when creating the link between representable entities and the corresponding symbolizer(s). Formalising this knowledge is comparable to the rules expressed by authors such as Brus et al. (2010) whose example of operationalisation of their *Cartographic Ontology* (presented in Chapter 4) is the rule "when there exists PointFeature then insert PointSymbol" and will be used in the following chapters for this purpose (in particular in Chapter 7). Finally we create a `symblzr:TemplateSymbolizer` class (and its respective subclasses `symblzr:TemplatePointSymbolizer`, `symblzr:TemplateLineSymbolizer`, etc.). This makes it possible to describe (using vocabulary elements present in *covikoa-derivation*) that some graphical properties of symbolizers (expressed with terms from *Graphic ontology*) do not have a fixed value but a value which depends on the entity to which the symbolizer applies. Classic examples include the creation of `symblzr:TextSymbolizer` whose textual value depends on the entity (allowing labels to be created) or the creation of `symblzr:PointSymbolizer` whose size varies proportionally to the value of a variable.

The *Graphic ontology*[9] remain unchanged and the declaration of a `symblzr:PointSymbolizer` qualified by elements of *Graphic* takes the following form (Listing 5.3).

```
:examplePointSymbolizer a symblzr:PointSymbolizer ;
    graphic:graphicSymbol [ a graphic:GraphicSymbol ;
        graphic:hasMark [ a graphic:Mark ;
            graphic:hasWellKnownName graphic:circle ;
            graphic:hasStroke [ a graphic:Stroke ;
                graphic:strokeColor "rgba(20,128,0,1)"^^graphic:cssColorLiteral ;
              ] ;
            graphic:hasFill [ a graphic:Fill ;
                graphic:fillColor "rgba(127,255,0,1)"^^graphic:cssColorLiteral ;
              ] ;
          ] ;
        graphic:size "5"^^xsd:decimal ;
      ] .
```

Listing 5.3: Example of a `symblzr:PointSymbolizer`.

[8] The *Symbolizer ontology* can be found in its turtle serialisation at the url https://github.com/mthh/covikoa/blob/master/rdf/voc-huang-symbolizer.ttl.

[9] The *Graphic ontology* can be found in its turtle serialisation at the url https://github.com/mthh/covikoa/blob/master/rdf/voc-huang-symbolizer.ttl.

5.3.2 Portrayal of a phenomenon or a concept by a coherent set of rules

We have seen that it is possible to define portrayals denoting a particular SDM concept and that these portrayals are qualified by portrayal rules that link to the appropriate symbol. We propose to use another vocabulary, *covikoa-derivation*[10], to describe the filters and constraints that give portrayal rules all their meaning by enabling to express to which entities they apply. While some of the elements of *covikoa-derivation* are linked to the framework presented in the following chapter, some of them are sufficient in themselves to add interesting knowledge to the description of the portrayals of a geovisualisation.

Property constraints

Thanks to *covikoa-derivation*, it is possible to describe, via constraints on properties, which entities correspond to which portrayal rule. Here we take up and modify vocabulary elements that are well known to SLD users, such as the filter mechanisms, and in particular the comparison operators[11].

We thus define a concept that we call `cvkd:PropertyConstraint` (Listing 5.4) and that can be qualified by a property indicating the property path (through `cvkd:propertyPath`) to reach the value or the object that is the subject of the constraint (we call this value or object the *target node* in what follows) and by the following comparison properties:

- `cvkd:valueIsLessThan` (to designate individuals whose target node value is less than the value given to this property),
- `cvkd:valueIsLessThanOrEqualTo` (to designate individuals whose target node value is less than or equal to the value given to this property),
- `cvkd:valueIsGreaterThan` (to designate individuals whose target node value is greater than the value given to this property),
- `cvkd:valueIsGreaterThanOrEqualTo` (to designate individuals whose target node value is greater than or equal to the value given to this property)
- `cvkd:valueOrObjectIsEqualTo` (to designate individuals whose target node value or object is equal to the value or to the object given to this property),
- `cvkd:valueOrObjectIsNotEqualTo` (to designate individuals whose target node value or object is not equal to the value or to the object given to this property),
- `cvkd:objectOfType` (to designate individuals whose target node object is of the type given as object to this property),
- `cvkd:objectNotOfType` (to designate individuals whose target node object is not of the type given as object to this property).

[10] The *covikoa-derivation* vocabulary can be found in its turtle serialisation at the url https://github.com/mthh/covikoa/blob/master/rdf/voc-derivation.ttl.

[11] These SLD mechanisms are for example described through their implementation in geoserver: https://docs.geoserver.org/stable/en/user/styling/sld/reference/filters.html.

It can be seen that some of these comparison operators are specific to RDF data: it is indeed possible to compare literal values (for *ordering* and *equality*) but also RDF resources of any type (for *equality*) as well as testing whether the reachable object is of a specific RDF type (by using `cvkd:objectOfType`) or not (`cvkd:objectNotOfType`). This makes it possible to deal with a wide range of cases that may emerge from the various data organisations possible in RDF.

```
:examplePropertyConstraint a cvkd:PropertyConstraint ;
    cvkd:propertyPath ex:country ;
    cvkd:valueOrObjectIsEqualTo "FR" .
```

Listing 5.4: Example of a `cvkd:PropertyConstraint` that can be used to qualify a `gviz:PortrayalRule`.

A classic example of a property constraint is to represent entities in different colours according to the value of a variable qualifying this entity. We take the example of NUTS2 entities, a statistical territorial unit of the European Union, here referred to as the class `ex:Nuts2Unit`, that we want to represent by their coloured surface according to the unemployment rate (`ex:unemploymentRate`)[12]. We define values corresponding to the limits of the classes used to group the data into sets that make sense in terms of the distribution of values. We define as many `gviz:PortrayalRule` as there are data groups, and we associate the appropriate `symblzr:Symbolizer` with each of these groups. The set of these portrayal rules (including in our example a portrayal rule using a special value designating the absence of the property on some individuals) gives rise to a complete `gviz:Portrayal` (Figure 5.4).

We note, on the bottom-right of Figure 5.4, the use of the special value `cvkd :absentProperty` (this is an individual of type `cvkd:SpecialMatchingValue`) which is a way of indicating that the portrayal rule applies to entities that are not qualified by the property path specified in the property constraint, allowing in our case to have a special class "No Data", indicating entities belonging to the study space but for which data is not available. Our vocabulary also defines `cvkd:presentProperty`, another individual of type `cvkd:SpecialMatchingValue`, which can be used to indicate the opposite: the fact that this property path is used, without any particular indication of the object or value linked by this property path.

Spatial constraints

The *covikoa-derivation* vocabulary also allows the definition of spatial constraints (`cvkd:SpatialConstraint`) that can be linked to portrayal rules, in a similar way to the property constraints seen on Figure 5.4.

These filters make it possible to express that only the entities passing the test of a spatial predicate are used by the portrayal rule in question. They are based on the use of one of GeoSPARQL's spatial predicates and on the use of a constant

[12] This model and some data instantiating it are provided in Appendix A, Listing A.1.

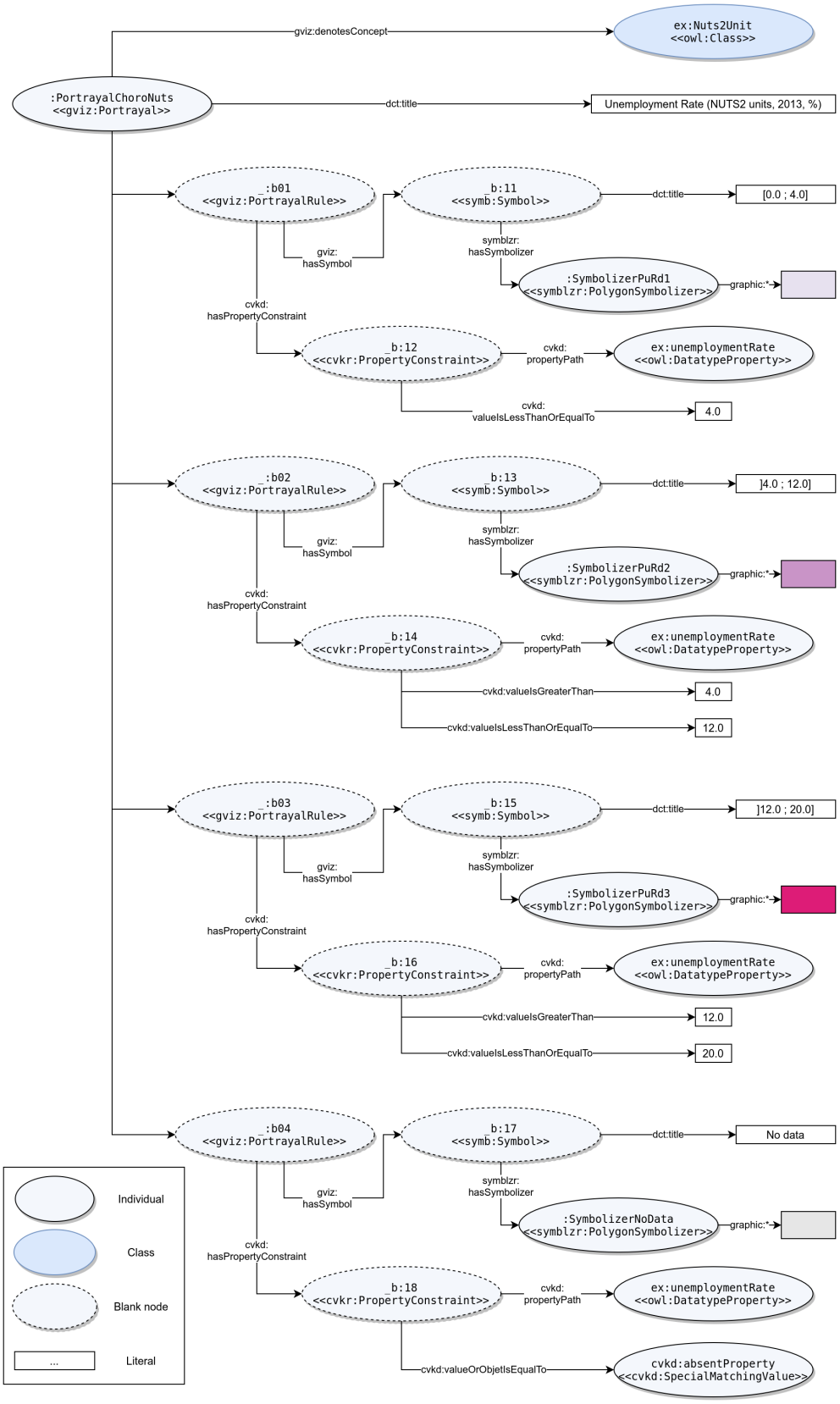

Figure 5.4: Portrayal rules allowing to describe a simple choropleth map.

value describing a geometry in its literal WKT form. Thus the example of Listing 5.5 allows to express that only the entities for which the spatial predicate *intersects* (line 2, property `cvkd:spatialPredicate`) is true with the provided geometry (line 3, property `cvkd:value`, here a point in its literal WKT form) must be used by the portrayal rule which is linked to this constraint.

```
1 :exampleSpatialConstraint a cvkd:SpatialConstraint ;
2     cvkd:spatialPredicate geof:sfIntersects ;
3     cvkd:value "POINT(1 1)"^^geo:wktLiteral .
```

Listing 5.5: Example of a `cvkd:SpatialConstraint` that can be used to qualify a `gviz:PortrayalRule`.

Logical articulation between constraints

In order to allow the expression of more complex property constraints and spatial constraints on a single portrayal rule, we also define the two logical operators *or* (property `cvkd:or`) and *and* (property `cvkd:and`). They are both intended to be used in the same way, they can be used on a subject being a property constraint (`cvkd:PropertyConstraint`) or a spatial constraint (`cvkd:SpatialConstraint`) and take as object an RDF list containing respectively property constraints or spatial constraints.

Listing 5.6 shows an example of an and-type logical articulation between two spatial constraints. It expresses that only the entities that satisfy both constraints are selected to be represented by the portrayal rule.

```
1  :examplePortrayalRule a gviz:PortrayalRule ;
2      cvkd:hasSpatialConstraint [
3          cvkd:and (
4              [ a cvkd:SpatialConstraint ;
5                  cvkd:spatialPredicate geof:sfIntersects ;
6                  cvkd:value "POINT(1 1)"^^geo:wktLiteral ;
7              ]
8              [ a cvkd:SpatialConstraint ;
9                  cvkd:spatialPredicate geof:sfTouches ;
10                 cvkd:value "POINT(2 2)"^^geo:wktLiteral ;
11             ]
12         ) ;
13     ] ;
14     gviz:hasSymbol [ ... ] .
```

Listing 5.6: Example of logical articulation between two `cvkd:SpatialConstraints` in a single `gviz:PortrayalRule`.

In the case of using the *or* operator, this expresses that entities that satisfy at least one of the constraints of the list are selected to be represented by the portrayal rule. We also note that these logical articulations can be nested to express constraints of the type (A *and* B) *or* C.

Other filters, which only make sense in the approach presented in the next chapter, are presented in due course as well as methods that do not use constant values

but values calculated from the data to describe transformation operations and spatial constraints.

5.3.3 Valid visualisation scale for the portrayal rules

We reuse here the terms from the *Scale* vocabulary proposed by Huang and Harrie (2020) and inspired by Carral et al. (2013) of which we adapt several elements (Figure 5.5)[13].

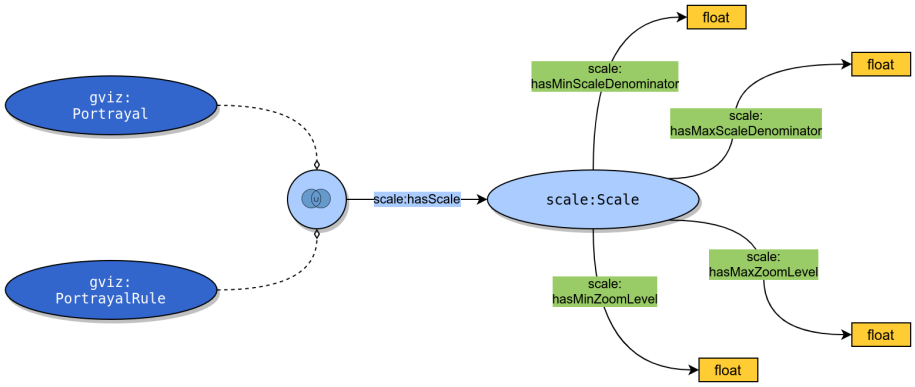

Figure 5.5: Overview of *Scale* ontology.

First of all, this vocabulary is originally intended to qualify a scale (`scale:Scale`) by its minimum and maximum scale denominator (respectively with the properties `scale:hasMinScaleDenominator` and `scale:hasMaxScaleDenominator`). We retain these features but also offer the possibility to define the range of validity of a scale by its minimum and maximum zoom levels (respectively with the properties `scale:hasMinZoomLevel` and `scale:hasMaxZoomLevel`). This choice is explained by the fact that our vocabularies are oriented towards Web technologies, in which the web-mapping frameworks frequently refer to this value. We believe that using these properties allows for rapid prototyping and that non-cartographer users might be equally comfortable with these values that they might have encountered elsewhere (as when browsing OpenStreetMap for example).

We also adapt the validity domain of the `scale:hasScale` property compared to Huang and Harrie (2020) since in our case we want to express this validity at the level of the `gviz:PortrayalRules` which describes how to link the individuals and their symbolizers or at the level of `gviz:Portrayals` that gather multiple `gviz:PortrayalRules` (the motivation is the same as when we were using an union between these two classes in *covikoa-interaction*). These changes are due to the difference in spectrum between the propositions from which the *Scale* ontology is developed and our ontologies as well as the system that accompanies them.

[13]The *Scale* vocabulary can be found in its turtle serialisation at the url https://github.com/mthh/covikoa/blob/master/rdf/voc-huang-scale.ttl.

5.4 Adding more cartographic semantics to the created geovisualisation

5.4.1 In between symbol composition and cartographic practices: a colour palette ontology

An ontology of discrete colour palettes is also proposed. Its main purpose is to describe palettes according to the number of steps they contain and to the fact that they can be correctly perceived by people with a colour-vision deficiency (subsequently called by the common but more reductive term of colour blindness). We called it the *dicopal* ontology (where *dicopal* means *discrete colour palette*) and published it online at the URL `http://purl.org/dicopal`.

The implementation was carried out by organising the concepts (Figure 5.6) of *Palette* and *Scheme* (either *sequential*, *divergent* or *qualitative*) and properties that may qualify them such as the provenance of a *Palette* and the number of steps it contains. We have chosen to use the `rdf:Seq` container to hold the sequence of colours that materialise a palette. The `dcpal:Palette` class thus has `rdf:Seq` as its superclass. This choice is motivated both by simplicity reasons while remaining an efficient choice (Daga et al., 2019). Each `dcpal:Palette` is thus linked to a set of members that represent the colours that compose it. These colours are each described by a literal of type `graphic:cssColorLiteral`.

We also create the `dcpal:Scheme` class along with its three sub-classes, `dcpal:SequentialScheme`, `dcpal:DivergingScheme` and `dcpal:CategoricalScheme`. What we are expressing in this way is twofold. On the one hand we formalise the fact that colour palettes are adapted to a specific use: either to represent a sequential progression of values, or to represent a double divergent progression, or to represent distinct categories. On the other hand, we formalise the fact that authors generally propose palette types (such as "a blue coloured progression", "pastel colours for categories", etc.), which are themselves implemented as concrete palettes containing a finite number of colours (the "blue coloured progression" in question may have been proposed by an author only in its 5- and 6-step, in particular to ensure the accuracy of their perception).

Finally, we describe the provenance of each of the implemented schemes by referencing scientific publications as instances of `prov:Entity` and we link each scheme to this entity with the property `dcpal:isFrom` that we made a sub-property of `prov:wasDerivedFrom` (not depicted on Figure 5.6).

The result is an ontology populated with 294 instances of `dcpal:Palette`, coming from the work of Okabe and Ito (2002), Brewer et al. (2003), Light and Bartlein (2004), and Crameri (2018) and distributed according to whether they fall into a sequential, divergent or qualitative scheme.

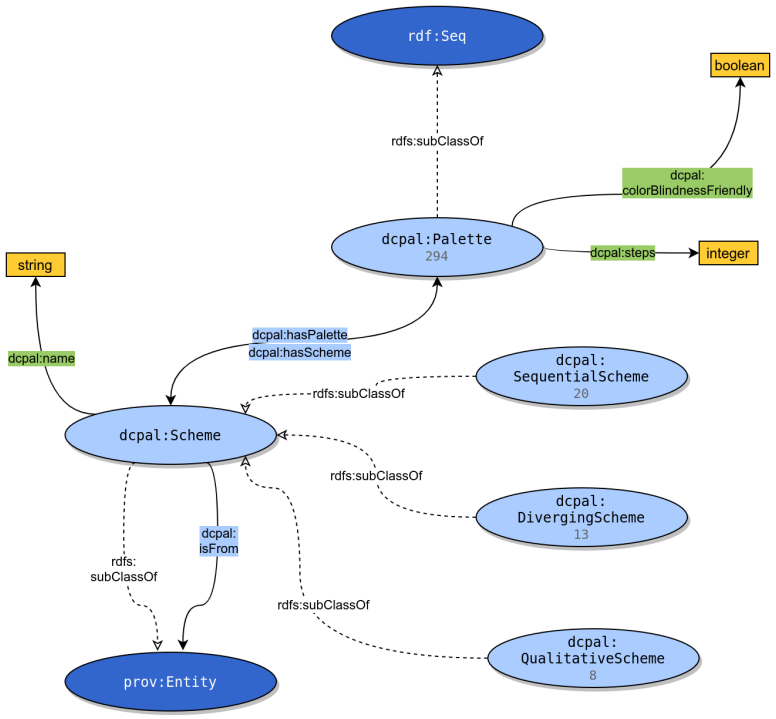

Figure 5.6: Overview of *dicopal* ontology.

Our ontology reflects a choice of orientation that is clearly different from Huang et al. (2020), who had chosen to propose interpolating values on continuous colour progressions. We believe that our choice to use discrete palettes, which also required more work upstream to prepare *dicopal* numerous palettes, has several advantages: it corresponds to a common practice in cartography, it allows for the encoding of divergent and categorical schemes in addition to the sequential scheme, and ensures that the palettes remain uniformly perceptible when so designed.

Regarding its production, the ontology was created by using RDFLib Python library and by consuming some palettes descriptions from the Palettable[14] library in order to transform them to fit our *dicopal* model. However, this library is not the only source used to integrate the palettes: those of Okabe and Ito, 2002 are not included in Palettable, for example, and the information concerning the colour blindness friendliness of the Brewer et al., 2003 palettes is not included in it either. Finally, all the variations of the palettes proposed by Crameri (2018) are not included because we have chosen to limit the number of data classes to 12 in the palettes we reference. Indeed, our palettes are intended for cartography, a discipline in which we believe that the stage of classification into a controlled number of data classes is essential.

The description of a palette, here the *BuPu* scheme from ColorBrewer (Brewer

[14] https://jiffyclub.github.io/palettable/

et al., 2003), which is a sequential scheme, and its implementation in a palette of 8 steps, alongside with the entity that references the scientific publication from which it originates, takes the following form in RDF (Listing 5.7 - our ontology also contains all other implementations of the BuPu scheme proposed by Brewer et al., 2003, from 3 to 9 steps, but we only present one for the sake of simplicity). The construction of these palettes by the authors who want them to be perceptually uniform implies creating each variation of the palette (for example here from 3 to 9). It is not possible to implement only the 9-step version and to take only 3 of its steps (for example the first step of the 3-step BuPu does not appear in any of the other BuPu definitions).

```
:BuPu a :SequentialScheme ;
    :isFrom <urn:doi:10.1559/152304003100010929> ;
    :name "BuPu" .

<urn:doi:10.1559/152304003100010929> a prov:Entity ;
    rdfs:label "Brewer, C. A., Hatchard, G. W., & Harrower, M. A. (2003).
    ColorBrewer in Print: A Catalog of Color Schemes for Maps. Cartography and
    Geographic Information Science, 30(1), 5-32. https://doi.org
    /10.1559/152304003100010929"@en ;
    dct:identifier "10.1559/152304003100010929" ;
    dct:title "ColorBrewer in Print: A Catalog of Color Schemes for Maps"@en ;
    bibo:doi "10.1559/152304003100010929" .

dcpal:BuPu_8 a owl:NamedIndividual , dcpal:Palette ;
    dcpal:hasScheme :BuPu ;
    dcpal:colorBlindnessFriendly "true"^^xsd:boolean ;
    dcpal:steps 8 ;
    rdf:_1 "rgb(247,252,253)"^^graphic:cssColorLiteral ;
    rdf:_2 "rgb(224,236,244)"^^graphic:cssColorLiteral ;
    rdf:_3 "rgb(191,211,230)"^^graphic:cssColorLiteral ;
    rdf:_4 "rgb(158,188,218)"^^graphic:cssColorLiteral ;
    rdf:_5 "rgb(140,150,198)"^^graphic:cssColorLiteral ;
    rdf:_6 "rgb(140,107,177)"^^graphic:cssColorLiteral ;
    rdf:_7 "rgb(136,65,157)"^^graphic:cssColorLiteral ;
    rdf:_8 "rgb(110,1,107)"^^graphic:cssColorLiteral .
```

Listing 5.7: BuPu scheme and its implementation in a palette with 8 steps.

It is thus possible to interact with the graph constituted by this ontology in SPARQL, for example to find all the divergent palettes with 8 steps (Listing 5.8) or to obtain the table which corresponds to the colours which compose a palette (Listing 5.9).

```
SELECT ?pal
WHERE {
    ?pal a dcpal:Palette ;
        dcpal:hasScheme/rdf:type dcpal:DivergingScheme ;
        dcpal:steps 8 .
}
```

Listing 5.8: SPARQL query to retrieve the IRI of each palette using a "diverging scheme" with 8 data classes.

```
SELECT ?element ?index
WHERE {
    ?pal a dcpal:Palette ;
```

```
        dcpal:hasScheme dcpal:BrBG ;
        dcpal:steps 8 .

    ?pal rdfs:member ?element .
    ?pal ?pt ?element .
    BIND(xsd:integer(SUBSTR(str(?pt), 45)) AS ?index) .
}
ORDER BY ?index
```

Listing 5.9: SPARQL query to retrieves the colours (in order) for the scheme 'BrBG' given 8 data classes.

5.4.2 An ontology of common map portrayals and corresponding data types

We have seen in Chapter 4 that several authors have proposed cartographic ontologies (Iosifescu-Enescu and Hurni, 2007; Smith, 2010; Brus et al., 2010; Ruzicka et al., 2013; Penaz et al., 2014). Some of them are explicitly concerned with the types of representations commonly encountered in cartography (Iosifescu-Enescu and Hurni, 2007; Brus et al., 2010; Penaz et al., 2014) and with the different types of data that can be mapped (Brus et al., 2010).

In order to have vocabulary elements specific to cartography from which we can derive meaning when exploiting a RDF graph, we also propose to formalise some of these elements in a vocabulary referred to as *carto* in what follows[15]. We believe that these new vocabulary elements allow us to improve the description of a geovisualisation alongside with the vocabularies previously presented.

We therefore formalise, in the manner of Brus et al. (2010) (Figure 4.4), the different types of data (`carto:Data`, cf. Figure 5.7) that can be encountered from three angles: its data-type with the class `carto:DataTypeDomain` (*is the information stored as text, numbers, etc.?*), the statistical type of variable with the class `carto:AttributeData` (*is the variable qualitative or quantitative, and of which kind?*) and that of the type of geometry of the entity with the class `carto:GraphicData` (*is the entity to be represented a Point, a Line, etc.?*).

Like Brus et al. (2010) and Huang et al. (2020) we identify two types of statistical data, qualitative and quantitative (respectively `carto:QualitativeAttributeData` and `carto:QuantitativeAttributeData`). However, unlike Brus et al. (2010) and Huang et al. (2020), we identify two sub-types within each of these categories, which we call `carto:NominalQualitativeData`, `carto:OrdinalQualitativeData`, `carto:RelativeQuantitativeData` and `carto:AbsoluteQuantitativeData`. This categorisation corresponds in particular to that proposed by Lambert and Zanin (2020) in their reference book on cartography.

We also formalise the fact that there are different categories of cartographic

[15]The *carto* vocabulary can be found in its turtle serialisation at the url https://github.com/mthh/covikoa/blob/master/rdf/voc-carto.ttl.

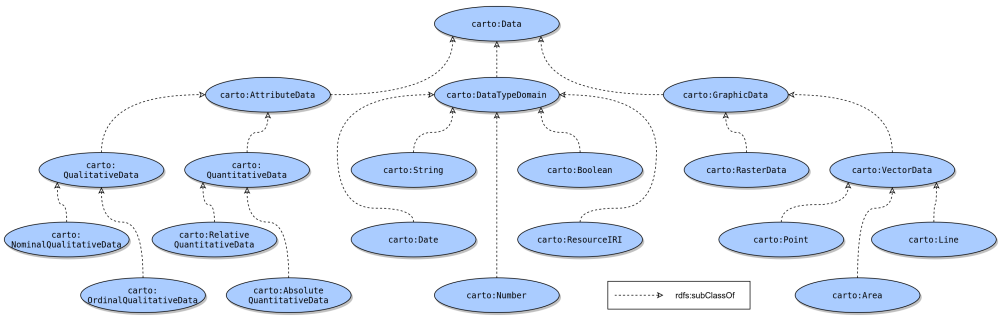

Figure 5.7: Overview of the hierarchy to qualify the data in *carto* ontology.

solutions (`carto:CartographicSolution`) through its two main subclasses that are `carto:QuantitativeCartographicSolution` and `carto:QualitativeCartographicSolution`. In contrast to Iosifescu-Enescu and Hurni (2007), Brus et al. (2010), and Penaz et al. (2014) we do not aim at exhaustiveness of cartographic methods and we choose to formalise only a restricted subset of them for now (Figure 5.8). This is mainly due to the fact that we want to ensure that the proposed solutions are compatible with the rest of the vocabularies and with the whole approach. In addition and in order to qualify the validity of the cartographic solutions we formalise with regard to existing data, each of them is linked, by annotation properties, to the data type that corresponds to it (through the properties `carto:forDataType`, `carto:forAttributeData` and `carto:forGraphicData`) For ease of reading we show only the linkage through the annotation property `carto:forAttributeData` in Figure 5.8. The use of an annotation property is not the only way in which these classes could have been related: it would also have been possible to treat these classes as individuals, a possibility offered by OWL2 and referred to as *punning*. While these annotations do not actually contain semantics in RDFS/OWL, they can be exploited later by a semantic rules mechanism.

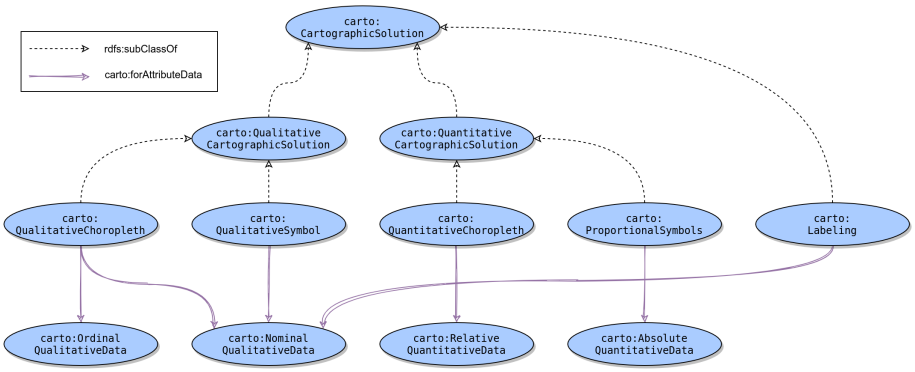

Figure 5.8: Overview of the hierarchy to qualify the cartographic solutions and their links with types of attribute data in *carto* ontology.

We note that some of the various ontologies for cartography that we have presented (Iosifescu-Enescu and Hurni, 2007; Brus et al., 2010) also endeavour to for-

malise the existing types of symbols (*Point*, *Line*, etc.) as well as their graphic parameters (*Fill*, *Stroke*, etc.). In our proposal, we do not need to formalise these elements because we consider that they are already covered by the *Symbol*, *Symbolizer* and *Graphic* ontologies, and this with a higher level of detail than in these various cartographic ontologies.

Finally, properties (not shown in our figures, but some of which are visible in Listing 5.10) were identified to qualify each of these mapping solutions and a `carto:PatronSymbolizer` class (a subclass of `symblzr:TemplateSymbolizer` presented previously) was created to describe the common parts of the symbolizers to be created as well as the parts subject to variation according to the value of the property used by the representation.

This vocabulary, used on its own or used in conjunction with *dicopal*, makes it possible to describe cartographic representations in a form (Listing 5.10) that can be used by a computer program (libraries such as *mapsf*, mentioned in Chapter 2, could use these parameters as input).

```
:choroplethUnempNuts a carto:QuantitativeChoropleth ;
    carto:targetsSpatialFeature ex:Nuts2Unit ;
    carto:targetsProperty ex:unemploymentRate ;
    carto:ofData [ a carto:Area , carto:RelativeQuantitativeData , carto:Number ] ;
    carto:numberDataClasses 3 ;
    carto:rendersNoData "true"^^xsd:boolean ;
    carto:hasClassificationMethod carto:Jenks ;
    carto:hasPalette dcpal:PuRd ;
    carto:hasPatronSymbolizer :patronSymbolizerChoroArea .
```

Listing 5.10: Description of a choropleth representation

When used in conjunction with the other vocabularies presented in this chapter, it can be used to add more cartographic semantics to the `gviz:Portrayal` that are created. It is thus possible to express that a `gviz:Portrayal` also implements a specific type of cartographic method (Listing 5.11). In addition, the properties of *carto* can be added incrementally (Listing 5.12) to add further semantics.

```
:PortrayalChoroUnempNuts a gviz:Portrayal , carto:QuantitativeChoropleth ;
    [... continuation of the usual description of the gviz:Portrayal ...] .
```

Listing 5.11: Incremental integration of *carto* with *covikoa-geoviz* (1)

```
:PortrayalChoroUnempNuts a gviz:Portrayal , carto:QuantitativeChoropleth ;
    carto:targetsProperty ex:unemploymentRate ;
    carto:hasPalette dcpal:PuRd ;
    [... continuation of the usual description of the gviz:Portrayal ...] .
```

Listing 5.12: Incremental integration of *carto* with *covikoa-geoviz* (2)

Actually, we can see that the description provided in Listing 5.10 corresponds to the instantiation shown in Figure 5.4 and could be used to add semantics regarding the cartographic choices made when creating the `gviz:Portrayal` and its `gviz:PortrayalRules`.

Of course, this adds work for the user and there is no guarantee that this additional information is coherent with regard to the `symblzr:Symbolizer` which is actually linked to the `gviz:Materialisation` of each individual. The next chapter presents an operationalisation of the information that can be described with *carto* vocabulary as well as its transformation into a `gviz:Portrayal` qualified by the all the necessary `gviz:PortrayalRule`s.

5.5 Conclusion

In this chapter we have presented several ontological vocabularies. Six of them are original creations (Table 5.2) and four of them are reused and have been slightly adapted from ontological vocabularies recently proposed in the literature.

	Knowledge domain	Classes	Individuals	Properties
Covikoa-geoviz	Geovisualisation interface	14	0	15
Covikoa-interaction	Interactions	14	8	13
Covikoa-context	Initial context of the geovisualisation application	23	12	24
Covikoa-derivation	Constraints, filters and transformation operation	17	2	30
Carto	Thematic cartography	25	4	14
Dicopal	Discrete colour palette	5	335	6

Table 5.2: A summary of the vocabularies presented in this chapter.

The joint use of all these vocabularies allows describing a geovisualisation interface, its components, its interactions and its initial state. One of the limitations is that the approach is very verbose: the description of a geovisualisation interface using these vocabularies can be tedious and requires the creation of many triples. For example, it is necessary to create all the `gviz:Materialisation`s and `gviz:GeoVisualIntermediateRepresentation`s and the same applies to the `gviz:isSymbolisedBy` link which must also be created from each `gviz:Materialisation`. Another limitation, related to the manual creation of these triples, is that if the semantics brought by the `gviz:PortrayalRule`s is interesting in itself, there is for the moment no guarantee (other than the confidence brought in the user) that the materialisations are linked to the appropriate `symblzr:Symbolizer` in the respect of the specified `cvkd:PropertyConstraint`s or `cvkd:SpatialConstraint`s.

However, we believe that it is possible to avoid creating `gviz:Materialisation`s and `gviz:GeoVisualIntermediateRepresentation`s (and all links to and from these individuals) manually, but to opt for a purely descriptive formalism that could create the necessary triples and be transformed into a complete graph according to our vocabularies.

The following chapter presents CoViKoa, a computer code framework allowing

to use all these ontologies to create, from a specification document, an RDF graph describing the geovisualisation interface and then, concretely, to generate the corresponding geovisualisation as a Web application. The complex constructions relating to the link between each individual and its materialisation on a component, which could have been cumbersome to define manually in RDF, is carried out by our framework on the basis of the specification document provided. It is this descriptive approach and its instrumentation by our framework that gives meaning and power to the ontology ecosystem we propose here.

Chapter 6

CoViKoa, a framework of semantic rules for geovisualisation generation

6.1 Introduction

In this chapter we propose a framework that enables to expand a specification document describing a geovisualisation into a well-formed RDF graph using the vocabularies presented in Chapter 5. This framework also allows to consume this RDF graph to create the corresponding geovisualisation in the form of a Web application.

This framework responds to the lack of an existing solution, identified by Villanova-Oliver (2018), to easily and directly build geovisualisation that exploit data described by an RDFS/OWL ontology and having geospatial features, which is itself part of a wider context, which is the lack of a presentation model for Semantic Web data (Pietriga et al., 2006). This framework eases the design of geovisualisations and helps creating executable geovisualisations without typing procedural computer code. It is aimed at a knowledge engineer who would like to create a geovisualisation for such data that he or she already has in RDF. The profile of a user having some skills in Semantic Web representation, is mainly the one that we refer to when we mention the *user* or the *geovisualisation designer* in the following of this chapter.

Since this framework aims at easing the design of geovisualisations and at allowing their generation, we are particularly attentive to the design choices made during the conception of this system. We recognise ourselves in two patterns identified by Heer and Agrawala (2006) when they review the design patterns for information visualisation:

- the *Reference Model* pattern, aiming at "*separate data and visual models to enable multiple visualizations of a data source, separate visual models from dis-*

plays to enable multiple views of a visualization (...)" (Heer and Agrawala, 2006),

- the *Production Rule* pattern which "*use a chain of if-then-else rules to dynamically determine visual properties using rule-based assignment or delegation*" (Heer and Agrawala, 2006).

We therefore opt for a declarative approach consistent with these statements. We propose the use of a specification document specific to geovisual knowledge (*separation of the data and the visual model*) which makes it possible to constitute an RDF graph dedicated to geovisualisation and which can be exploited by different client applications (*enabling multiple views of a visualization*). In addition, the link between individuals and geovisual knowledge is established through a rule-based mechanism (*dynamic determination of visual properties using rule-based assignment*).

Finally, our proposal can lighten the workload of a user wishing to create a geovisualisation from RDF data. If we refer to the design process of geovisualisations formalised by Robinson et al. (2005) and presented in Chapter 2 (Figure 2.17), our proposal intervenes at several of the identified stages. Indeed, by its declarative aspect, our approach emphasises the conceptual analysis and formalisation of the geovisualisation to be created (which components to define? what to display? etc. - which refers to the *conceptual development* stage) and thus to prototype a geovisualisation interface with dedicated vocabularies (*prototyping* stage). Our approach also makes it possible to obtain the geovisualisation described in the form of a Web application, saving time on the implementation and guaranteeing the conformity of the interface with the conceptualisation previously carried out (*implementation* stage).

It is from this descriptive approach, initiated as soon as the conceptualisation of a geovisualisation (cf. Figure 2.17), that our framework takes its name, CoViKoa, which is an approximate acronym of the French, albeit incorrect, sentence "*comment on visualise quoi*" which translates to "how do we visualise what". Such an intention is not new since it is in line with the work of Pietriga et al. (2006) who observed that the data presentation process was organised around two high-level problems, specifying *what* information to present, and specifying *how* to present it.

6.2 Overview of the framework characteristics

The use of CoViKoa therefore supposes that a data model in RDFS/OWL describes a geospatial domain. As seen in Chapter 5, this model deals with the data to be geovisualised and we call it the Semantic Data Model (SDM). The use of CoViKoa then requires the writing, by the user, of a specification document, called the Derivation Model in the following. The Derivation Model allows the expression of which individuals from the SDM are to be represented in the geovisualisation and of which components and interactions the geovisualisation is composed of. A rich set of elements can be described in the Derivation Model (constraints, transformation of geometries, selection of entities to be represented according to advanced criteria, etc.).

It also makes use of the *covikoa-derivation* vocabulary that we started to introduce in the previous chapter.

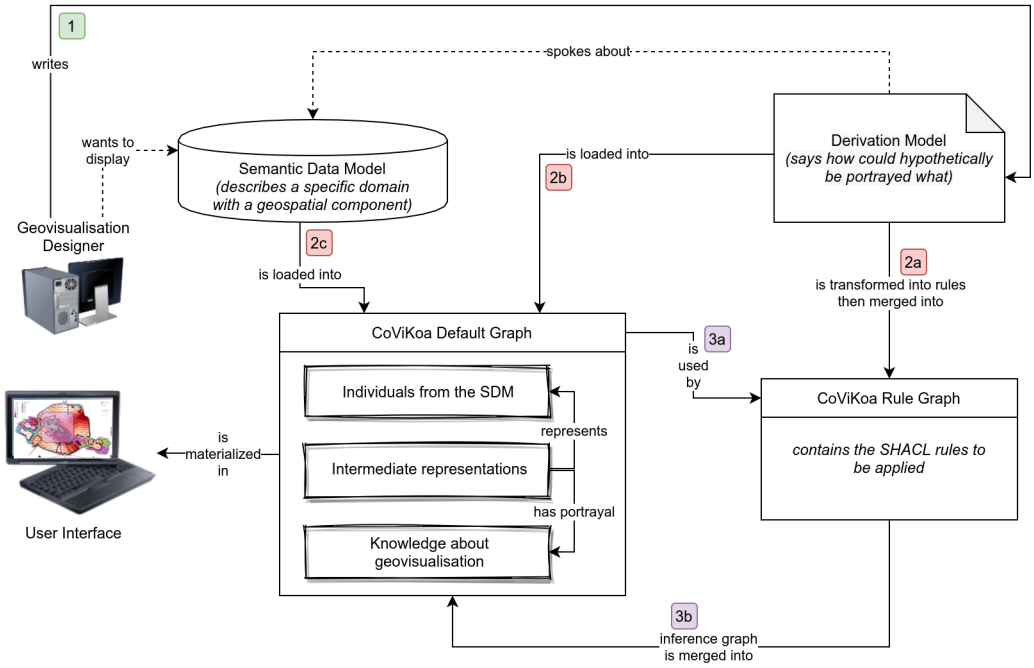

Figure 6.1: Overview of the framework characteristics.

The Figure 6.1 presents the workflow that leads to the creation of a geovisualisation with CoViKoa. A geovisualisation designer, the *user* of our framework, wants to create a geovisualisation of the data related to some concepts of the SDM so he or she writes the Derivation Model (1) that spokes about the aforementioned SDM and explains what the geovisualisation is to be composed of and how to represent the individuals of the SDM. The Derivation Model is then transformed into SHACL rules that are merged in the CoViKoa Rule Graph (2a) while the Derivation Model is also loaded (2b) in CoViKoa Default Graph alongside with the SDM (2c). The CoViKoa Rule Graph can now be applied to the CoViKoa Default Graph (3) in order to realise the necessary links to publish a graph that can be used to create a geovisualisation. The knowledge stored in the graph is materialised geovisually in a user interface accessible from a Web browser.

In Figure 6.1, we again encounter the concept of Intermediate Representation (IR), which was mentioned during the presentation of *covikoa-geoviz*. We thus express that the knowledge linked to geovisualisation is not encoded directly at the level of individuals of the SDM but that our approach uses special individuals that can be qualified as ontological Intermediate Representations that represent each of the individuals to be used in the geovisualisation. These IRs are topologically (from a graph point of view) located between the individuals to be represented in the SDM and the geovisual knowledge that is inferred by CoViKoa. Finally the joint mobilization of these intermediate individuals and the inferred geovisual knowledge

provides all the elements to build a geovisualisation.

Figure 6.2 presents a simplified and sequential view of the events presented in Figure 6.1. It allows us to introduce the 3 main stages of the framework's operation which are detailed in the following subsections:

1. Specification: the user writes a Derivation Model. He or she can benefit from the feedback of its validation by SHACL constraints we have defined.
2. Loading: the Derivation Model is loaded, model transformations are applied if any, and declarative statements from the Derivation Model give raise to specific rules we generate making use of SCHACL rules.
3. Runtime: the rules generated during the *loading* are applied to the data of the SDM and the resulting graph is published in a graph database that can be queried by the user.

6.2.1 Specification step: writing the Derivation Model

Writing the Derivation Model allows producing an augmented ontology describing how IRs should be derived from the SDM and how the individuals to whom they refer should be depicted. The elements of the Derivation Model allow to operate on TBox (Terminological Components) and ABox (Assertional Components) and to enrich these two aspects of the models. This extension of the SDM at the ontological level enables the creation of the appropriate geovisualisation.

In the following we describe the minimum elements to be written in the Derivation Model as well as various optional features to specify precisely the correspondence between the data from the SDM and the desired portrayals.

Specifying minimum and essential elements of the Derivation Model

Writing the Derivation Model corresponds to the conceptualisation and prototyping steps of the geovisualisation, performed in RDF, that allows to describe which components (e.g. a map, a table, etc.) are desired and to describe which individuals of the SDM (through their classes and various filters) should be represented in it.

We reuse during this chapter the simple modelling introduced in Chapter 5 and containing the class `ex:Nuts2Unit` (it can be seen on Appendix A, Listing A.1). The case studies presented in Chapter 7 and Chapter 8 will focus on exploiting existing models that are richer in semantics, which will also highlight CoViKoa functionalities designed to exploit these semantics.

As announced in Chapter 5, our proposed default approach is to create a special subclass of `gviz:GeoVisualIntermediateRepresentation` for each class of the SDM to be shown in the geovisualisation (see example in Listing 6.1 for `:Nuts2Gvr`). We use

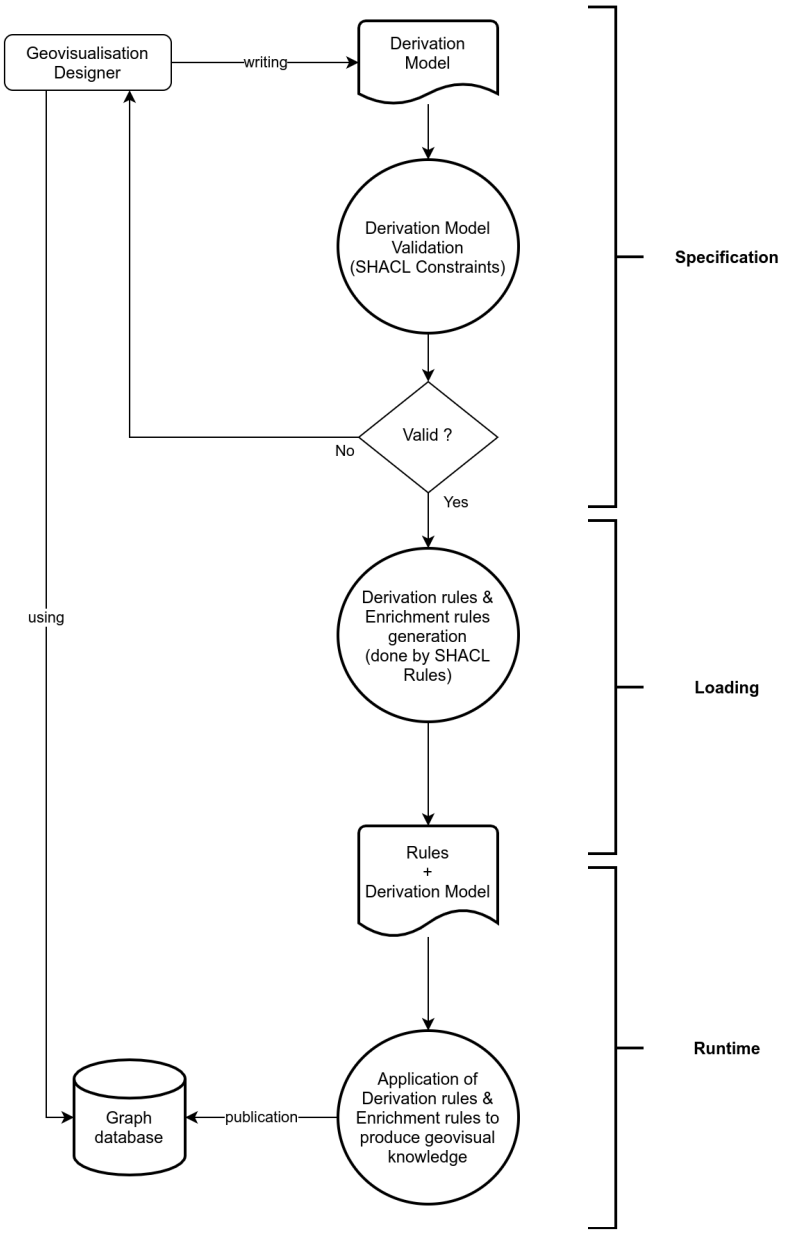

Figure 6.2: User main workflow.

here the annotation property `cvkd:represents` when defining this class to indicate which SDM class it refers to (here `:Nuts2Gvr` represents the class `ex:Nuts2Unit`). This property will be read and will be automatically materialised in the graph that will be created with the `gviz:represents` property (cf. Figure 5.1), between each IR and the individual it represents.

```
:Nuts2Gvr a owl:Class ;
    rdfs:subClassOf gviz:GeoVisualIntermediateRepresentation ;
    cvkd:represents ex:Nuts2Unit .
```

Listing 6.1: The simplest Derivation Model for a mock concept.

The next step is to define a `gviz:Portrayal` that indicates how to select the data (possibly using property constraints or spatial constraints) and how to represent them (using a symbol linked to a symbolizer). To keep this example concise (Listing 6.2) and to remain focused on writing the Derivation Model, we choose to represent all entities by the same grey symbolizer.

```
:myPortrayal a gviz:Portrayal ;
    cvkd:denotesGVR :Nuts2Gvr ;
    gviz:hasPortrayalRule [
        gviz:hasSymbol [
            symblzr:hasSymbolizer :SymbolizerNutsGrey ;
          ] ;
      ] .

:SymbolizerNutsGrey
    a symblzr:PolygonSymbolizer ;
    graphic:hasStroke [
        graphic:strokeColor "rgba(99,99,99,1)"^^graphic:cssColorLiteral ;
      ] ;
    graphic:hasFill [
        graphic:fillColor "rgba(200,200,200,1)"^^graphic:cssColorLiteral ;
      ] .
```

Listing 6.2: Defining a portrayal

Note that we do not use the `gviz:denotesConcept` property (which is intended to point to an SDM class) but the `cvkd:denotesGVR` property. This is what allows our system to generate the rules and to make the necessary links between this Portrayal and the IR generated thanks to the declaration of the Listing 6.1. As such, the `gviz :denotesConcept` property will be inferred by our system at runtime at the same time as the IRs and the materialisations will be created, thus making it possible to obtain a complete graph respecting the links foreseen by *covikoa-geoviz*.

Finally, since we are specifying a geovisualisation interface, it is also necessary to describe one or more components and link them to an application (Listing 6.3)

```
:myApplication a gviz:GeoVisualApplication ;
    gviz:hasGeoVisualComponent :myMap .

:myMap a gviz:Map2dComponent .
```

Listing 6.3: Defining a simple map component within a geovisualisation application

These three Listings (6.1, 6.2 and 6.3), put together in a document, constitute a valid Derivation Model for the model and data presented in Appendix A, Listing A.1.

One can note that it is not necessary to specify for which component the Portrayal is intended when the Derivation Model contains only one map component (`gviz:Map2dComponent`). However, this link must be made explicit (with the property `gviz:onComponent`) when several components are defined. Similarly, we saw in the previous chapter that it was possible to use the *Scale* ontology to qualify the validity of a Portrayal and its PortrayalRules, this is not shown here but it is exemplified in the case studies of the following chapters.

Leveraging the semantics expressed by the SDM and transforming its data

The flexibility of our approach, both because of the richness of the vocabularies used to specify the geovisualisation to be created and because it exploits data stored in an RDF graph, makes it possible to simply exploit the sometimes strong semantics embedded in the model being exploited. The use of filters, which can be used at different levels, enables the expression of spatial constraints, of constraints on properties and of complex filters in SPARQL as well as the integration of data using federated query.

While property constraints and spatial constraints were discussed in the previous chapter, other data processing facilities are essential for the creation of geovisualisations.

Indeed, the particular organisation of data in RDF may lead to the need to perform the **selection at the application level** of the entities to be represented, before the use of filters or constraints. This is for example the case with NUTS2 if we always want to represent individuals from a specific country, where `ex:Country` is a class. Let us assume we want to represent only French regions (NUTS2 entities that are linked to `ex:France` by the property `ex:hasCountry`, cf. Appendix A, Listing A.1). This is possible by indicating several additional pieces of information (Listing 6.4) to what has been seen so far: at the level of the `gviz:GeoVisualApplication` sub-class that is created (with the annotation property `cvkd:represents` to point to a specific class), at the level of its instantiation (with the object property `gviz:represents` to point to a specific individual of the class mentioned just before), and at the level of the declaration that is used to produce the IRs (with the annotation property `ckvd:pathFromGVA` to define the path between the individuals of this class and the specific individual mentioned just before). With such a declaration, only IRs corresponding to `ex:Nuts2Unit` linked to the country `ex:France` will be created. These triples thus allow the early selection of appropriate entities to be expressed, at the level of the application itself, offering in our example the flexibility to easily specify other applications referring to a different country.

```
1 :NutsAppCountry a owl:Class ;
```

```
    rdfs:subClassOf gviz:GeoVisualApplication ;
    cvkd:represents ex:Country . # ex:Country is a owl:Class

:nutsAppFr a :NutsAppCountry ;
    gviz:represents ex:France . # ex:France is an instance of ex:Country

:Nuts2Gvr a owl:Class ;
    rdfs:subClassOf gviz:GeoVisualIntermediateRepresentation ;
    cvkd:represents ex:Nuts2Unit ;
    cvkd:pathFromGVA [ # the directed path between ex:Country and ex:Nuts2Unit
        sh:inversePath ex:hasCountry ;
      ] .
```

Listing 6.4: Definition of a `gviz:GeoVisualApplication` designed to represent the individuals attached to a specific instance of a class.

We have seen in the previous chapter that **spatial constraints** can be defined using a constant value. This is also the case for **transformation operations** that are used to transform the geometry of the entities selected by a `gviz:PortrayalRule` or by all the `gviz:PortrayalRules` of a `gviz:Portrayal` and to link the transformed geometry to the `gviz:Materialisation` that is created (for example in Listing 6.5, which expresses that the geometry that will be represented will be the result of the `geof:difference` operation between a constant value and the geometry of each of the selected entities). An other typical example is the calculation of the centroid for a representation that would require it (e.g. a proportional symbols representation applied to zonal entities), however the centroid function is not available in GeoSPARQL and we will see later how we solve this (Section 6.3.1).

```
:somePortrayal
    a gviz:Portrayal ;
    cvkd:hasTransformOperation [ a cvkd:TransformOperation ;
        geof:difference (
            [cvkd:value "POLYGON (...)"^^geo:wktLiteral]
            [cvkd:variable "?thisGeometry"]
        )
      ] .
```

Listing 6.5: Example of `cvkd:TransformOperation` using a constant value.

We also believe that the user should not only be able to refer to constant values when writing `cvkd:TransformOperation` and `cvkd:SpatialConstraint`, but also to existing data. To this end we also provide vocabulary elements that allow us to pre-bind a variable and use this variable as an argument to the geometric transformation operations (Listing 6.6) as well as when defining spatial constraints (Listing 6.7). It should be noted that we expect the pre-binding to bind only one value to the variable in question (more complex cases of pre-binding involving an aggregation function will be presented in the next chapters).

```
:somePortrayal
    a gviz:Portrayal ;
    cvkd:hasTransformOperation [ a cvkd:TransformOperation ;
        cvkd:preludeBindings "?someSparql :stuff [ ?thatBindA ?variable ] ." ;
        geof:difference (
```

```
            [cvkd:variable "?variable"]
            [cvkd:variable "?thisGeometry"]
        )
    ] .
```

Listing 6.6: Example of `cvkd:TransformOperation` using a variable bound beforehand.

```
:someSpatialConstraint
    a cvkd:SpatialConstraint ;
    cvkd:preludeBindings "?someSparql :stuff [ ?thatBindA ?variable ] ." ;
    cvkd:hasPrediate geof:sfIntersects ;
    cvkd:variable "?variable" .
```

Listing 6.7: Example of `cvkd:SpatialConstraint` using a variable bound beforehand.

The syntax we use to define transformation operations is strongly inspired by the syntax of SHACL-AF *Function Expressions*[1]. The main difference with function expressions is that we do not qualify the path to a value but either a variable, with `cvkd:variable` (referring to a pre-bound variable such as `?thisGeometry` which refers to the geometry of the current entity to which the transformation applies) or a constant value with `cvkd:value` (typically a geometry serialised in WKT).

Finally we mentioned in Chapter 5 the existence of the `symblzr:TemplateSymbolizer` class allowing to define **customizable symbolizers** according to individuals to which they apply. Indeed the portrayal rules can either be linked to a `symblzr:Symbolizer` or to a `symblzr:TemplateSymbolizer` correctly qualified by customisable object of type `cvkd:CustomisableGraphicValue`. Listing 6.8 shows an example of `symblzr:TemplateSymbolizer` definition in which we want to customize two elements: the value of the `graphic:textLabel` property, so that the name of the entity is used (`ex:name` property) and the value of the `graphic:fontSize` property so that this size is proportional to the population of the entity (`ex:population` property) . This mechanism not only allows to transfer the value of a property (as it is the case for the label) but also to calculate a new value using a formula (`cvkd:formula`). This formula is written in its SPARQL literal form and the value targeted by the property is already bound to the variable with the name `?value`.

```
:labelNuts
    a symblzr:TemplateTextSymbolizer ;
    graphic:hasFont [ a graphic:Font ;
        graphic:fontFamily "Arial" ;
        graphic:fontSize [ a cvkd:CustomisableGraphicValue ;
            cvkd:valueOnProperty ex:population ;
            cvkd:formula "afn:max(afn:sqrt(?value/afn:pi()) / 60, 12)" ;
          ] ;
        graphic:fontWeight "bold" ;
          ] ;
    graphic:hasFill [ a graphic:Fill ;
        graphic:fillColor "rgba(99,99,99,1)"^^graphic:cssColorLiteral ;
      ] ;
    graphic:textLabel [ a cvkd:CustomisableGraphicValue ;
        cvkd:valueOnProperty ex:name ;
```

[1] https://www.w3.org/TR/shacl-af/#node-expressions-function

```
16      ] .
```

Listing 6.8: Example of `symblzr:TemplateTextSymbolizer`.

Writing property path in the Derivation Model

When writing various elements of the Derivation Model we allow the user to provide either a property, a list of properties, an alternative path between two properties or an inverse path. This is for instance the case for the annotation property `cvkd:pathFromGVA`, for `cvkd:propertyPath` (presented in Chapter 5) and for `cvkd:valueOnProperty` (which is used to point to a value to be used in a `symblzr:TemplateSymbolizer`). In order to adopt a well-known way of writing these different types of paths, we follow the SHACL way of writing property paths[2], knowing that we only support (because the others are not necessary for the needs of the framework) the following types :

- *Predicate Path* (a single RDF property),
- *Sequence Path* (a RDF list of property paths),
- *Alternative Path* (a blank node having one triple with `sh:alternativePath` as predicate and a RDF list of two elements, themselves property paths, as object),
- *Inverse Path* (a blank node having one triple with `sh:inversePath` as predicate and a property path as object).

Writing property paths in this form allows us to rewrite them in valid SPARQL property path (Listing 6.9) for use in the `sh:SPARQLRule` that are generated by our framework during the loading step. This rewriting is itself performed during the rule generation with in-house[3] SPARQL functions (`sh:SPARQLFunction`), as allowed by the SHACL-AF specification[4].

We believe that all of these mechanisms for selecting the data corresponding to a Portrayal Rule, for transforming its geometry if necessary and for customising symbolizers cover an essential part of the preparation of geospatial data prior to the creation of a geovisualisation interface.

```
1  SHACL PredicatePath: ex:isIn
2  Rewritten SPARQL Property path: ex:isIn
3
4  SHACL SequencePath: (ex:isIn geo:hasGeometry)
5  Rewritten SPARQL Property path: ex:isIn/geo:hasGeometry
6
7  SHACL AlternativePath: [ sh:alternativePath ( ex:father ex:mother ) ]
8  Rewritten SPARQL Property path: ex:father|ex:mother
9
10 SHACL InversePath: [ sh:inversePath ex:contains ]
```

[2] https://www.w3.org/TR/shacl/#property-paths

[3] These SPARQL functions can be consulted in the turtle file on https://github.com/mthh/covikoa/blob/master/rdf/rules-base-functions.ttl.

[4] https://www.w3.org/TR/shacl-af/#functions

```
11 Rewritten SPARQL Property path: ^ex:contains
```

Listing 6.9: SHACL Property path that are valid in CoViKoa Derivation Model to describe a path of property.

Writing elements of the derivation model using cartographic vocabulary where possible

We have proposed in Chapter 5 the vocabulary *covikoa-carto* dedicated to cartography and allowing to describe well known representations by a set of parameters. We also showed that there could be a correspondence between the `gviz:Portrayals` of *covikoa-geoviz* (qualified by `gviz:PortrayalRules` and themself by `cvkd:PropertyConstraint`) and the solutions described in *covikoa-carto* .

We thus propose to the user of our approach to use *carto* to describe the portrayals that allow it (Listing 6.10). This type of transformation is currently implemented for three of the solutions we have identified in *carto* vocabulary (`carto:QuantitativeChoropleth`, `carto:QualitativeChoropleth` and `carto:ProportionnalSymbols`) but we plan to describe new cartographic solutions in *carto* and support them in CoViKoa.

Using this approach considerably reduces the number of triples to be written in the Derivation Model, thus reducing the risk of error when writing `gviz:PortrayalRule`, and also allows to avoid writing `symblzr:Symbolizer` manually if the graphical type of the data to be represented is specified (using subclasses of `carto:GraphicData`).

```
1 :ChoroplethUnempNuts a carto:QuantitativeChoropleth ;
2     dct:title "Unemployment Rate (NUTS2 units, 2013, %)" ;
3     carto:targetsSpatialFeature ex:Nuts2Unit ;
4     carto:targetsProperty ex:unemploymentRate ;
5     carto:ofData [ a carto:Area , carto:RelativeQuantitativeData , carto:Number ] ;
6     carto:hasDataBreaks (0.0 4.0 10.4 16.6 24.2 36) ;
7     carto:rendersNoData "true"^^xsd:boolean ;
8     carto:hasPaletteScheme dcpal:Blues .
```

Listing 6.10: Defining a portrayal by instantiating a `carto:CartographicSolution`

Because some of the portrayals needed in the geovisualisation to be created may be part of classic cartographic solutions described in *carto*, while others may require a more advanced selection of entities or choice of symbolizers, it is possible to mix, in the same Derivation Model, the two types of declarations (the declarations in the form of `carto:CartographicSolution` are in fact transformed into `gviz:Portrayal` and `gviz:PortrayalRules` in order to take advantage of the constraint mechanism that we have presented in Chapter 5). The specifics of the use and transformation of this statement is highlighted later, in Section 6.2.2.

SHACL shapes to validate the derivation model and to enforce good geovisualisation practices

Since writing RDF can sometimes be tedious or error-prone, we provide *SHACL shapes*[5] that check and validate the Derivation Model written by the user when opened by the framework.

These shapes are intended to be informative, to allow the user to easily solve his or her problems thanks to the error messages returned (these are *constraint violations* in the sense of SHACL, implying the invalidity of the tested graph (the Derivation Model) and therefore an early exit from the CoViKoa application as shown on Figure 6.2. A sample validation report is included in Appendix B, Listing B.1.

In addition, these shapes also include various elements relating to good cartographic practices, and in particular concerning the various essential elements to be shown on a map or in a geovisualisation, some of which were identified in Chapter 2 and formalised by the authors mentioned in Chapter 4. These good practices we propose as rules are of various kinds and non-compliance with them does not lead to a constraint violation because they only use the *warning* level of severity.

Some of them can be seen as adaptations of the constraints formalised in OWL by Penaz et al. (2014)[6]. Indeed we would like to encourage the user to give titles to the elements represented (the map(s) and also the portrayals and symbols they mobilise) as well as to encourage him or her to show the elements in the legend for example. We also encourage the implementation of some interactions that are almost indispensable in geovisualisations.

We believe that, even if some of these functionalities are not desired by the user, and this is probably a conscious choice, the display of these recommendations in the form of warnings makes it possible to engage or renew the user's thinking about the geovisualisation he or she is about to create.

6.2.2 Loading step: rule generation

When the framework is started and after its validation, the Derivation Model is read by the framework in order to be transformed according to the needs of the framework and to generate rules, specific to the data to be exploited and to the elements present in the said Derivation Model.

Indeed, if the Derivation Model contains declarations in the form of instances

[5]These SHACL shapes can be consulted in the turtle file on https://github.com/mthh/covikoa/blob/master/rdf/shapes-derivation-model.ttl.

[6]Where we can see rules such as "*... it contains at least one element of map composition which is the mapped area*", "*... it contains at least one element of map composition which is title and subtitle*", etc.

of `carto:CartographicSolutions` (as Listing 6.10), they are expanded[7] into their corresponding `gviz:Portrayal` and inserted into the Derivation Model (an example of validation report is provided in Appendix C, Listing C.2). Declarations in the form of `carto:CartographicSolutions` are also subject to validation by the SHACL mechanism presented in the previous sub-section so that no invalid declaration can go through to this stage.

Next comes the SHACL rule generation step. This step is itself performed using SHACL rules included in the framework[8]. Depending on the Derivation Model, the rules that can be created are of two or three kinds:

- derivation rules (generated in all cases)
- symbolizer creation rules (only if necessary)
- enrichment rules (generated in all cases)

A fourth type of rules, the data integration rules, are SHACL rules written directly by the user in the derivation model and allowing to integrate data in the CoViKoa graph. These rules can be executed when the CoViKoa reasoning is first applied or when a specific condition is met (allowing for example the integration of data, in the form of a federated query, relating to a study area described in a particular dataset - this is exemplified in both Chapter 7 and Chapter 8).

Derivation rules, to create the IRs

Derivation rules (`cvkd:DerivationRule`) are used to produce the IRs in accordance with the Derivation Model. A rule of this type is created for each sub-class of `gviz:GeoVisualIntermediateRepresentation` written in the derivation model.

The logic that is implemented to create these rules depends on whether the application has been defined to perform an early selection of the individuals from classes to portray related to a particular individual of another class (as in Listing 6.4) or not.

If not (this is the default case we presented in Listing 6.1 and 6.3) the rule generated by our system creates an IR (of the subclass of `gviz:GeoVisualIntermediateRepresentation` that is defined by the user, such as `:Nuts2Gvr` in Listing 6.1) for each individual of the represented concept (with the annotation property `cvkd:represents`).

In contrast, if the application has been defined to represent individuals related to a particular individual (this is the case we presented in Listing 6.4), then the IRI of that individual as well as the property path between it and the individuals of

[7] These SHACL rules can be consulted in the turtle file on https://github.com/mthh/covikoa/blob/master/rdf/rules-carto-to-geoviz.ttl.

[8] These SHACL rules can be consulted in the turtle file on https://github.com/mthh/covikoa/blob/master/rdf/rules-generation-rules.ttl.

each class to be derived (defined with the annotation property `cvkd:pathFromGVA`) is injected into the rule. This allows IRs to be created (as before, from the sub-class of `gviz:GeoVisualIntermediateRepresentation` defined by the user in the Derivation Model) only for those individuals that require them, those related to the specified individual.

Symbolizer creation rules, to create custom symbolizers from template symbolizers

The *symbolizer creation rules* (`cvkd:SymbolizerCreationRule`) allow to pass from `symblzr:TemplateSymbolizer` to the corresponding `symblzr:Symbolizers` for each of the individuals that requires it with regard to the `gviz:PortrayalRule` which is linked to it.

Internally, this is done by reading the various `cvkd:CustomisableGraphicValue` present at the level of a `symblzr:TemplateSymbolizer` and by injecting the appropriate values (i.e. the SPARQL property path allowing to reach the targeted value and possibly the specified formula) in the symbolizer creation rule created. A symbolizer creation rule is created for each `symblzr:TemplateSymbolizer`.

```
:labelNuts-b893421c-4187-4734-b0d8-87d9df2d4bfd
    a symblzr:TextSymbolizer ;
    graphic:hasFont [ a graphic:Font ;
        graphic:fontFamily "Arial" ;
        graphic:fontSize 12 ;
        graphic:fontWeight "bold" ;
      ] ;
    graphic:hasFill [ a graphic:Fill ;
        graphic:fillColor "rgba(99,99,99,1)"^^graphic:cssColorLiteral ;
      ] ;
    graphic:textLabel "Ankara".
```

Listing 6.11: Example of `symblzr:TextSymbolizer` derived from a `symblzr:TemplateTextSymbolizer`.

They thus make it possible when executing the rules, for example, to create, from the description of a `symblzr:TemplateTextSymbolizer` with multiples `cvkd:CustomisableGraphicValue` (Listing 6.8), the corresponding `symblzr:TextSymbolizer` (Listing 6.11), given an individual[9] who would have the value "Ankara" as the object of the property `ex:name` (used for the value of the `graphic:textLabel` property) and who would have "5045083" as the object of the property `ex:population` (used for the value of the `graphic:fontSize` property).

It should be noted that the *symbolizer creation rules* only create the symbolizers but do not relate them to the `gviz:Materialisation` of the corresponding individuals. This is done by the rules described in what follows.

[9] It can be seen in Appendix A, Listing A.1.

Enrichment rules, to create the materialisation of an individual on a component and link it to its symbolizer

Enrichment rules (`cvkd:EnrichmentRule`) make it possible to create the necessary `gviz:Materialisation` of each IRs and to establish the link between these materialisations and the component on which they appear as well as with the corresponding symbolizer(s).

It is in these *enrichment rules* that the various spatial constraints and property constraints as well as any geometric transformation operations are injected, thus making it possible to link the materialisations created to the proper symbolizer and to attach to them the transformed geometry if necessary. As such, one *enrichment rule* is created for each `gviz:PortrayalRule` present in the Derivation Model.

Internally, we distinguish two subclasses of enrichment rules that we call *default enrichment rules* and *problem-specific enrichment rules*. The latter are generated by `gviz:Portrayals` that are defined in the Derivation Model using a SPARQL filter (`cvkd:SPARQLFilter`) in contrast to the former. Indeed, the definition of a portrayal with a SPARQL filter is what allows, for example, to implement a complex logic of selection of the individuals to be portrayed, exploiting the way concepts are organised in SDM and corresponding to the resolution of a specific problem, hence the name of *problem-specific enrichment rules* that we give them.

6.2.3 Runtime: rule application and publication

It is during the runtime stage that the four types of rules described in the previous subsection are executed. This execution allows the integration of any data that may be required (*data integration rules*), the creation of intermediate representations of individuals that are to appear on one of the components of the geovisualisation (*derivation rules*), the creation of symbolizers from templates if necessary (*symbolizer creation rules*) and finally the creation of the materialisation, for each component, of each individual that requires it, in particular by linking this materialisation with the symbolizer(s) that symbolize(s) an individual (*enrichment rules*).

Recursive reasoning / cached inference

Since the knowledge necessary for the execution of *enrichment rules* is created by the *derivation rules* (and possibly by the *symbolizer creation rules*), it is necessary to set up a specific strategy concerning the knowledge produced by these rules, which is produced in a new empty graph as per the SHACL-AF specification.

To do this we use a strategy sometimes referred to as "cached inference" (Allemang and Hendler, 2011) in which the triples produced by these rules are unioned

with the graph containing the SDM and the rule mechanism is restarted on this union graph until no new triples are produced. This is this union that we call the CoViKoa Default Graph (cf. Figure 6.1).

When an UPDATE operation occurs, this mechanism is retriggered:

- either directly if knowledge is only added to the graph,
- or after having taken care to detach the knowledge previously produced by the rule mechanism during an operation updating existing values or deleting data in the graph.

We also note that the data graph is subject to an OWL inference regime (by default using Jena's OWL Micro reasoner which covers most of RDFS-Plus and part of OWL2 RL presented in Chapter 3 - a rule-based OWL2 RL reasoner can also be used if required). This makes it possible to materialise the various deductions that can be made from the SDM or from the ontologies that we propose, thus facilitating the queries that can be made.

We refer to this whole mechanism as "CoViKoa reasoning" in the rest of the manuscript.

An overview of the individuals and the links created by the rules

CoViKoa reasoning which executes the previously generated rules creates different individuals and different links in the CoViKoa graph which is published.

In the case of the simplest derivation model presented by the concatenation of Listings 6.1, 6.2 and 6.3 :

- the derivation rule creates as many IRs of type `:Nuts2Gvr` as there are NUTS2 entities,
- the enrichment rules creates a `gviz:Materialisation` for each individual and link this materialisation to the appropriate IR and to the appropriate symbolizer (there is only one in our example),
- the enrichment rules also creates the link `gviz:denotesConcept` between the portrayal and the SDM class `ex:Nuts2Unit`.

Figure 6.3 shows these various elements, in grey (step 1, specification) an extract of the elements written by the user in the Derivation Model, in green (step 2, loading) the rules generated by our system, and in pink (step 3, runtime) the individuals and links created by our system.

Other knowledge could have been created by CoViKoa reasoning if template symbolizers had been used or if a data integration rule had been used in the DM. This is exemplified in the case studies of Chapter 7 and Chapter 8.

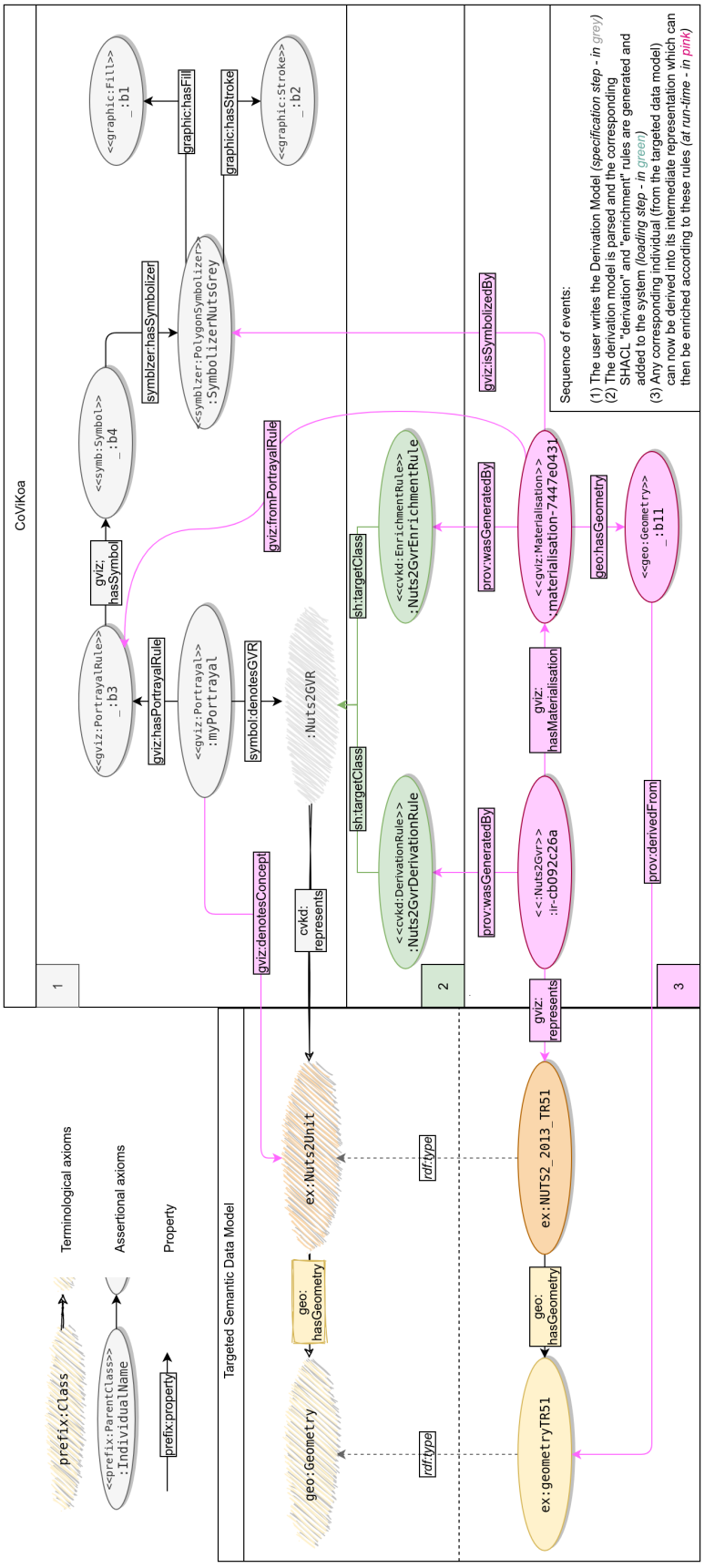

Figure 6.3: Individuals and links created by CoViKoa.

6.3 Implementation

In this section, we describe the specificities of the implementation of CoViKoa without focusing on its use (installation, launching, etc.). All the instructions for its use are available on the Git repository, as well as a Docker[10] recipe allowing to use CoViKoa to play the seven case studies proposed on the repository without having to install any dependencies (except Docker).

6.3.1 RDF data management and reasoning with Apache Jena

The implementation of the framework (Figure 6.4) is mostly done in Java and uses the Jena Semantic Web framework (presented in Section 3.3.2). The SHACL engine used is the one developed by TopQuadrant (presented in Section 3.3.3), which is the reference implementation and implements all the SHACL specification documents. In addition, various features specific to frameworks for the Semantic Web and offered by Jena are used to make our framework more capable and easier to use.

Easing the consumption of some RDF data by implementing custom Property Functions

Indeed, Jena offers the possibility to define, in Java, custom property functions[11]. These are properties that extend the SPARQL language by allowing a function to be executed by the query processor.

We use this possibility to provide a simpler JSON representation of the symbolizers contained in the CoViKoa graph. Indeed, the RDF structure of these symbolizers makes their querying tedious, in particular when a query do not know what type the symbolizer is, which is often the case when consuming the portrayals described in the graph by a generic client like the one presented in the next section (the structure of these symbolizers in RDF is recalled in Listing 6.12). The use of the property function created greatly simplifies the querying (Listing 6.13) and processing of symbolizer descriptions by allowing the entire description of a symbolizer to be retrieved, in a single SPARQL column, in JSON format (Listing 6.14).

```
:exampleSymbolizerPoint1 a symblzr:PointSymbolizer ;
    graphic:graphicSymbol [ a graphic:GraphicSymbol ;
        graphic:hasMark [ a graphic:Mark ;
            graphic:hasWellKnownName graphic:circle ;
            graphic:hasStroke [ a graphic:Stroke ;
                graphic:strokeColor "rgba(20,128,0,1)"^^graphic:cssColorLiteral ;
              ] ;
            graphic:hasFill [ a graphic:Fill ;
                graphic:fillColor "rgba(127,255,0,1)"^^graphic:cssColorLiteral ;
```

[10] https://www.docker.com/

[11] https://jena.apache.org/documentation/query/writing_propfuncs.html

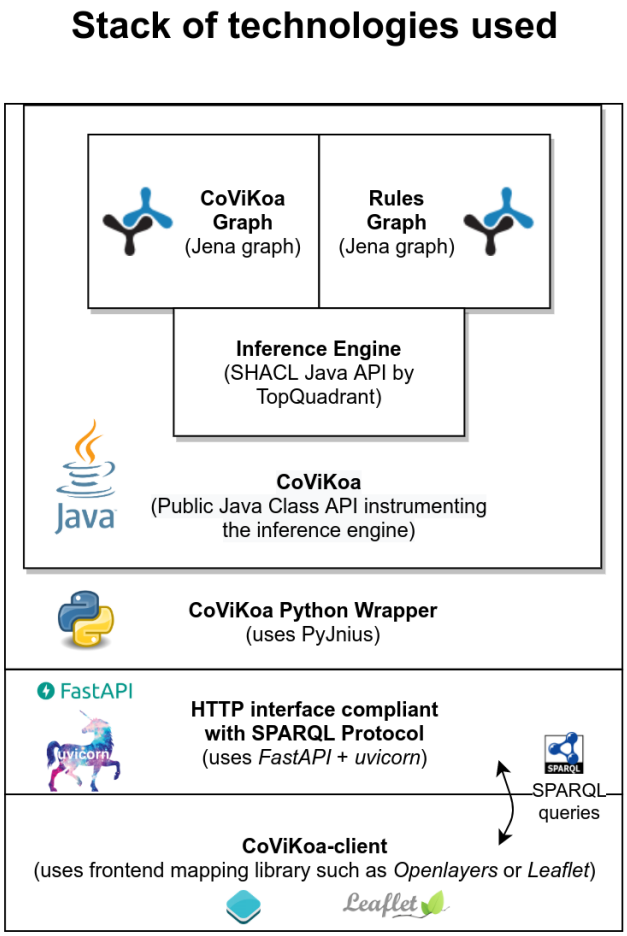

Figure 6.4: Technological stack of CoViKoa implementation.

```
            ] ;
        ] ;
      graphic:size "5"^^xsd:decimal ;
    ] .


:exampleSymbolizerPoint2 a symblzr:PointSymbolizer ;
    graphic:graphicSymbol [ a graphic:GraphicSymbol ;
        graphic:hasExternalGraphic [ a graphic:ExternalGraphic ;
            graphic:onlineResource """[...]"""^^xsd:anyURI ;
            graphic:format "image/png" ;
        ] ;
        graphic:size "24"^^xsd:decimal .
    ] .
```

Listing 6.12: Example of symbolizers.

```
SELECT ?jsonProperties WHERE {
    :exampleSymbolizerPoint symblzr:asJSON ?jsonProperties
}
```

Listing 6.13: Example of SPARQL query to obtain the JSON description of the symbolizer `:exampleSymbolizerPoint`.

```
{
  "color": "rgba(127,255,0,1)",
  "kind": "Mark",
  "wellKnownName": "circle",
  "radius": 5,
  "strokeColor": "rgba(20,128,0,1)"
}
```

Listing 6.14: Resulting JSON description of the symbolizer `:exampleSymbolizerPoint`

In order not to introduce too many new vocabulary elements (this is also the reason why we use the *Symbolizer* and *Graphic* ontologies, whose vocabulary is well known to users of OGC standards), the JSON description produced uses the vocabulary of the geostyler style format[12]. Indeed, this format is intended to be an exchange format, capable of storing the styling capabilities of existing formats such as SLD, OpenLayers, CartoCSS, etc. Although it is not widely adopted, we believe it is beneficial to use an existing format rather than crafting our own.

Extending GeoSPARQL functions by implementing custom Aggregate Functions

We also found that GeoSPARQL does not define all the classical functions of spatial DBMSs. For example, the possibility to calculate a centroid is not defined. Similarly, no aggregate function (taking an undefined number of entities in a query calling the `GROUP BY` clause and computing a single value - `SUM` is a classical example of aggregate function) is defined by GeoSPARQL. Such features can be useful during

[12] https://github.com/geostyler/geostyler-style

the process of creating a geovisualisation, whether it is to form the geometric union of a set of entities, to compute the geometry resulting from the intersection of a set of geometries or even to compute the extent represented by a set of entities.

To remedy this, we define three spatial aggregate functions: *extent*, *union* and *intersection* (here using the prefix `geo-agg` that corresponds to the namespace http://lig-tdcge.imag.fr/steamer/covikoa/geo-agg#). The extent function (Listing 6.15) is for example used by one of the rules of our framework in order to calculate and insert in the graph the initial extent (with `cvkc:hasMapExtent`) of `gviz:Map2dComponent`s when the user used the `cvkd:hasDefaultExtent` property to link to a class that is portrayed (such as `ex:Nuts2Unit` used in the previous examples).

```
SELECT (geo-agg:extent(?geom) as ?extentGeoms) WHERE {
    ?indiv a :SomeClass .
    ?indiv geo:hasGeometry [ geo:asWKT ?geom ] .
}
```

Listing 6.15: Example of use of custom GeoSPARQL aggregate function.

In a similar way, we implement some classical functions that we have found to be missing from GeoSPARQL and useful in the process of data transformations for geovisualisation. This is for example the case of the centroid function that we have implemented in SPARQL (Listing 6.16). It is typically used to calculate the position of point or text symbolizers when applied to polygonal features (although the choice of centroid for this purpose is debatable and there are solutions such as the PolyLabel algorithm[13] that might be more relevant for these purposes).

```
SELECT ?centroid WHERE {
    :someIndividual geo:hasGeometry [ geo:asWKT ?geom ] .
    BIND(fun:centroid(?geom) as ?centroid)
}
```

Listing 6.16: Example of use of custom GeoSPARQL function.

6.3.2 Exposing the knowledge behind SPARQL interface

Due to the particular instrumentation of the reasoning that we needed to code in Java, it is not possible to publish the graph directly with Jena Fuseki.

We have therefore chosen to make a Java API for CoViKoa publicly available (allowing notably to make the various types of SPARQL queries and updates on the CoViKoa graph). This public Java API is then wrapped in Python, using PyJnius[14], to create a web server partially implementing the SPARQL protocol. The implementation of the web server is done with the Python libraries FastAPI[15] and uvicorn[16].

[13] https://github.com/mapbox/polylabel
[14] A Python library for interacting with Java classes: https://github.com/kivy/pyjnius.
[15] https://github.com/tiangolo/fastapi
[16] https://github.com/encode/uvicorn

As such, the server allows to process queries[17] via the GET HTTP method as well as queries[18] and updates[19] via the POST HTTP method using URL-encoded parameters.

Conversely, what our implementation does not support are queries[20] and updates[21] via POST directly (i.e. with non URL-encoded parameters). The other element that we do not support (it is not mandatory in the SPARQL protocol specification) is the possibility of specifying an RDF dataset to which the query[22] or the update[23] refers. In our implementation it always refers to the CoViKoa default graph.

Assuming the server is started to listen at address `http://0.0.0.0:8000`, the exposed paths are:

- `http://0.0.0.0:8000/CoViKoa`: allows to process queries via GET (using the query string parameter "query"), queries and updates via POST, and returns the dataset materialised in Turtle serialisation if a GET request is made without a query string parameter (such as asking for `http://0.0.0.0:8000/CoViKoa` in a web-browser),
- `http://0.0.0.0:8000/CoViKoa/query`: allows to process queries via GET and POST methods,
- `http://0.0.0.0:8000/CoViKoa/update`: allows to process updates via POST,
- `http://0.0.0.0:8000/`: shows the web page corresponding to the geovisualisation (on GET method requesting html content).

6.3.3 Direct consumption by a Web client

All the knowledge specific to geovisualisation is thus available behind a SPARQL interface. We have chosen to query this knowledge directly from the client in order to avoid adding a level of indirection to the architecture proposed for our proof of concept.

While such an architecture would not be conceivable with relational databases, we believe that this is a realistic possibility in an architecture using Semantic Web technologies in which such SPARQL endpoints can be publicly exposed on the Web.

[17] https://www.w3.org/TR/2013/REC-sparql11-protocol-20130321/#query-via-get
[18] https://www.w3.org/TR/2013/REC-sparql11-protocol-20130321/#query-via-post-urlencoded
[19] https://www.w3.org/TR/2013/REC-sparql11-protocol-20130321/#update-via-post-urlencoded
[20] https://www.w3.org/TR/2013/REC-sparql11-protocol-20130321/#query-via-post-direct
[21] https://www.w3.org/TR/2013/REC-sparql11-protocol-20130321/#update-via-post-direct
[22] https://www.w3.org/TR/2013/REC-sparql11-protocol-20130321/#dataset
[23] https://www.w3.org/TR/2013/REC-sparql11-protocol-20130321/#update-dataset

Furthermore, the implementation demonstrated here is not intended to be published anywhere other than on a local machine or network.

Overview of the web client

In this work we implement a generic client, *covikoa-client* using the OpenLayers[24] webmapping library (`cvkc:openlayers` in the *covikoa-context* ontology) and the OpenGlobus[25] 3d webmapping library (`cvkc:openglobus`).

The idea behind this client is to implement each component that can be described with the *covikoa-geoviz* vocabulary. This is what we do by subclassing the `Widget` class of the Lumino library[26] to create a specific class representing each component of *covikoa-geoviz*. The use of the Lumino library (which is also behind JupyterLab[27] application) makes it easy to obtain the resizing and moving behaviour of the various components of the interface while benefiting from a messaging and signaling system allowing them to communicate with each other.

In order to use the knowledge of the graph to prepare the geovisualisation interface, several SPARQL queries are performed by *covikoa-client*. A first query is made to retrieve information about the application: what is its title, what components does it consist of and what are the characteristics of these components. This query can be seen as retrieving information about the "outside" of the components (size, position in the interface, title, etc.) and corresponds roughly to the knowledge depicted on the right of Figure 6.6.

Then, for each of its components, a SPARQL query is performed to obtain the description of its "interior", to know what data appear in it and how as well as the possible interaction defined on the entities of the component. These queries, executed on the locally published CoViKoa graph, are efficient and executed in a non-blocking manner in order to obtain reasonable loading times for the application.

This efficiency was verified using the Lighthouse application[28], which allows audits to be carried out notably on the performance of Web applications. Using Lighthouse[29], we were able to verify that we were getting values falling into the 'fast' category (Figure 6.5) for measures such as *speed index* (the amount of time it takes for the contents of a page to be visibly populated) and *time to interactive* (the amount of time it takes for the page to become fully interactive) when auditing a geovisualisation of 323 NUTS2 features containing two maps (one with two portrayals and the other with a single portrayal) and their legends as well as a table (as shown in

[24] https://openlayers.org/
[25] http://www.openglobus.org/
[26] https://github.com/jupyterlab/lumino/
[27] https://github.com/jupyterlab/jupyterlab
[28] https://developers.google.com/web/tools/lighthouse/
[29] This test was performed in Chromium browser, on a laptop powered by an Intel®Core™i7-8650U CPU @ 1.90GHz and 32GB of RAM, running a GNU/Linux operating system.

Figure 6.6[30]).

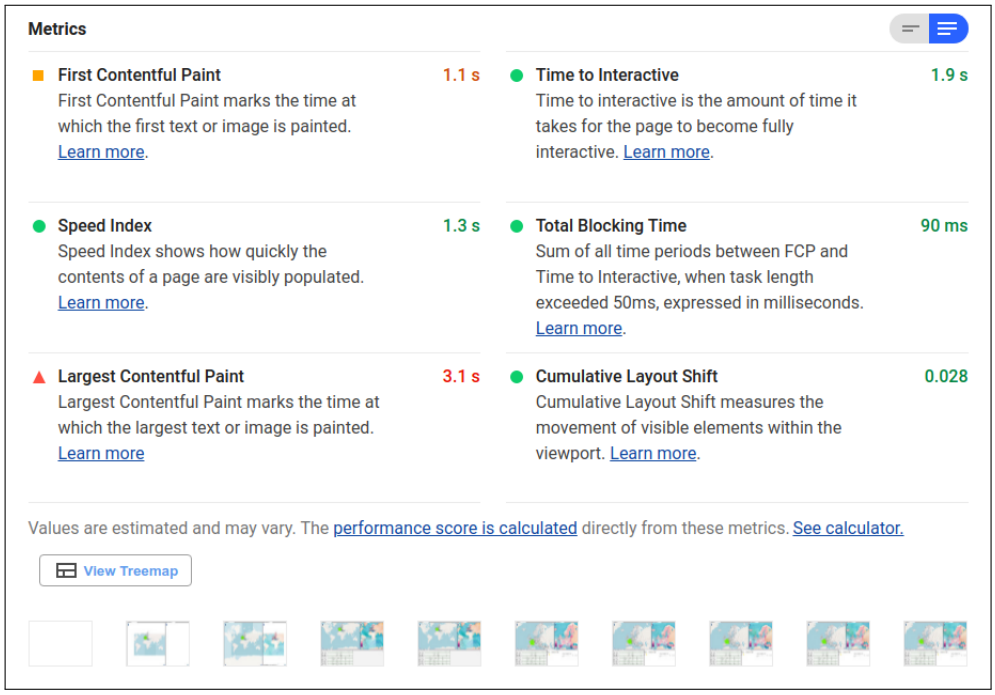

Figure 6.5: Screenshot of Lighthouse Report.

Despite the existence of *covikoa-client* to serve as a proof of concept, we note that the graph published by CoViKoa and using the ontologies described in Chapter 5 could be consumed by different types of clients (each providing their own implementation of the specified components such as `gviz:Map2dComponent`, etc.) and that the implementation of such a client does not necessarily have to be done as a Web application (a desktop application querying the CoViKoa graph is also a possibility, for example if one wishes to choose a specific library which does not exist in Web version to implement `gviz:Map3dComponent`). Beyond other clients for CoViKoa, we believe that it is possible to exploit the knowledge graph in other ways that may be either more efficient (by directly generating application-specific code) or serve slightly different purposes (creating a WMS map server for instance).

6.3.4 Other possible techniques for exploiting the knowledge base

Generation of the application code

The approach developed so far in this thesis implies that its user uses the generic client *covikoa-client* presented in the previous subsection. However, this approach

[30]The code and data for the case study allowing to produce this geovisualisation interface is provided in the code repository as "case study #1".

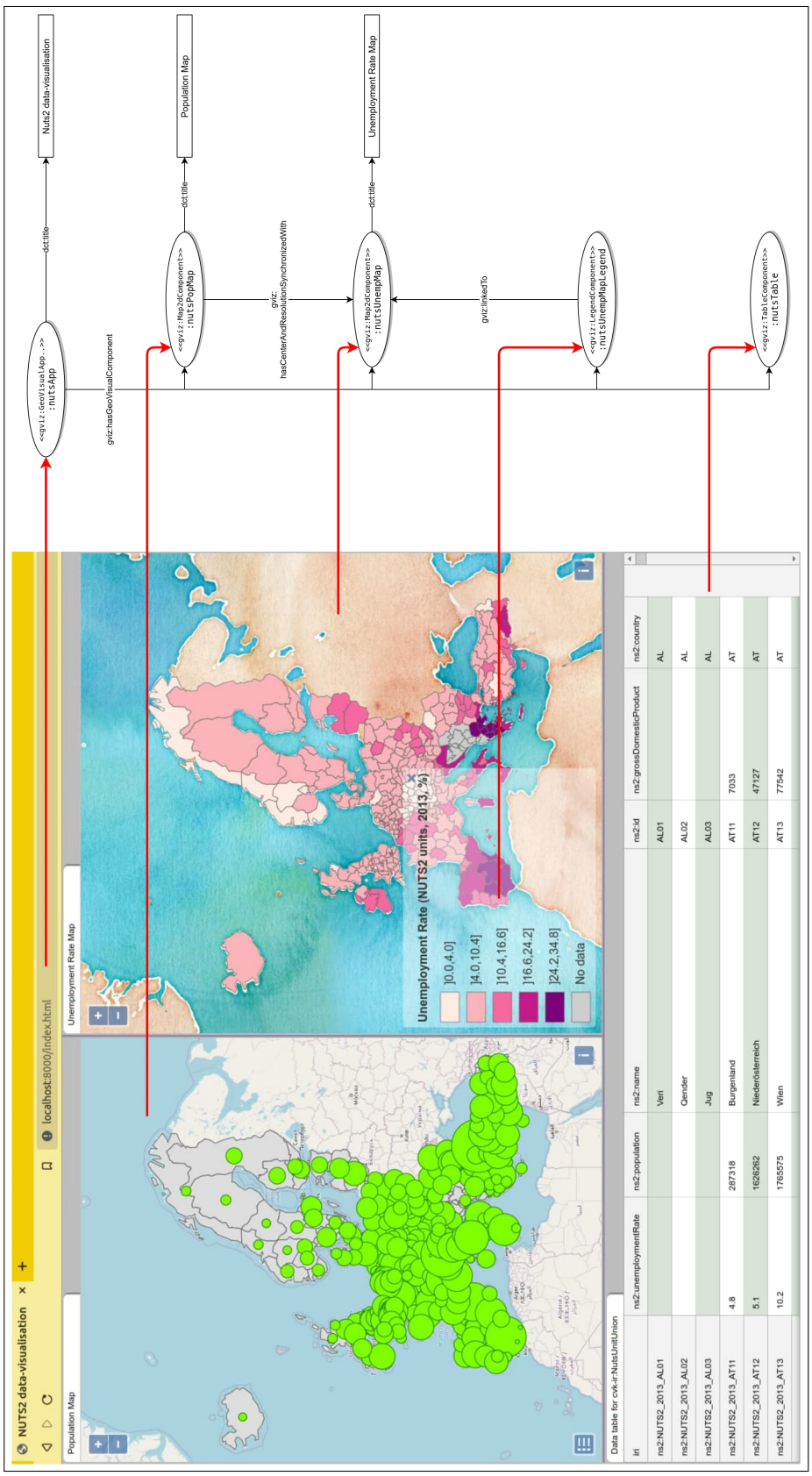

Figure 6.6: Screenshot of covikoa-client with NUTS2 data (Appendix A).

does not result in JavaScript code tailored to the application being described.

We make the hypothesis that another interesting and more powerful approach could be to use the graph describing the geovisualisation to produce executable source code (still using HTML and JavaScript for example) that implements specifically this geovisualisation.

The interface produced would not be generic but specific to the corresponding CoViKoa graph. The data graph could thus not be used when the web page is opened but only when the geovisualisation is produced. This approach would allow a web developer user to easily integrate his or her own components (e.g. data related diagrams) by adding its JavaScript code to that already produced.

Implementation of a WMS server

Although this hides the aspects of our proposal relating to the geovisualisation interface and its components, we believe that our approach could easily be used to allow the description of styles to be used to produce WMS tiles from RDF data.

Indeed, our approach allows the declaration of graphical styles, but one of its strengths is the establishment of the link between these styles and the data. A user wanting to develop a WMS server for RDF data could easily use CoViKoa to prepare the link between the data and its symbolizers and then connect the tile creation process to the triplestore containing the CoViKoa graph rather than to a spatial DBMS as it would traditionally be the case, and thus retrieve in the same query the geometries of the entities to be represented and their corresponding symbolizers.

Such a possibility strongly echoes the proposal of Iosifescu-Enescu and Hurni (2007) about a generic architecture for rule-based symbolisation.

6.4 Conclusion

The CoViKoa framework uses the vocabularies presented in Chapter 5 to describe a geovisualisation in a specification document, the Derivation Model. When writing the Derivation Model (Section 6.2.1), the user does not establish the links between the SDM data and the graphical elements to be used but describes how this link should be established. This can be done in a purely declarative way using various selection mechanisms, constraints and filters defined in *covikoa-derivation* vocabulary. Thanks to the Derivation Model, it is also possible for the user to describe in a much more succinct manner the portrayals to be implemented when it comes to the implementation of a well-known cartographic solution already described in *carto* vocabulary.

The loading of the Derivation Model in the CoViKoa application (Section 6.2.2) allows to build the RDF graph containing the SDM data and the links with the elements allowing to qualify how to represent them in the geovisualisation application. The linkage between the SDM data and the graphical elements is done using SHACL rules that were generated when the framework was loaded in order to contain the various constraints, filters and transformations specific to the derivation model provided by the user. The RDF graph so produced by the execution of these rules (Section 6.2.3) is stored in a graph database. This graph database is implemented using Jena which we use notably to instrument the SHACL engine and to add new SPARQL functions (Section 6.3.1). This graph database is finally published behind a SPARQL interface allowing the user to query the geovisualisation knowledge (Section 6.3.2).

The whole mechanism is implemented and we also implement a Web client, allowing to transform the RDF graph describing a geovisualisation into the corresponding Web application (Section 6.3.3).

In the end, the use of CoViKoa framework makes it possible to describe in RDF and to generate complex geovisualisation interfaces for Semantic Web data.

In the following chapters we illustrate the usefulness of our approach to describe and create several geovisualisation interfaces. In Chapter 7 we start by presenting the Choucas Alert Ontology (OAC) that we created in the context of the CHOUCAS project, and then examples of specification documents intended for the geovisual exploitation of data described using OAC. Then, in Chapter 8, we present a geovisualisation interface dedicated to exploiting data described by the TSN and TSN-change ontologies (Bernard, 2019) a recently proposed model used to describe changes in statistical territorial units.

Chapter 7

Case study using data from the CHOUCAS project

7.1 Introduction

In Chapter 5, we have presented a set of ontological vocabularies that enables the description of a geovisualisation under several aspects: its components, the way data appear in it and the interactions that can occur.

We then have shown in Chapter 6 how the CoViKoa framework allows a geovisualisation designer to move from a specification document applying to a data model to be used to a concrete geovisualisation interface.

This chapter exemplifies how our approach can be enforced through a case study from the CHOUCAS project (Olteanu-Raimond et al., 2017). Therefore, we first present an ontological model dedicated to the processing of victim search in mountains (Section 7.2), produced within the project and which serves here as the data model to which our framework applies. Then this whole case study is unfolded, through the presentation of the Derivation Model written for allowing the creation of a geovisualisation application, which will be illustrated with the help of screenshots[1]. This case study highlights several of the strengths of our framework about the various possibilities of data selection and filtering (Section 7.3), the possibilities of portrayal depending on the zoom level (Section 7.4) as well as the possibility of defining several components and interactions (Section 7.5).

[1]The code and data for this chapter are provided in the code repository as "case study #2" (up to and including Section 7.4) and as "case study #3" and "case study #4"(from Section 7.5, depending on the data displayed).

7.2 Choucas Alert Ontology

The Choucas Alert Ontology (OAC[2]) is an ontological model we have created in collaboration with the members of the CHOUCAS project and notably with the PGMH's referring rescuer in order to explicit the knowledge related to the handling of a mountain victim search. The OAC's construction results from the analysis of real cases of alerts, from which we have identified the concepts on which is based the reasoning of the rescuers from the PGHM.

The OAC formalises the concepts involved in the processing of an alert from raising the alert to identifying a victim's location, operated in a geovisualisation environment. The formalisation thus integrates concepts related to the restitution of information and to the interactions proposed in the tool since they are vectors of this reasoning. This ontology has a high level of expressivity (OWL 2 Full) in order to describe different subtleties in the organisation of spatial relations between the victim and the area he or she describes. This ontology, designed for the needs of the CHOUCAS project has been documented in French and is published at URL http://purl.org/oac.

We present in this manuscript a subset of OAC that we call LOAC (for Light OAC). It is published at URL http://purl.org/loac. The concepts present in LOAC are also present in OAC and the organisation between these concepts is similar. However, this ontology is less expressive and is documented in English. It contains all the essential concepts to describe the reasoning of the rescuer in a simple geovisual environment. This simplified version of the OAC is a valid example of what could be a Semantic Data Model to which applying the CoViKoa framework.

Both versions of the ontology (OAC and LOAC) are integrated in the use of Semantic Web technologies and mobilise concepts and properties of high level ontologies such as GeoSPARQL and OWL-Time which were discussed in Chapter 3. This use ensures a better understanding of the concepts we define by adding the semantics of the elements they inherit. Finally, the use of these technologies allows us to benefit from the definition of complex concepts, related to the geospatial data we make use of (with concepts such as `geo:Feature` and `geo:Geometry`) and to the temporal characterization of some of the elements of the ontology (with properties such as `time:hasBeginning`, `time:hasEnd` and `time:hasInstantOrDuration`)

7.2.1 Essential elements of the alert processing domain

Figure 7.1 provides a synthetic view (without datatype properties) of the proposed ontology.

The first concept we define is naturally the *alert* (`loac:Alert`): it is the entire

[2]The acronym comes from its name in French: *Ontologie d'Alerte Choucas*.

process of searching for the victim, from the call of the caller until the end of the alert (the victim is found or the search is abandoned).

The caller is the person who raises the alert by calling the rescue team and provides information about the victim. The *victim* (`loac:Victim`) is the person to rescue, whose location is to be determined. Victim and caller can be the same person, or separate people who may or may not be together (e.g., when a family member calls about a missing relative). While other situations are possible (e.g., a single caller for multiple victims who departed together but got separated on their way), we are currently working on cases where the victims are together, resulting in a single alert.

A search *hypothesis* (`loac:Hypothesis`) corresponds, during an alert, to a set of clues considered in order to obtain a probable location area (both *clue* and *probable location area* are terms defined subsequently). An alert thus includes one or more hypotheses depending on the different sets of clues retained by the rescuer who processes it.

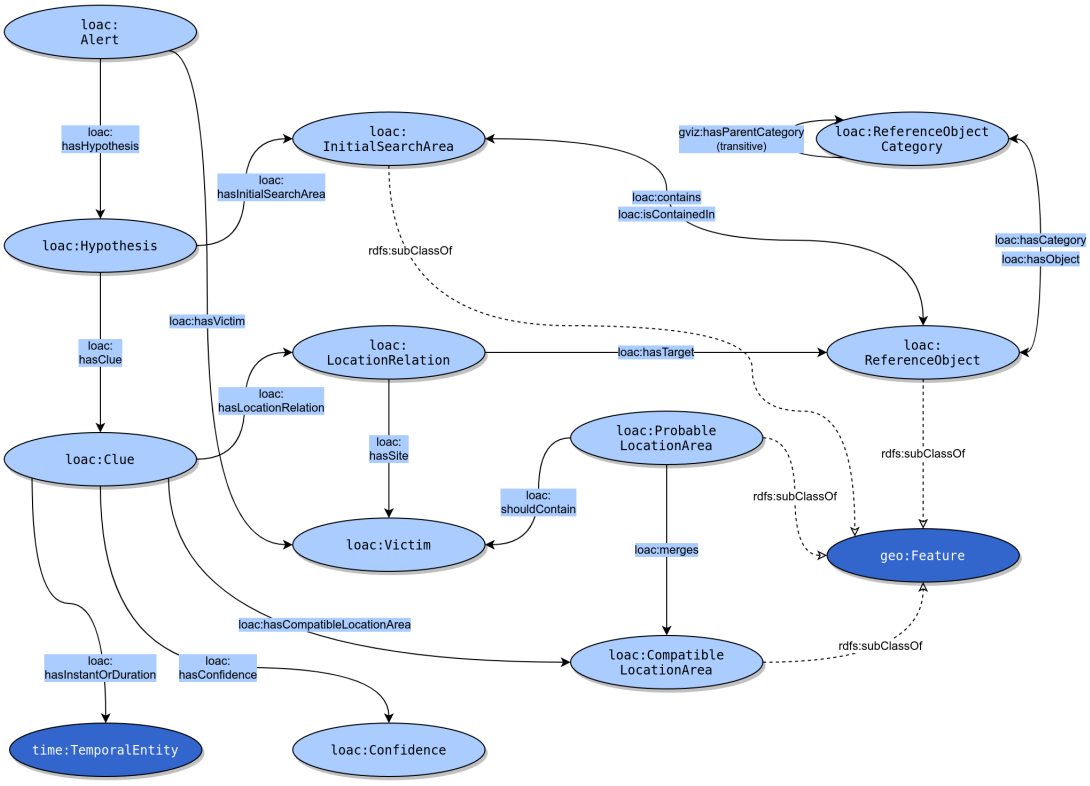

Figure 7.1: Overview of *Light Choucas Alert Ontology*.

The *Initial Search Area* (`loac:InitialSearchArea`) is the possibly large area, defined at the beginning of each new search hypothesis, in which the victim is guaranteed to be found. It corresponds to the translation of a first clue (see below): it can be, for

example, the whole mountain range if the victim is "lost in the Chartreuse[3]", or a radius of several dozen kilometers around his or her last known point.

A *clue* (`loac:Clue`) corresponds to a fragment of speech from the caller (who is talking to the rescue team) and describes the current or past position of the victim. This fragment of speech constitutes a location element including spatial but also temporal information. A *clue* can be described by a time or a duration of validity and ordering properties (next and/or previous *clue* - not shown on Figure 7.1) allow to describe the chronology between location elements, especially useful when a precise temporal marking is not possible. The *confidence* (`loac:Confidence`) given to a *clue* is also modelled as a confidence value among three modalities: low, medium or high, that we define as individuals. The rescuer can thus modulate the weight of a *clue* as needed (this responds to situations where what the caller says seems implausible to the rescuer, for example).

A *clue* implies the definition of a *location relation* (`loac:LocationRelation`) between a site (the victim, through the property `loac:hasSite`) and a target (one or more reference objects, through the property `loac:hasTarget`). We define here two subclasses to the concept of location relation: `loac:ProximityRelation` and `loac:VisibilityRelation` (not shown on Figure 7.1). This is one of the essential differences with the full version of OAC, in which location relations are formalised in a more complex way particularly in order to be compatible with the work of the project's partners (such as the work of Bunel, 2021 which is particularly aimed at qualifying these relationships).

As such, a *clue* describing the location of the victim who would have said, with certainty (`loac:HighConfidence`, an instance of `loac:Confidence`), that he or she was "near the XYZ road" (property `loac:naturalLanguage`) at the time of the call (property `loac:hasInstantOrDuration`) could be expressed as follows (Listing 7.1).

```
:exampleClue1 a loac:Clue ;
    loac:hasConfidence loac:HighConfidence ;
    loac:naturalLanguage "near the XYZ road" ;
    loac:hasInstantOrDuration [ a time:Instant ;
        time:inXSDDateTimeStamp "2004-04-12T13:22:00Z"^^xsd:dateTimeStamp
      ] ;
    loac:hasCompatibleLocationArea :exampleCla1 ;
    loac:hasLocationRelation [ a loac:ProximityRelation ;
        loac:hasSite :exampleVictim ;
        loac:hasTarget loac:obj-c130-4a67-8733-2b682c518f9c ; # the XYZ road
      ] .
```

Listing 7.1: Example of an instantiated `loac:Clue`.

A *clue* is intended to be translated into a corresponding location zone. The point is to move from a conceptual representation of location information to the geometric representation of what it expresses. We are formalising here only one type of location zone, the *Compatible Location Area* (`loac:CompatibleLocationArea`, CLA) which corresponds to *clues* expressed affirmatively ("being right by a lake" as shown by the drawn sketchup of Figure 7.2 and the green ellipse around the lake for *Clue*

[3]The Chartreuse Mountains are a mountain range in the Alps.

#1). Clues expressed in the negative ("not seeing a lake") are not taken into account in the approach presented here because of the difficulties in interpreting them and calculating their corresponding area.

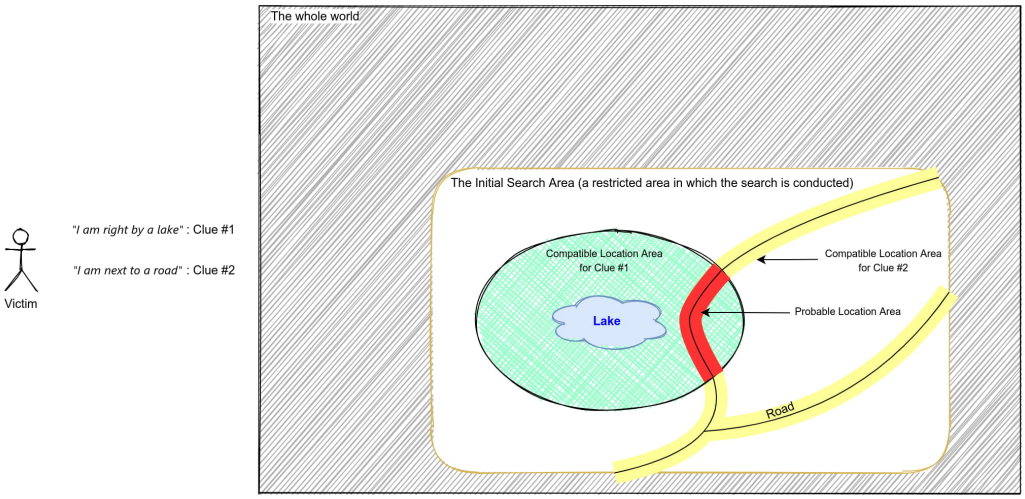

Figure 7.2: Example of *clues* transformed into *Compatible Location Areas* and allowing the calculation of a *Probable Location Area*.

Having a set of *Compatible Location Areas* created from *clues*, it is then possible to combine them to obtain the *Probable Location Area* (`loac:ProbableLocationArea`, PLA) of the victim. This is the area in which the victim is supposed to be located according to the *clues* provided (in red on Figure 7.2 is depicted the *Probable Location Area* considering *Clue #1* and *Clue #2*). As such, merging the *Compatible Location Areas* means determining the portion of the territory in which, by combining the different *clues*, the victim must be located. This combination can, for example, correspond to the geometric intersection of the different CLAs if they are characterised by vector geometries. We adopt this approach in this chapter. This *Probable Location Area* may be composed of one or more disjointed portions of territories that must be visually rendered to the rescuer.

Because we have chosen not to include formal constraints (e.g. as OWL restrictions) in LOAC, making it a lightweight ontology, the model is valid in both RDFS-Plus non-standard profile and in OWL 2 RL standard profile. However, we provide more details on how this model can and should be instantiated, notably through the creation of SHACL shapes which are presented in the following subsection.

7.2.2 Specifying constraints with SHACL

As previously announced, the LOAC ontology does not contain any formal constraint and only defines a few properties with special semantics (*transitive*, *inverse of*). Nevertheless, to ensure the validity of the knowledge embedded in the data

model, we propose SHACL shapes for validating LOAC data. These shapes contains useful informations about what is expected to be a well-formed data graph using LOAC.

We specify ten constraints which take up the elements evoked during the course of the concepts presentation. They are provided in the form of an RDF file to build the SHACL shape graph for validation with the SHACL engine. These constraints are the following:

- Each instance of `loac:Alert` has to be linked to one and only one `loac:Victim`.
- Each instance of `loac:Alert` has to be linked to one or more `loac:Hypothesis`.
- Each instance of `loac:Hypothesis` has to be linked to one or more `loac:Clue`.
- Each instance of `loac:Hypothesis` has to be linked to one and only one `loac:InitialSearchArea`.
- Each instance of `loac:Hypothesis` has to be linked to one `loac:ProbableLocationArea` if there are two or more `loac:Clue`s linked to this `loac:Hypothesis`.
- Each instance of `loac:Clue` has to be linked to one and only one instance of `loac:LocationRelation` (or an instance of one of its subclasses).
- Each instance of `loac:Clue` has to be linked to one and only one instance of `loac:CompatibleLocationArea`.
- Each instance of `loac:LocationRelation` has to be linked to one and only one site which is an instance of `loac:Victim`.
- Each instance of `loac:LocationRelation` has to be linked to one or more target which is/are instance of `loac:ReferenceObject`.
- Each instance of `loac:ReferenceObject` has to be linked to one and only one `loac:ReferenceObjectCategory`.

7.2.3 Populating the hierarchy of reference object categories

We have seen (Figure 7.1) that LOAC includes the `loac:ReferenceObject` class (subclass of `geo:Feature`) and the `loac:ReferenceObjectCategory` class. The former allows us to describe spatial entities (possibly qualified by a set of properties of their own) which are linked to their category (through the property `loac:hasCategory` and its inverse property `loac:hasObject`) formalised through the latter.

We also define a transitive property (property `loac:hasParentCategory`) allowing each category to be linked to a parent category, thus allowing to form a hierarchy of individuals that can describe reference objects of the territory as present in reference databases (OpenStreetMap, IGN BD TOPO, etc.).

Since the ontology is not provided with the data corresponding to these classes, we first propose to define a hierarchy of categories (Figure 7.3) in the form of individuals of `loac:ReferenceObjectCategory` linked by the property `loac:hasParentCategory`. This hierarchy is an extract of the one used in the GASPAR geovisualisation tool (Viry and Villanova-Oliver, 2020) that we presented in Chapter 2.

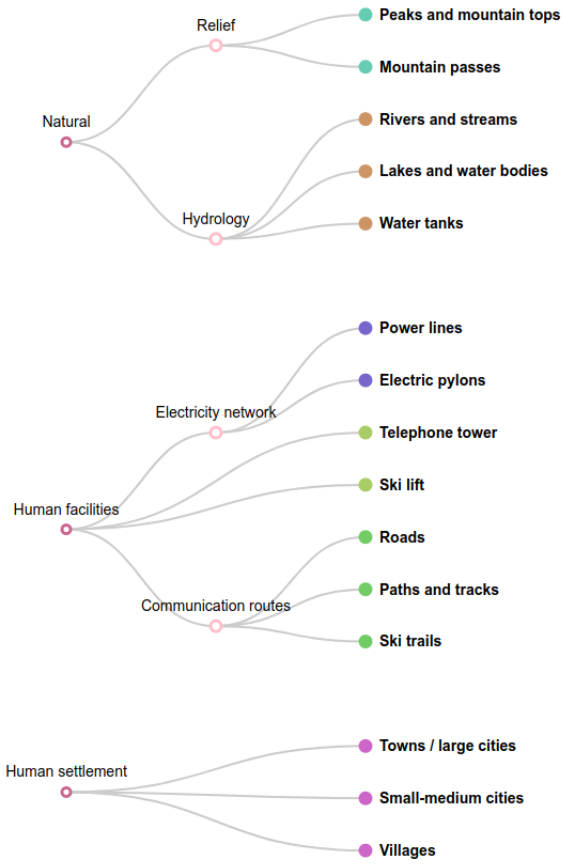

Figure 7.3: Hierarchy of reference objects.

Although we would have preferred to use an existing Web-based triplestore containing OpenStreetMap data, at the time of writing this manuscript we have not found any that expose such data for our study area in the French Alps[4].

We therefore instantiate 103106 `loac:ReferenceObjects` linked to these categories (through the `loac:hasCategory` property) from OpenStreetMap data and store it in a triplestore distinct from CoViKoa, a Fuseki triplestore with GeoSPARQL capabilities enabled, and expose it in our network. It is thus available for the case study we present in this chapter. Indeed, this triplestore will be queried during the processing of the case study, as shown in section 7.4.2, in order to integrate into the SDM only the objects located in the Initial Search Area.

7.3 Writing the core of the derivation model for LOAC

7.3.1 Framing the situation

The actual processing of an alert by a rescuer requires the implementation of a user interface combining the possibility to define various elements (such as ISA and *clue*) by the rescuer, different calls to the appropriate services (integration of reference data, calculation of CLAs and combination of CLAs into a PLA) and the definition of other interface components (such as a search engine among *reference objects* and a tool for tagging part of the victim's speech). Some of the elements necessary to acquire and process *clues* in a geovisualisation application have been discussed in Viry et al. (2019) and in Viry and Villanova-Oliver (2020) while others are dealt with by partners of the CHOUCAS project (in particular the computation of CLAs from the *clues* and their combination into a PLA, cf. Bunel, 2021).

It is worth noting that these elements are outside the scope of the CoViKoa framework presented here which focuses on graphic rendering and interactions in a geovisualisation interface. We therefore consider an existing instantiation of LOAC, that we assume it depicts a fictitious alert of which we consider several states to illustrate the approach (Figure 7.4).

We more especially consider to be at a given state in the processing of the alert that can be entirely played with CoViKoa from the Derivation Model, meaning here that the display can be anticipated as it does not depend on actions from the user in the interface (such as the creation of new *clues*) nor on a computation service (such as the one allowing to transform a *clue* into a *Compatible Location Area*).

The alert presented here corresponds to the search for a victim who cannot be precisely located by GNSS and describes his or her position using two elements. The

[4]When we did our tests, the SPARQL endpoint of LinkedGeoData was down (cf. http://linkedgeodata.org/docs/updates.html) but it would be interesting to do our case studies with data from LinkedGeoData in order to really benefit from geospatial Linked Data.

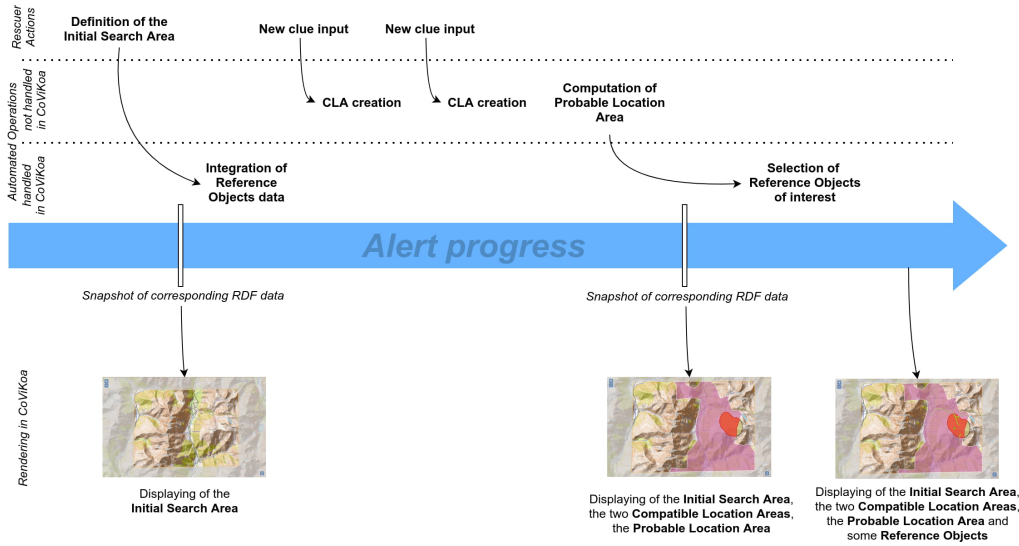

Figure 7.4: Temporal sequence of an alert.

first one corresponds to the speech fragment "I left Chantepérier towards a mountain peak 4 hours ago". The second corresponds to the speech fragment "I see a coarse gravel hiking trail". According to this set of information, the rescuer defines an *Initial Search Area* largely encompassing the peaks surrounding the starting locality and then enters the two *clues*, allowing the calculation of the two respective CLAs and the calculation of the PLA.

The data corresponding to this alert are thus exported in RDF at the two moments that interest us: after the definition of the ISA by the rescuer (but before the integration of the reference data, in order to be able to show how CoViKoa allows it), and after the calculation of the PLA corresponding to the two *clues* captured (Figure 7.4).

In the following sections, we show how CoViKoa can exploit the semantics of LOAC to represent its data in order to relieve the final user from the task of choosing the appropriate representations.

We provide portrayals for each of the concepts of LOAC which can be spatialised (these are the 4 concepts which are subclasses of `geo:Feature` and which are schematised on Figure 7.2). One of the challenges regarding the portrayals to be proposed for LOAC is that we want to draw attention to some elements of the map that may be useful for the rescuer at a given moment. This could indeed be an important help in the reasoning process performed by the rescuer during a geovisual analysis from a situational map. We will thus see how to automate the fact of drawing the user's attention to relevant elements of the map via the specification of ad-hoc statements in the Derivation Model that encode the principles we want to be applied. Over the next three sections, we define portrayals that can be seen as an incremental improvement

of the support offered for the geovisual reasoning. This enables a gradual discovery of the mechanisms offered by CoViKoa while seeking to improve the portrayal proposals made to the rescuer.

7.3.2 Taking advantage of the semantics encoded in the SDM

We first specify for which concepts of LOAC a graphical representation must be obtained. In our case all the concepts that are spatial entities (subclasses of `geo:Feature`) are concerned. Other concepts of the ontology (such as the concepts of *alert*, or *clue*), although useful for knowledge management, do not need a graphical display. As presented in Section 6.2.1, we create a subclass of `gviz:GeoVisualIntermediateRepresentation` for each of them and we define a `gviz:Map2dComponent` and a `gviz:GeoVisualApplication` (Listing 7.2).

We illustrate here the specificity seen in Section 6.2.1 concerning the early filtering of entities, at the application level. This allows us to exploit the specific structure of the data described with LOAC: each alert is linked to at least one hypothesis, and possibly several. However, at a given moment, we only wish to visualise the data linked to a specific hypothesis. This filtering is done by specifying, on the instance level of the geovisualisation application, the individual (`loac:hypo1`, line 35, a specific `loac:Hypothesis`) to which we are referring, and by specifying on each subclass of `gviz:GeoVisualIntermediateRepresentation` the path between the `loac:Hypothesis` class and the class that is derived via the annotation property `cvkd:pathFromGVA`.

```
1  # TBox
2  :AppChoucasHypo a owl:Class ;
3      rdfs:subClassOf gviz:GeoVisualApplication ;
4      cvkd:represents loac:Hypothesis .
5
6  :InitialSearchAreaIR a owl:Class ;
7      rdfs:subClassOf gviz:GeoVisualIntermediateRepresentation ;
8      cvkd:represents loac:InitialSearchArea ;
9      cvkd:pathFromGVA loac:hasInitialSearchArea .
10
11 :ReferenceObjectIR a owl:Class ;
12     rdfs:subClassOf gviz:GeoVisualIntermediateRepresentation ;
13     cvkd:represents loac:ReferenceObject ;
14     cvkd:pathFromGVA (loac:hasInitialSearchArea loac:contains) .
15
16 :CompatibleLocationAreaIR a owl:Class ;
17     rdfs:subClassOf gviz:GeoVisualIntermediateRepresentation ;
18     cvkd:represents loac:CompatibleLocationArea ;
19     cvkd:pathFromGVA (loac:hasClue loac:hasCompatibleLocationArea) .
20
21 :ProbableLocationAreaIR a owl:Class ;
22     rdfs:subClassOf gviz:GeoVisualIntermediateRepresentation ;
23     cvkd:represents loac:ProbableLocationArea ;
24     cvkd:pathFromGVA loac:hasProbableLocationArea .
25
26
27 # ABox
```

```
:HypothesisMap a gviz:Map2dComponent ;
    cvkc:widthPx 980 ;
    cvkc:heightPx 600 ;
    cvkc:hasBaseMap cvkc:openTopoMapBaseMap ;
    cvkd:hasDefaultExtent loac:InitialSearchArea .

:AppHypothesisChoucas a :AppChoucasHypo ;
    gviz:represents loac:hypo1 ;
    gviz:hasGeoVisualComponent :HypothesisMap .
```

Listing 7.2: Definition of SDM classes to be derived and of the application that visualise them.

These declarations will allow our framework to produce the *derivation rules* (as seen in Section 6.2.2). As such, these rules will only create IRs for individuals in the four classes mentioned (`loac:InitialSearchArea`, `loac:ReferenceObject`, `loac:CompatibleLocationArea` and `loac:ProbableLocationArea`) that are related to the hypothesis (`loac:hypo1`) to be displayed. It is then necessary to define the portrayals that correspond to each of these classes: these declarations will allow our framework to produce the *enrichment rules* (as seen in Section 6.2.2) that will allow us to link the materialisation of each individual to its symbolizer.

Initial Search Area, *Probable Location Area* and *Compatible Location Area* are shown immediately, from Section 7.3.3, and *reference objects* are covered in Section 7.4.3.

7.3.3 Representation of the initial Search Area with a transformation of its geometry

The first concept we wish to represent is the Initial Search Area (ISA), considering a base map layer is provided in Listing 7.2. Each hypothesis is linked to only one ISA that corresponds to the part of the territory in which the search for the victim is taking place. As such, it seems interesting, from a geovisual and cognitive point of view, not to materialise the interior of the area (which would prevent the elements it contains from being properly perceived) but, conversely, to provide an attenuate representation for the entire territory not belonging to the ISA. The expected result is shown on Figure 7.5.

From the Derivation Model side, the definition of a portrayal for the ISA allows us to illustrate the use of a `cvkd:TransformOperation`, which transforms the geometry of the ISA into the geometry to be represented: the difference between the ISA and the rest of the territory (Listing 7.3). During the loading step of CoViKoa, this transform operation will be rewritten into its SPARQL form and injected into the *enrichment rule* generated for the portrayal rule corresponding to the ISA. We also define in Listing 7.3 the polygon symbolizer to be used by the entities that will have this portrayal applied to them.

Since we do not link this portrayal to a scale, it is valid at all levels of visualisation. We will see later that we also apply a specific portrayal to this ISA when it is viewed at a high zoom level.

```
:PortrayalISA a gviz:Portrayal ;
    cvkd:denotesGVR :InitialSearchAreaIR ;
    gviz:displayIndex 1000 ;
    gviz:hasPortrayalRule [
        gviz:hasSymbol [ a symb:Symbol ;
            dct:title "Initial Search Area" ;
            symblzr:hasSymbolizer :SymbolizerISA ;
          ] ;
      ] ;
    cvkd:hasTransformOperation [
        geof:difference (
            [cvkd:value "POLYGON ((-180 -90, -180 90, 180 90, 180 -90, -180 -90))"^^
    geo:wktLiteral]
            [cvkd:variable "?thisGeometry"]
          )
      ] .

:SymbolizerISA a symblzr:PolygonSymbolizer ;
    graphic:hasStroke [ a graphic:Stroke ;
        graphic:strokeColor "rgb(255,165,0)"^^graphic:cssColorLiteral ;
      ] ;
    graphic:hasFill [ a graphic:Fill ;
        graphic:fillColor "rgba(211, 211, 211, 0.7)"^^graphic:cssColorLiteral ;
      ] .
```

Listing 7.3: Definition of the portrayal and the symbolizer for the Initial Search Area.

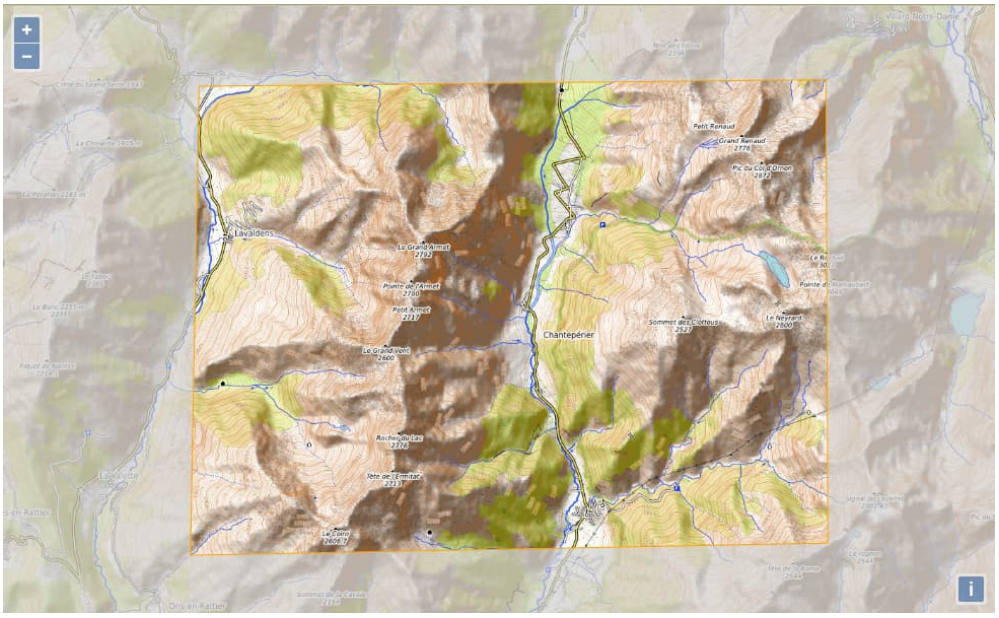

Figure 7.5: Portrayal for the *Initial Search Area*.

7.3.4 Representation of the Probable Location Area

We also propose a portrayal for the *Probable Location Area* (PLA). This representation is quite simple and does not currently use any specific features of CoViKoa. The definition of the symbolizer `:SymbolizerPLA` (a `symblzr:PolygonSymbolizer` with a *red* colouring) is omitted here for the sake of brevity but the corresponding depiction can be seen in Figure 7.6.

```
:PortrayalPLA a gviz:Portrayal ;
    cvkd:denotesGVR :ProbableLocationAreaIR ;
    gviz:displayIndex 2 ;
    gviz:hasPortrayalRule [
        gviz:hasSymbol [ a symb:Symbol ;
            dct:title "Probable Location Area" ;
            symblzr:hasSymbolizer :SymbolizerPLA ;
          ] ;
      ] .
```

Listing 7.4: Definition of the portrayal and the symbolizer for the Probable Location Area.

7.3.5 Representation of Compatible Location Areas with property constraints

The last element that we present here is the *Compatible Location Area* (CLA) which corresponds to the spatialisation of a *clue* given by the victim.

We propose different way of representing a CLA depending on the type of *clue* from which it was created but also depending on the type of reference object that was mobilised in the *clue*. We only show here their writing for two categories of reference objects (Listing 7.5), namely `loac:VILLAGE` and `loac:PATHWAY`. This allows us to illustrate :

- the writing of complex property paths (as seen in Section 6.2.1), see for instance line 3 (`sh:inversePath`) in Listing 7.5,
- the use of property constraints using a logical articulation (as seen in Section 5.3.2), see for instance line 10 (`cvkd:and`) in Listing 7.6.

In addition, we can take advantage of the definition of property constraints as named individuals and thus reuse them in several portrayal rules.

```
:constraintVillage
    a cvkd:PropertyConstraint ;
    cvkd:propertyPath ([sh:inversePath loac:hasCompatibleLocationArea] loac:
    hasLocationRelation loac:hasTarget loac:hasCategory) ;
    cvkd:valueOrObjectIsEqualTo loac:VILLAGE .

:constraintPathway
    a cvkd:PropertyConstraint ;
```

```
    cvkd:propertyPath ([sh:inversePath loac:hasCompatibleLocationArea] loac:
    hasLocationRelation loac:hasTarget loac:hasCategory) ;
    cvkd:valueOrObjectIsEqualTo loac:PATHWAY .

:constraintRelationProximity
    a cvkd:PropertyConstraint ;
    cvkd:propertyPath ([sh:inversePath loac:hasCompatibleLocationArea] loac:
    hasLocationRelation) ;
    cvkd:objectOfType loac:ProximityRelation .

:constraintRelationVisibility
    a cvkd:PropertyConstraint ;
    cvkd:propertyPath ([sh:inversePath loac:hasCompatibleLocationArea] loac:
    hasLocationRelation) ;
    cvkd:objectOfType loac:VisibilityRelation .
```

Listing 7.5: Definitions of the property constraints to be reused in the various portrayal rules relating to Compatible Location Areas.

It is then possible to define a portrayal with the different portrayal rules corresponding to the different cases to be processed (Listing 7.6). The definition of symbolizers is still omitted for the sake of brevity but the corresponding depictions can be seen in Figure 7.6: the type of location relation is represented by the colour of the thick border used for the CLA (*black* for `loac:VisibilityRelation` and *white* for `loac:ProximityRelation`) while the colour of the interior of the area depends on the type of reference object (here a *reddish purple* colour for `loac:VILLAGE` and a *pale gold* colour for `loac:PATHWAY`).

```
:PortrayalCLA a gviz:Portrayal ;
    cvkd:denotesGVR :CompatibleLocationAreaIR ;
    gviz:displayIndex 3 ;
    gviz:hasPortrayalRule [
        gviz:hasSymbol [ a symb:Symbol ;
            dct:title "Compatible Location Area (Village / Visibility)" ;
            symblzr:hasSymbolizer :SymbCLA-Village-Visibility ;
          ] ;
        cvkd:hasPropertyConstraint [ a cvkd:PropertyConstraint ;
            cvkd:and (:constraintRelationVisibility :constraintVillage)
          ] ;
      ] ;
    gviz:hasPortrayalRule [
        gviz:hasSymbol [ a symb:Symbol ;
            dct:title "Compatible Location Area (Village / Proximity)" ;
            symblzr:hasSymbolizer :SymbCLA-Village-Proximity ;
          ] ;
        cvkd:hasPropertyConstraint [ a cvkd:PropertyConstraint ;
            cvkd:and (:constraintRelationProximity :constraintVillage)
          ] ;
      ] ;
    gviz:hasPortrayalRule [
        gviz:hasSymbol [ a symb:Symbol ;
            dct:title "Compatible Location Area (Pathway / Visibility)" ;
            symblzr:hasSymbolizer :SymbCLA-Pathway-Visibility ;
          ] ;
        cvkd:hasPropertyConstraint [ a cvkd:PropertyConstraint ;
            cvkd:and (:constraintRelationVisibility :constraintPathway)
```

```
        ] ;
    ] ;
  gviz:hasPortrayalRule [
      gviz:hasSymbol [ a symb:Symbol ;
          dct:title "Compatible Location Area (Pathway / Proximity)" ;
          symblzr:hasSymbolizer :SymbCLA-Pathway-Proximity ;
        ] ;
      cvkd:hasPropertyConstraint [ a cvkd:PropertyConstraint ;
          cvkd:and (:constraintRelationProximity :constraintPathway)
        ] ;
    ] .
```

Listing 7.6: Definition of the portrayal for Compatible Location Areas.

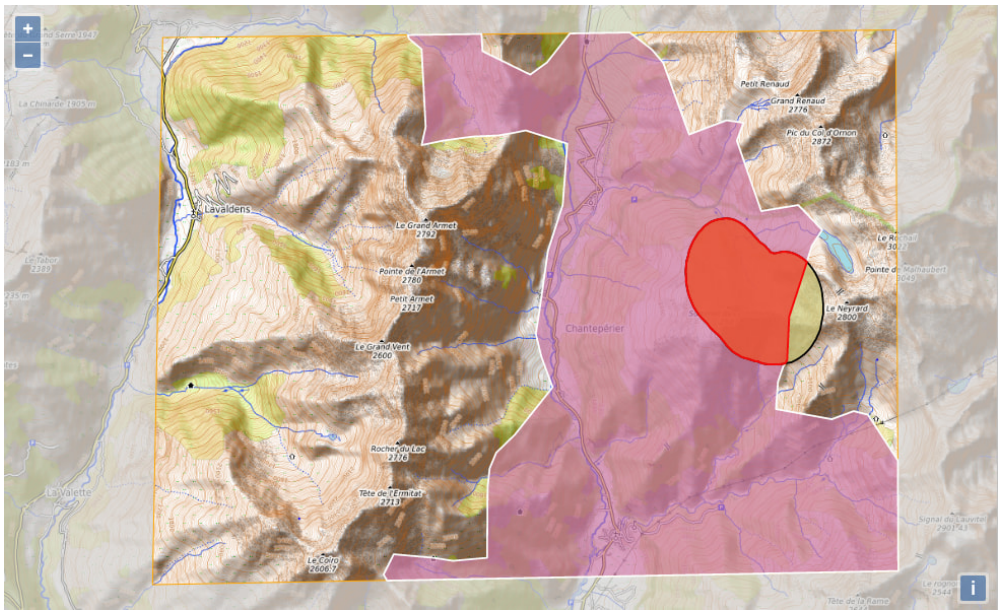

Figure 7.6: Example of data portrayal with one *Initial Search Area*, two *Compatibles Location Areas* and the resulting *Probable Location Area*.

The portrayals proposed so far allow the representation of three of the four targeted concepts (ISA, CLA and PLA). Their specification call upon some of CoViKoa's advanced functionalities (filtering at the application level, transformation of geometries with regard to a constant value, property constraints to select the entities to be represented).

However, we believe that improvements can be made, in particular in taking into account the zoom level at which the map is displayed. In addition, we would like to be able to express a more complex selection rule to highlight specific *reference objects* (`loac:ReferenceObject`, the fourth spatial concept in LOAC) that might be useful to the rescuer. This is what is shown in what follows.

7.4 Going further in the portrayals that can be defined

7.4.1 Portrayals in accordance with the zoom level

This section presents the possibilities offered by CoViKoa to define different portrayal rules according to the zoom level. We have seen (Section 5.3.3) that this can be done using the *Scale* vocabulary.

In the CHOUCAS project, one of the most important elements is the *Probable Location Area* (PLA), the area in which the victim is believed to be located by combining the various *clues* provided by the caller. On the one hand, we assume that it is necessary to draw attention to this area when the map is displayed at a low zoom level, so that the rescuer perceives where the different fragments of the PLA are located in the territory, how many of them there are, etc. On the other hand, at a high zoom level (which may no longer correspond to an overall view as before, but to the need to visually explore in detail one of the fragments of the PLA), we assume that it is necessary not to mask the interesting elements of the map that are in the PLA.

We implement this representation logic using the *Scale* vocabulary (Listing 7.7). We instantiate two scales which correspond to a low zoom level and a high zoom level. The advantage of the formalisation adopted is that it is possible to reuse these two scales across multiple portrayals and portrayals rules.

```
:LowZoomScale a scale:Scale ;
    scale:hasMinZoomLevel "0.0"^^xsd:float ;
    scale:hasMaxZoomLevel "13.0"^^xsd:float .

:HighZoomScale a scale:Scale ;
    scale:hasMinZoomLevel "13.0"^^xsd:float ;
    scale:hasMaxZoomLevel "19.0"^^xsd:float .
```

Listing 7.7: Definition of two `scale:Scale`s.

We therefore represent the PLA to be completely transparent and surrounded by a thick red border at a high zoom level and restrict the validity of the portrayal previously defined for the PLA to the low zoom level (Listing 7.8)

```
:PortrayalPLAHighScale a gviz:Portrayal ;
    cvkd:denotesGVR :ProbableLocationAreaIR ;
    gviz:displayIndex 2 ;
    scale:hasScale :HighZoomScale ;
    gviz:hasPortrayalRule [
        gviz:hasSymbol [ a symb:Symbol ;
            dct:title "Probable Location Area" ;
            symblzr:hasSymbolizer :SymbolizerHighScalePLA ;
          ] ;
      ] .

:PortrayalPLA scale:hasScale :LowZoomScale .
```

Listing 7.8: Definition of the new portrayal for the Probable Location Area.

Representing the PLA in this way has several implications for the portrayals that have already been defined:

- it is necessary to compute the difference of each of the CLAs with the PLA so as not to mask the PLA,
- in order not to introduce any risk of confusion between the areas outside the CLAs and the PLA (now totally transparent), we need to colour this area the same colour as the outside of the ISA.

We define these new portrayals below and also restrict the scale of validity of the previously defined portrayal for the CLAs (Listing 7.9). These elements highlight the use of a Transform Operation with the *prelude binding* mechanism (property `cvkd:preludeBindings`, line 5 and line 25), allowing the use of a variable referring to graph data to transform the data. This is necessary to obtain the geometry to be represented for the CLAs.

```
1  :PortrayalISAHighZoom a gviz:Portrayal ;
2      cvkd:denotesGVR :InitialSearchAreaIR ;
3      gviz:displayIndex 1000 ;
4      cvkd:hasTransformOperation [
5          cvkd:preludeBinding """loac:hypo1 loac:hasProbableLocationArea [ geo:
       hasGeometry [ geo:asWKT ?geomPla ] ] ."""  ;
6          geof:difference (
7              [cvkd:variable "?thisGeometry"]
8              [cvkd:variable "?geomPla"]
9            )
10       ] ;
11     gviz:hasPortrayalRule [
12         gviz:hasSymbol [ a symb:Symbol ;
13             dct:title "Initial Search Area" ;
14             symblzr:hasSymbolizer :SymbolizerISAHighZoomScale ;
15           ] ;
16       ] ;
17     scale:hasScale :HighZoomScale .
18
19
20 :PortrayalCLAHighScale a gviz:Portrayal ;
21     cvkd:denotesGVR :CompatibleLocationAreaIR ;
22     gviz:displayIndex 3 ;
23     scale:hasScale :HighZoomScale ;
24     cvkd:hasTransformOperation [
25         cvkd:preludeBinding  """loac:hypo1 loac:hasProbableLocationArea [ geo:
       hasGeometry [ geo:asWKT ?geomPla ] ] ."""  ;
26         geof:difference (
27             [cvkd:variable "?thisGeometry"]
28             [cvkd:variable "?geomPla"]
29           )
30       ] ;
31     [... same portrayal rules as in PortrayalCLA...] .
32
33 :PortrayalCLA scale:hasScale :LowZoomScale .
```

Listing 7.9: Definition of the portrayals for the Initial Search Area and the Compatible Location Areas at a high zoom level.

Now that we have defined the portrayals adapted to a high level of zoom (adapted to the exploration of the different fragments of the PLA by the rescuer), we propose to highlight some data present in the PLA that could help the rescuer to relevantly interrogate the *victim* to create a new *clue*. The objective is to reduce the number of fragments of the PLA or its surface area by combining it with a new CLA. This requires reference data (which describes concrete elements of the territory, such as road, water and electricity networks) that will need to be integrated into the alert data graph. This graph indeed only includes at this stage data that are involved in *clues* elicitation.

7.4.2 Integration of new data

We show here a functionality of CoViKoa which was mentioned in Section 6.2.2 but not yet exemplified: the possibility for the user to define *data integration rules*. These rules consist in writing a SHACL rule (precisely a `sh:NodeShape` linked to a `sh:SPARQLRule`) containing a federated query and allowing to describe which data to integrate (Listing 7.10).

This rule allows using the properties of the Initial Search Area (its geometry and the fact that it is an ISA) to retrieve all the reference objects (from the aforementioned triplestore) it encompasses so that they can be used as reference during the alert. At the same time we also read the type of geometry (in the WKT representation of the geometry) and describe it with the appropriate class of the *Simple Feature* (this is done with the custom SPARQL function `cvkd:UnpackGeomType`, line 32).

```
1  :OSMRefObjectContainedInISA
2      a sh:NodeShape , cvkd:DataIntegrationRule ;
3      sh:rule [ a sh:SPARQLRule ;
4          sh:construct """
5  prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
6  prefix loac: <http://purl.org/loac#>
7  prefix gviz: <http://lig-tdcge.imag.fr/steamer/covikoa/geoviz#>
8  prefix geo: <http://www.opengis.net/ont/geosparql#>
9  prefix geof: <http://www.opengis.net/def/function/geosparql/>
10 prefix cvkd: <http://lig-tdcge.imag.fr/steamer/covikoa/derivation#>
11
12 CONSTRUCT {
13     ?ft a loac:ReferenceObject ;
14         loac:hasCategory ?cat ;
15         loac:isContainedIn ?zir ;
16         geo:hasGeometry [ a ?geomType ; geo:asWKT ?geomFtWkt] .
17     ?ft loac:osmName ?osmName .
18 } WHERE {
19     NOT EXISTS { ?ft loac:isContainedIn ?zir . }
20     ?alerteIR gviz:represents [
21         loac:hasInitialSearchArea ?zir ;
22     ] .
23     ?zir geo:hasGeometry [geo:asWKT ?geomZirWkt] .
24
25     SERVICE <http://localhost:3330/ds> {
26         ?ft a loac:ReferenceObject ;
```

```
            loac:hasCategory ?cat ;
            geo:hasGeometry [geo:asWKT ?geomFtWkt] .
        OPTIONAL { ?ft loac:osmName ?osmName . }
    }
    FILTER(geof:sfIntersects(?geomZirWkt, ?geomFtWkt)).
    BIND(cvkd:UnpackGeomType(?geomFtWkt) AS ?geomType).
}""" ;
      ] .

:InitialSearchAreaIR cvkd:hasDataIntegrationRule :OSMRefObjectContainedInISA ;
```

Listing 7.10: Definition of a `cvkd:DataIntegrationRule`.

7.4.3 Selection of useful information using SDM semantics and portrayals promoting the focus of attention

We have presented so far how to implement portrayals which exploit a selection of entities which can be expressed directly in RDF. Some cases, made possible by the sometimes strong semantics of SDM, are however not directly exploitable by the constraint mechanisms of CoViKoa.

In order to go further in the selection of the entities to be represented and in order not to limit the user of CoViKoa to existing property or spatial constraints, we propose the use of *SPARQL Filters* (`cvkd:SPARQLFilter`) which can be linked to a portrayal (through the property `cvkd:hasSPARQLFilter`).

We saw previously that the generated *enrichment rules* were qualified as *default enrichment rules*. Those using a SPARQLFilter and thus containing more complex logic are, on the contrary, qualified as *problem-specific enrichment rules* (Section 6.2.2).

In our case, we want to encode the selection of the *reference objects* located in the *Probable Location Area* and which do not belong to a *category of object* already used in the *clues* that enabled the constitution of this *Probable Location Area*. The translation of this selection logic into a SPARQL Filter is shown in Listing 7.11. The main expectation for this SPARQL Filter is that the individuals pulled by the query are bound to the variable `?somethingRepresentable`. In any case, this is an expectation that is checked during the validation of the Derivation Model which precedes the *loading step* of CoViKoa (Section 6.2.2).

```
:ConditionFilterRefObjectPLA a cvkd:SPARQLFilter ;
    rdfs:label "ObjectOfInterestInISA" ;
    rdf:value """
        SELECT DISTINCT ?somethingRepresentable WHERE {
          {
            SELECT
              (GROUP_CONCAT(?catTarget ; separator=\",\") as ?cats)
              (GROUP_CONCAT(?pcatTarget ; separator=\",\") as ?pcats)
            WHERE {
                ?probableLocArea loac:merges ?zone .
```

```
                ?clue loac:hasCompatibleLocationArea ?zone ;
                      loac:hasLocationRelation/loac:hasTarget ?ftTarget .
                ?ftTarget loac:hasCategory ?catTarget .
                ?catTarget loac:hasParentCategory ?pcatTarget .
            }
        }
        loac:hypo1 loac:hasProbableLocationArea ?probableLocArea .
        ?probableLocArea geo:hasGeometry [geo:asWKT ?geomProbableLocArea ].
        ?somethingRepresentable a loac:ReferenceObject ;
                                loac:hasCategory ?cat ;
        geo:hasGeometry [geo:asWKT ?geomObjet ] .
        ?cat loac:hasParentCategory ?pcat .
        FILTER(!CONTAINS(?cats, STR(?cat)) && !CONTAINS(?pcats, STR(?pcat))) .
        FILTER(geof:sfIntersects(?geomProbableLocArea, ?geomObjet)) .
    }
""" .
```

Listing 7.11: Definition of a `cvkd:SPARQLFilter`.

Note that no prefix definition is included at the start of the query. This is possible thanks to two mechanisms:

- the fact that this filter is intended to be injected into a SHACL rule already declaring the prefixes (through the preferred prefix defined in the ontology in question if applicable) of all the ontologies used by CoViKoa as well as the `geo:` and `geof:` prefixes which correspond to the GeoSPARQL vocabulary and GeoSPARQL functions respectively,
- the fact that it is possible to declare, at the level of the declaration of the OWL ontology (Listing 7.12) represented by the Derivation Model (this declaration is optional apart from the mechanism we are describing here), prefixes to be injected into the SHACL rules containing the SPARQL Filter written by the user.

Here we use vocabulary and mechanisms borrowed from SHACL, as it was the case previously with the writing of property paths or the syntax of transform operations.

```
:
    a owl:Ontology ;
    sh:declare [
        sh:prefix "loac" ;
        sh:namespace "http://purl.org/loac#"^^xsd:anyURI ;
      ] .
```

Listing 7.12: Definition of the Derivation Model as an OWL ontology and attaching prefixes to it for later injection into SPARQL Filters.

It is now possible to declare a portrayal for entities that match this selection logic (Listing 7.13).

```
:PortrayalReferenceFeatureHighlightInsidePLA a gviz:Portrayal ;
    cvkd:denotesGVR :ReferenceObjectIR ;
    gviz:displayIndex 1 ;
```

```
4       cvkd:multipleGeometryType "true"^^xsd:boolean ;
5       cvkd:hasTransformOperation [
6           cvkd:preludeBinding """loac:hypo1 loac:hasProbableLocationArea [ geo:
        hasGeometry [ geo:asWKT ?geomPla ] ] ."""" ;
7           geof:intersection (
8               [cvkd:variable "?thisGeometry"]
9               [cvkd:variable "?geomPla"]
10          )
11      ] ;
12      gviz:hasPortrayalRule [
13          gviz:hasSymbol [ a symb:Symbol ;
14              symblzr:hasSymbolizer :SymbolizerReferenceFeatureHighlightPt , :
        SymbolizerReferenceFeatureHighlightLine , :
        SymbolizerReferenceFeatureHighlightPolygon ;
15          ] ;
16      ] ;
17      cvkd:hasSPARQLFilter :ConditionFilterRefObjectPLA .
```

Listing 7.13: Definition of a `gviz:Portrayal` dedicated to highlighting specific entities inside the PLA.

We also show in Listing 7.13 the possibility of providing multiple symbolizers on the `symblzr:hasSymbolizer` property in conjunction with the use of the property `cvkd:multipleGeometryType` (line 4) when defining the portrayal. This is used as a shortcut to define a portrayal that depends on the type of geometry, e.g. in a case like ours when we do not know beforehand the geometric type of the objects to be depicted. Internally, in the *enrichment rule*, a pattern matching is injected in order to match between the geometry type of the encountered individual and the type of symbolizer (we had formalised, cf. Section 5.3.1, for which geometry type each type of symbolizer was intended with the help of an annotation property).

As stated in Listing 7.13, we calculate here the intersection between the reference objects to be displayed and the PLA. This allows us to represent with a particular symbolizer the part of the reference objects located in the PLA, but we think it is also necessary to represent the remaining part of these objects, in a similar but less salient representation. This is what we do by reusing the SPARQL filter defined previously (Listing 7.11) in a new portrayal intended to display this subset of the reference objects (Listing 7.14). It is here not necessary to define a symbolizer for the point (as there can be no remaining part of the intersection of a point and the PLA).

```
1   :PortrayalReferenceFeatureHighlightOutsidePLA a gviz:Portrayal ;
2       cvkd:denotesGVR :ReferenceObjectIR ;
3       gviz:displayIndex 1 ;
4       cvkd:multipleGeometryType "true"^^xsd:boolean ;
5       cvkd:hasTransformOperation [
6           cvkd:preludeBinding """loac:hypo1 loac:hasProbableLocationArea [ geo:
        hasGeometry [ geo:asWKT ?geomPla ] ] ."""" ;
7           geof:difference (
8               [cvkd:variable "?thisGeometry"]
9               [cvkd:variable "?geomPla"]
10          )
11      ] ;
```

```
12      gviz:hasPortrayalRule [
13          gviz:hasSymbol [ a symb:Symbol ;
14              symblzr:hasSymbolizer :SymbolizerReferenceFeatureHighlightLineOutside ,
        :SymbolizerReferenceFeatureHighlightPolygonOutside ;
15          ] ;
16        ] ;
17      cvkd:hasSPARQLFilter :ConditionFilterRefObjectPLA .
```

Listing 7.14: Definition of a `gviz:Portrayal` for the part outside the PLA for the entities that intersect it.

We present these statements here as two separate portrayals, to accompany the narrative, but note that it would have been possible to define only one portrayal linked to two portrayal rules, each attached to its transform operation.

The result obtained at a low zoom level is shown in Figure 7.7. It shows the portrayals corresponding to the four concepts to be depicted: the *Initial Search Area* (materialised by the colouring of its difference with the rest of the world, in grey), two *Compatible Location Areas* (in reddish purple and pale gold), a *Probable Location Area* (in bright red) and three *reference objects* of particular interest to the rescuer if he or she wishes to ask the victim about his or her immediate environment in order to create a new *clue* (in green, the part inside the PLA being represented differently from the part outside).

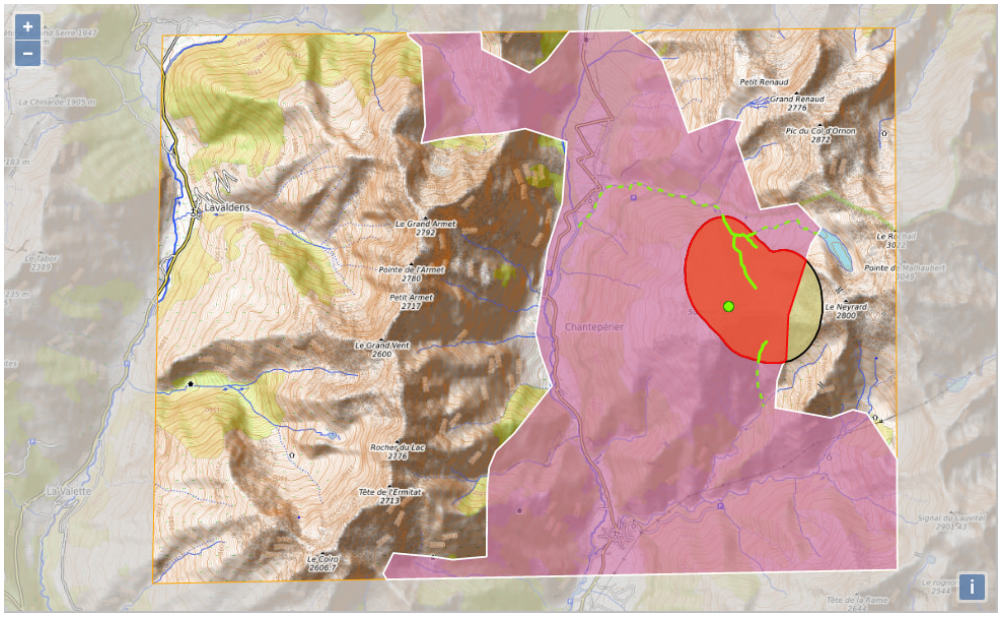

Figure 7.7: Example of data portrayal with *reference objects* of interest highlighted, at a low zoom level.

At a high zoom level, in Figure 7.8, we can see that the area corresponding to the *Probable Location Area* is no longer filled in, but only bordered by a thick red line (to recall the colour of its interior at a low zoom level). To allow for this, only

the difference between the *Compatible Location Areas* and the PLA is shown, in the same colour as before. It is also noted that, in order not to create confusion between the interior of the *Probable Location Area* and the non-interesting areas of the *Initial Search Area*, these areas are now coloured grey, like the exterior of the *Initial Search Area*. This representation allows a focus on the interior of the *Probable Location Area* and on the *reference objects* that are highlighted.

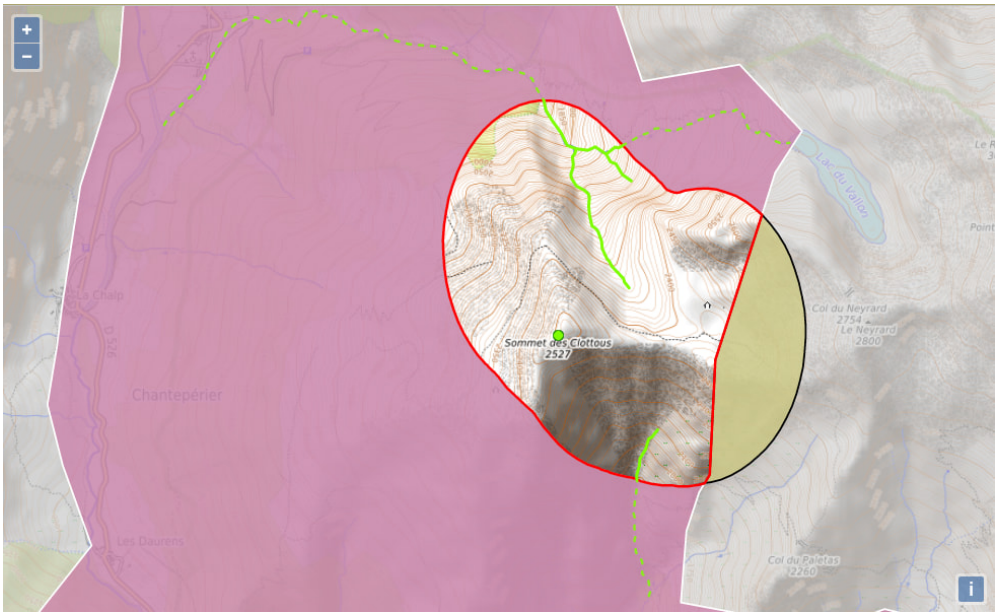

Figure 7.8: Example of data portrayal with *reference objects* of interest highlighted, at a high zoom level.

7.5 Going beyond the display of a single map

7.5.1 Definition of several components and their initial state

Covikoa-client is able to create all the various components described in *covikoa-geoviz* and this implies that it is able to create several maps if this is requested by the user.

We proposed (in Section 5.2.3) a vocabulary related to the initial context of visualisation, *covikoa-context*. This vocabulary is particularly useful for specifying the size and layout of components in the interface when several components are defined. In Listing 7.2 we have shown that we can use properties that allow the size of components to be specified. The declaration in Listing 7.15 states that components can be moved and resized by the user. The use of these properties is exclusive of defining a fixed size for the components. However, it is possible to define the order of appearance of the components in the window which will be divided by our layout

mechanism.

```
:HypothesisMap a gviz:Map2dComponent ;
    dct:title "Overview map" ;
    cvkc:order 1 ;
    cvkc:movable true ;
    cvkc:resizable true ;
    cvkc:hasBaseMap cvkc:openTopoMapBaseMap ;
    cvkd:hasDefaultExtent loac:InitialSearchArea .


:PLA-map a gviz:Map2dComponent ;
    dct:title "PLA map" ;
    cvkc:order 2 ;
    cvkd:position "right" ;
    cvkc:movable true ;
    cvkc:resizable true ;
    cvkc:hasBaseMap cvkc:openTopoMapBaseMap ;
    cvkd:hasDefaultExtent loac:InitialSearchArea ;
    gviz:hasCenterAndResolutionSynchronizedWith :HypothesisMap .

:terrain-map a gviz:Map3dComponent ;
    dct:title "Terrain view" ;

    cvkc:order 3 ;
    cvkc:position "bottom" ;
    cvkc:movable true ;
    cvkc:resizable true ;
    cvkc:hasBaseMap cvkc:arcgisOnlineWorldImagery ;
    cvkd:hasDefaultExtent loac:InitialSearchArea .

:identifyPopup a gviz:PopupComponent ;
    gviz:linkedTo :HypothesisMap .

:AppHypothesisChoucas a :AppChoucasHypo ;
    gviz:represents loac:hypo1 ;
    gviz:hasGeoVisualComponent :HypothesisMap ;
    gviz:hasGeoVisualComponent :PLA-map ;
    gviz:hasGeoVisualComponent :terrain-map ;
    gviz:hasGeoVisualComponent :identifyPopup .
```

Listing 7.15: Specification for the layout of the components.

In addition, we have defined in *covikoa-geoviz* several links that allow components to be linked together when they are declared. The existence of these links is exploited by *covikoa-client* and thus makes it possible to obtain complex behaviours in the application simply by defining them in RDF. This is for example the case of the `gviz:hasCenterAndResolutionSynchronized` link which can connect two `gviz:Map2dComponents` (as shown on Listing 7.15). When read by the application, the two maps will have their center and zoom level synchronised (the other case of synchronisation being the synchronisation of the map extent, which may be more relevant if the maps have quite different sizes).

This is also the case for the `gviz:linkedTo` property which allows the linking of `gviz:Map2dComponent` and `gviz:PopupComponent` as well as `gviz:Map2dComponent` and `gviz`

`:LegendComponent`: using this link allows to automatically benefit from the default behaviour of the popup (provided that an interaction uses it) and legend components (as shown on Listing 7.15 for the popup component - the interaction that uses it is defined in what follows and the result is shown on Figure 7.12).

7.5.2 Flexible multi-map display for a better understanding of the victim search situation

The possibilities offered by the creation of several components, and in particular several maps, depend on the context of use of the geovisualisation created. In our case we believe that a multi-map display can be useful in several situations in a victim search:

- using one map to represent the general context of the search (like a context map or a map inset in cartography) and one map to explore the PLA (Figure 7.9),
- combining one or more maps with a terrain view displaying the same layers but offering a satellite imagery as background (Figure 7.10),
- by using two synchronised views (map or terrain), showing different representations or the same representation on different background layers (Figure 7.11),
- using each map to display different fragments of the PLA.

We do not show the necessary declarations for the portrayals[5] of these figures which are almost the same as those shown in Section 7.3 and in Section 7.4, with the exception that each portrayal must now be explicitly linked to the map(s) it is intended for by using the `gviz:onComponent` property. Not linking a portrayal to a map has the effect of making it appear on all the defined maps.

The data presented in Figure 7.10 and Figure 7.11 also correspond to another alert. This *alert* is composed of three *clues*: "to have left Bourg-d'Oisans", "to be under a power line" and "to have fallen from a path". Figure 7.10 shows the map in the upper right-hand corner for an overview of the *clues* of the search *hypothesis*, while the map on the right and the terrain view show one of the fragments of the PLA. Figure 7.11 shows an overview of the search *hypothesis*, both on a map and on a terrain view.

We have discussed here the interest of a multi-map display to represent the data of a single search hypothesis. Such a multi-map display is also of obvious interest to allow the visualisation of data relating to two search hypotheses which would be carried out simultaneously by the rescuer to resolve the same alert.

[5] All of them are available in the code repository.

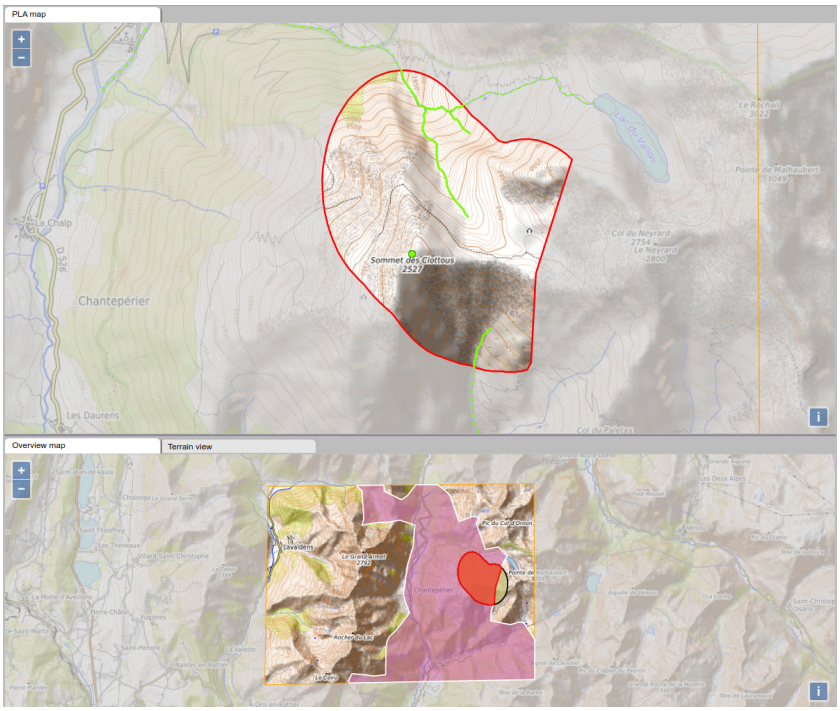

Figure 7.9: Example of the use of two maps of the same situation at different zoom levels.

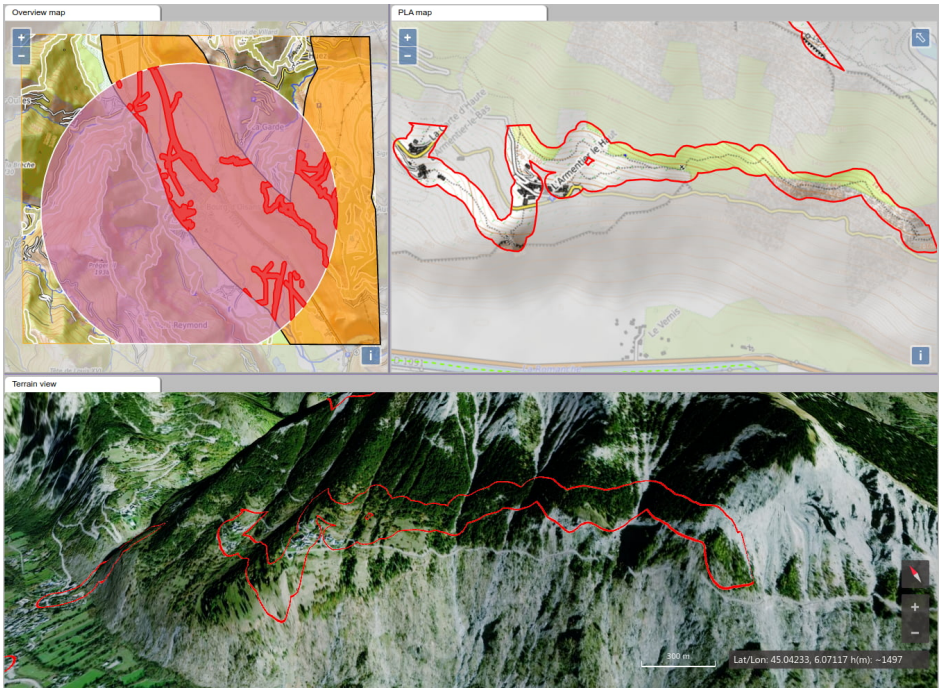

Figure 7.10: Example of the use of two maps and a terrain view.

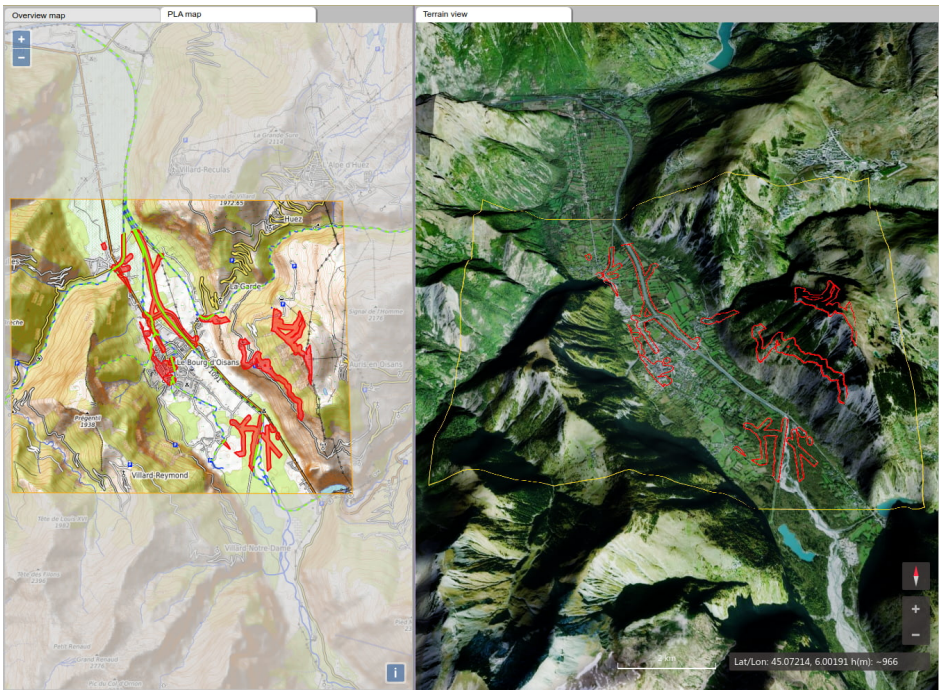

Figure 7.11: Example of the use of a map view with a terrain view.

7.5.3 Adding interactions and portrayals for effective highlighting of information

The *covikoa-interaction* vocabulary has not been used so far. However, it is important to define various interactions in order to obtain a geovisualisation application that can provide additional information about the data and link the data from different components.

We propose here to define two interactions (Listing 7.16) and to show their materialisation in the resulting application. These interactions emphasise two of the different ways of writing them that we presented in Section 5.2.2. The first (`:identifyToPopup`) concerns the display of information on reference objects that are highlighted because they are in the PLA (`:PortrayalReferenceFeatureHighlightInsidePLA`). When they are clicked (Figure 7.12), the additional information is displayed on a dedicated `gviz:PopupComponent` that was defined earlier (Listing 7.15). The second (`:highlightOnTerrain`) concerns the highlighting of CLAs on the terrain view, when they are hovered over on the overview map (Figure 7.13). This interaction can be triggered when hovering over the CLA whatever the scale of visualization because the two portrayals (`:PortrayalCLAHighScale` and `:PortrayalCLA`) allow this interaction. Doing so allows the rescuer to perceive the influence of each of the CLAs on the PLA, possibly prompting the rescuer to remove one of the *clues* that would restrict the PLA too much or of which the caller is not completely sure.

```
1 :identifyToPopup a ion:Interaction ;
```

```
    ion:isTriggeredBy ion:singleClick ;
    ion:hasAnalyticalPurpose ion:identify ;
    ion:hasEnding [ a ion:Closable ] ;
    ion:hasTargetOutcome [ a ion:Outcome ;
        ion:onComponent :identifyPopup ;
      ] .

:PortrayalReferenceFeatureHighlightInsidePLA ion:allowsInteraction :identifyToPopup.

:highlightOnTerrain a ion:Interaction ;
    ion:hasAnalyticalPurpose ion:identify ;
    ion:isTriggeredBy ion:mouseOver ;
    ion:hasEnding [ a ion:DuringEvent ] ;
    ion:hasTargetOutcome [ a ion:Outcome ;
        ion:hasInteractionSymbolizer  [ a symblzr:PolygonSymbolizer ;
            graphic:hasStroke [ a graphic:Stroke ;
                graphic:strokeColor "rgba(99,99,99,1)"^^graphic:cssColorLiteral ;
              ] ;
            graphic:hasFill [ a graphic:Fill ;
                graphic:fillColor "rgba(255,255,0,0.7)"^^graphic:cssColorLiteral ;
              ] ;
        ] ;
        ion:hasOutcomeSelectionStrategy [ a ion:SameIndividual ;
            ion:targetsEntitiesFrom :PortrayalCLATerrain ;
          ] ;
      ] .

:PortrayalCLAHighScale ion:allowsInteraction :highlightOnTerrain .
:PortrayalCLA  ion:allowsInteraction :highlightOnTerrain .
```

Listing 7.16: Definition of two `ion:Interaction`s.

7.6 Conclusion

In this chapter, we have addressed the implementation of our approach on a case study dealing with the processing of victim search in the mountains. The Choucas Alert Ontology (OAC) is an ontology that we have defined in the CHOUCAS project to handle the knowledge involved in such a rescue activity. This ontology mobilises the concepts used in the CHOUCAS project and provides elements of vocabulary used in the exchanges between the different members of the project. We have introduced a light version of OAC called LOAC (Section 7.2) and used it to exemplify how to operate using the CoViKoa framework to build geovisualisations from this data model.

We have presented the writing of the specification document, the Derivation Model, which describes how to geovisualise the data described by LOAC. This allows us to highlight the strengths of the CoViKoa framework by building a geovisualisation for the different elements of the targeted SDM by selecting the data using constraints and possibly transforming their geometries (Section 7.3) but also by integrating new data to be transformed and represented according to a more complex selection logic

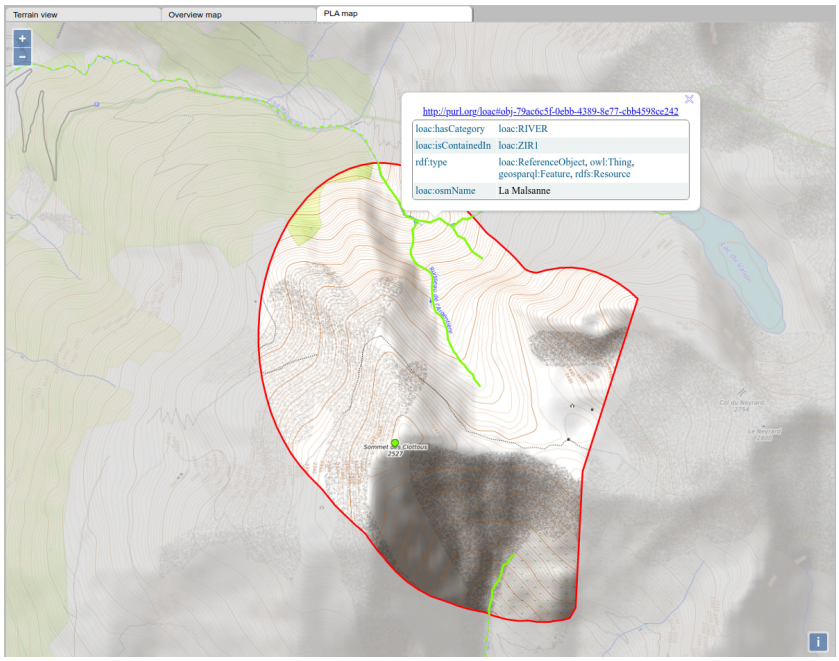

Figure 7.12: Result of the interaction to display a popup when clicking on a *reference object*.

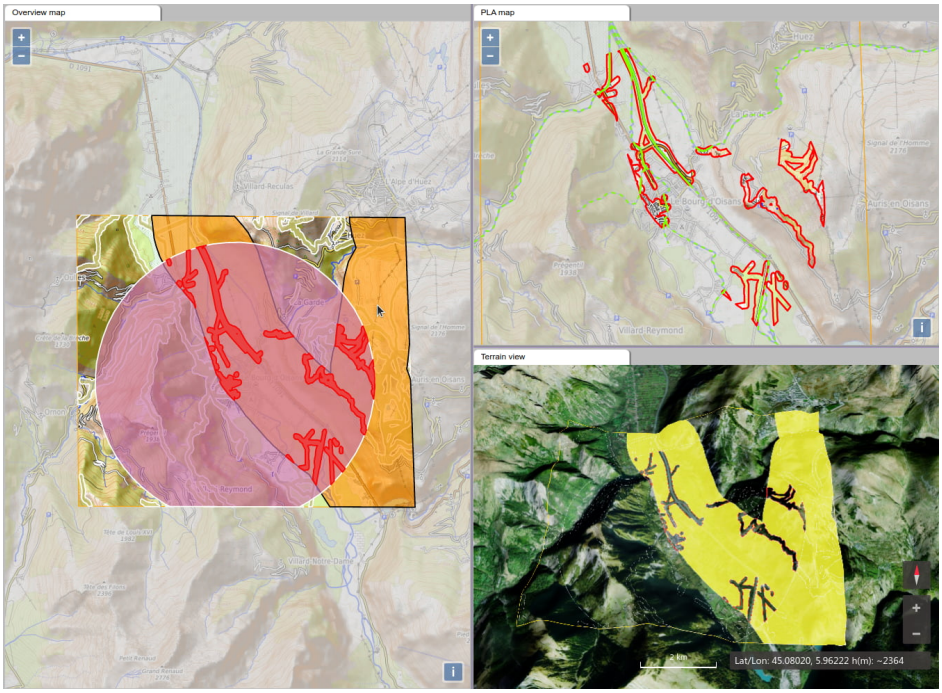

Figure 7.13: Result of the interaction to highlight a *Compatible Location Area* on the terrain view when hovering it in the map view.

(Section 7.4). The other strong points put forward concern the possibility of defining and organising the various components proposed by *covikoa-geoiz* vocabulary as well as the possibility of defining interactions between the data of these components (Section 7.5).

In the following chapter we show the interest of CoViKoa to exploit data described by an ontology which was not created by us, in order to demonstrate the generic and versatile features of CoViKoa.

Chapter 8

Towards genericity validation against other ontological models and discussion

8.1 Introduction

In Chapter 7 we presented the use of CoViKoa to exploit the *Choucas Alert Ontology*, an ontological model that we designed in the context of the CHOUCAS project.

In this chapter we wish to validate the genericity of our framework by using it to exploit another ontological model describing spatial entities. For this purpose, we have chosen the models *TSN* and *TSN-Change* (Bernard, 2019). In a first step we briefly present these two ontologies and discuss representations and interactions that could be useful (Section 8.2). Secondly, we describe some specific elements of the Derivation Model that allow us to exploit these ontologies to produce a geovisualisation interface that makes sense with regard to the data to be visualised and the knowledge that can be constructed from these data (Section 8.3). Finally, we discuss the limitations encountered in the use of CoViKoa (Section 8.4).

8.2 TSN and TSN-change models and their geovisualisation potential

8.2.1 TSN ontology

The *TSN* ontology (Bernard, 2019) allows the description of the hierarchical structure of Territorial Statistical Nomenclatures (TSN). It has been designed to allow the description of any TSN in the world. Its central concepts (Figure 8.1) are:

- *Nomenclature*, which describes "*an abstract representation of the subdivisions of a territory into Levels and Territorial Units (i.e. also called 'Structure'). A Nomenclature can be seen as a hierarchy of sets of Units with at least one Level. It is designed to answer administrative, electoral and especially statistical needs.*" (Bernard et al., 2019),
- *Level*, which describes "*a level of observation of the Territory defined within the Nomenclature. Several levels of observation may be defined (e.g., region or district levels).*" (Bernard et al., 2019),
- *Unit* which describes "*a piece of geographic space that belongs to one Level of a Nomenclature, and may in turn contain sub-units (or not, if the unit belongs to the lowest level of the nomenclature). Statistical observation are made on Unit.*" (Bernard et al., 2019),
- *Territory* which describes "*an abstract representation of a portion of geographic space that is claimed or occupied by a person or group of persons or by an institution*" (Bernard et al., 2019) and which is a subclass of `geo:Feature`.

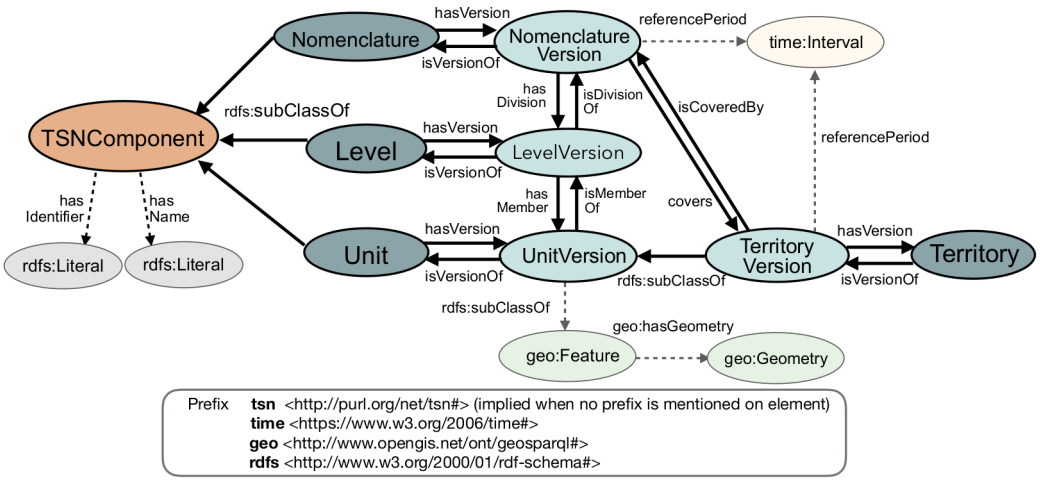

Figure 8.1: *TSN* ontology overview. Source: Image from Bernard (2019).

TSN ontology also contains the concepts needed to describe their different versions (*NomenclatureVersion*, *LevelVersion*, *UnitVersion* and *TerritoryVersion*).

This model is published online, its documentation and the RDF model can be found at the URL http://purl.org/net/tsn.

8.2.2 TSN-Change ontology

The second ontology presented is *TSN-Change* (Bernard, 2019). It is a model designed to describe territorial changes over time. Its central concept is the *Change* (Figure 8.2) which is subsumed by two broad categories, *StructureChange*, and *FeatureChange*, themselves subsumed by several types of change.

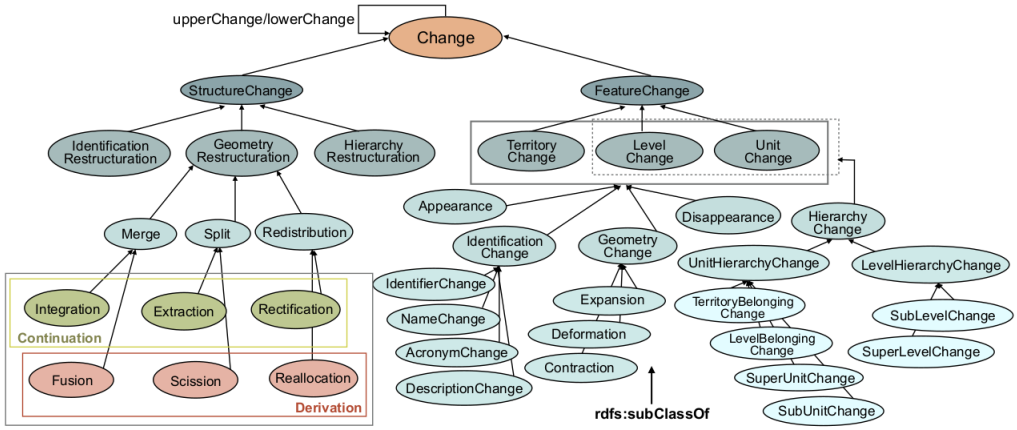

Figure 8.2: Overview of the change hierarchy in *TSN-Change* ontology. Source: Image from Bernard (2019).

One of the strengths of the proposed ontology is that it allows expressing that a change at one level of the TSN can be linked to changes at lower or higher hierarchical levels (e.g. if the boundary of a French region is moved, and the boundary of some of the departments that compose it is also moved). Bernard (2019) thus proposes a structure for qualifying these changes which she calls *XChange-Bridge* (X for eXtended) and that can be seen on Figure 8.3.

We also note the existence of two properties (not shown in Figure 8.3), `tsnchange:hasNextVersion` and `tsnchange:hasPreviousVersion`, which allow the `tsn:Versions` located on either side (input and output) of a `tsnchange:Change`, as well as consecutive `tsn:Versions` that do not change, to be linked directly.

This model is published online, its documentation and the RDF model can be found at the URL http://purl/org/net/tsnchange.

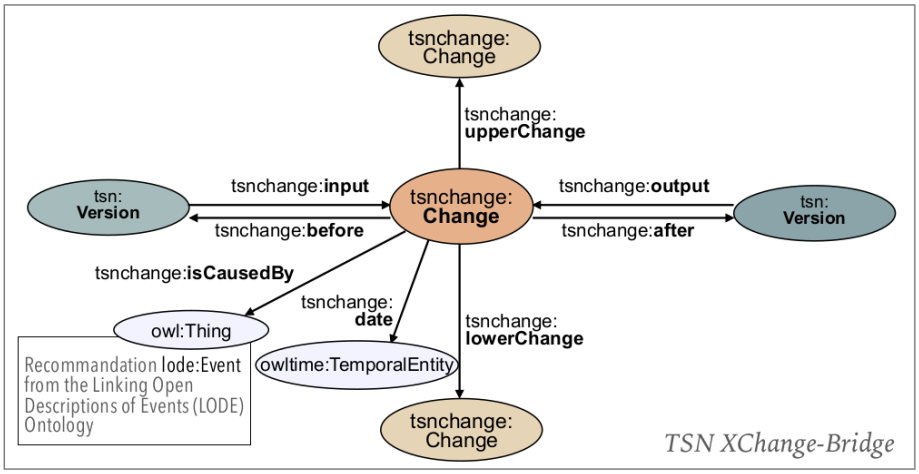

Figure 8.3: *TSN-Change* Ontology XChange-Bridge Model. Source: Image from Bernard (2019).

8.2.3 Datasets published in the web of data

The author then used these two ontologies as well as the other blocks of the framework she proposes (*Theseus*) to create the graphs containing the description of the changes undergone through time by TSNs at several hierarchical levels, those of the Switzerland Administrative Units (SAU), those of the Australian Statistical Geography Standard (ASGS) and those of the European Nomenclature of territorial units for statistics (NUTS).

In this chapter we focus on NUTS entities. The graph corresponding to the changes in each of these NUTS is available at the URL `http://purl.org/steamer/nuts` and is made queryable online using a SPARQL endpoint at the URL `http://steamerlod.imag.fr/repositories/geochange`.

8.2.4 Initial proposals of portrayal

In her manuscript, Bernard (2019) suggests a simple but potentially effective graphical symbology to represent the changes undergone by TSNs at a specific level between two versions.

This representation, which is first presented on the company's GitHub blog, takes the form of a *diff* between two versions of the same geographic layer. Figure 8.4 shows such a representation for a single entity: the 4th congressional district of Illinois after the 2011 redistricting, a textbook case of gerrymandering. As such, this representation shows in red the parts of the territory lost in 2011 (they no longer belong to the district) and in green the parts of the territory gained in 2011 (they

now belong to the district). In yellow/orange are the parts of the territory that were and still are part of the 4th congressional district of Illinois.

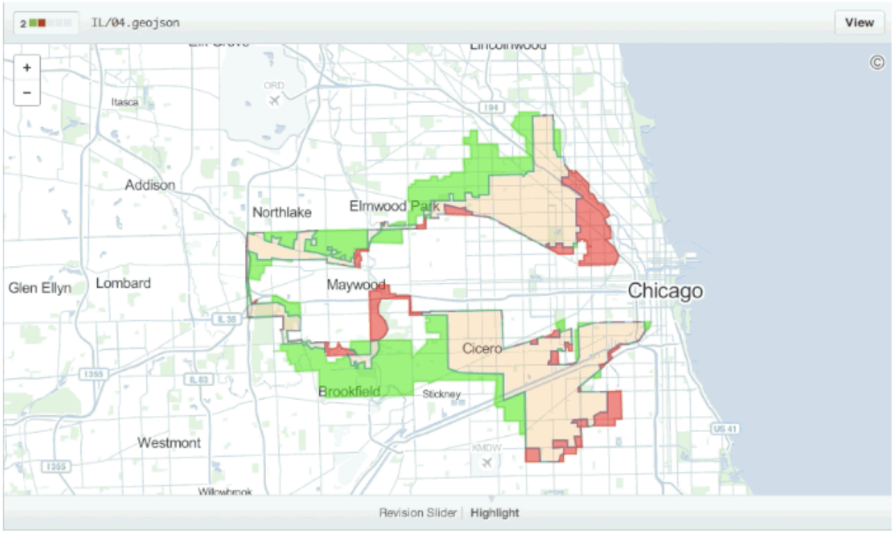

Figure 8.4: Visualising geometry changes between two versions. Source: https://github.blog/2014-02-05-diffable-more-customizable-maps/.

Bernard (2019) also proposes the queries that could be used to extract the entities to be represented for each of the 3 categories (Listing 8.1, Listing 8.2 and Listing 8.3) to represent the changes in European NUTS1 between the 1999 and 2003 versions. In her proposal, this representation could be applied as follows: in green the units that do not change, in orange the entities that change but do not disappear, and in red the entities that disappear. This proposition remains conceptual in her work.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX tsn: <http://purl.org/net/tsn#>
PREFIX nuts: <http://purl.org/steamer/nuts/>
PREFIX tsnchange: <http://purl.org/net/tsnchange#>

SELECT DISTINCT ?TU WHERE {
    ?TU rdf:type tsn:UnitVersion .
    ?TU tsn:isMemberOf ?level .
    ?level tsn:hasIdentifier "NUTS_V1999_L1" .
    FILTER (!EXISTS { ?TU tsnchange:input ?change . })
}
```

Listing 8.1: SPARQL query that requests all the territorial units at a specific level that do not change. Source: Bernard (2019).

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX tsnchange: <http://purl.org/net/tsnchange#>
PREFIX tsn: <http://purl.org/net/tsn#>
PREFIX nuts: <http://purl.org/steamer/nuts/>

SELECT DISTINCT ?TU WHERE {
    ?TU rdf:type tsn:UnitVersion .
    ?TU tsnchange:input ?change .
```

```
    ?change a tsnchange:Change .
    ?TU tsn:isMemberOf ?level .
    ?level tsn:hasIdentifier "NUTS_V1999_L1" .
    MINUS {?change a tsnchange:Disappearance . }
}
```

Listing 8.2: SPARQL query that requests all the territorial units that change from one version to another at a specific territorial level. Source: Bernard (2019).

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX tsnchange: <http://purl.org/net/tsnchange#>
PREFIX tsn: <http://purl.org/net/tsn#>
PREFIX nuts: <http://purl.org/steamer/nuts/>

SELECT DISTINCT ?TU WHERE {
    ?TU rdf:type tsn:UnitVersion .
    ?TU tsn:isMemberOf ?level .
    ?level tsn:hasIdentifier "NUTS_V1999_L1" .
    ?TU tsnchange:input ?change .
    ?change rdf:type tsnchange:Disappearance .
}
```

Listing 8.3: SPARQL query that requests all the territorial units at a specific level that disappear. Source: Bernard (2019).

We believe that we can implement and improve these proposals through several elements that are directly achievable with CoViKoa framework.

8.3 Using CoViKoa to visualise changes undergone by TSNs

We thus start from the intention of Bernard (2019) to visualise the changes undergone by a specific level of a TSN between two versions, as well as from her proposal to distinguish three main categories, depending on whether there is no change, there is a change of type *Disappearance* type or another change.

8.3.1 Requirements for geovisualising TSN changes

We can take advantage of several of CoViKoa's elements to propose an effective geovisualisation of TSN changes.

Firstly, we propose to use two maps, the first displaying the "old" territorial divisions and the second displaying the "new" territorial divisions, rather than displaying the difference between two versions on a single map. We think that this is more appropriate because of the changes in the TSNs we are interested in: if the proposal of Bernard (2019) seems to be effective in describing *Geometry Changes*,

this type of change constitutes only a part of the changes. Other types of change (as modification of the code or of the name of a unit in a TSN) exist and have to be consider in the rendering.

Secondly we decide to propose a symbology adapted to these two maps. The first map is thus intended to represent the territorial divisions at time *T-1* and the second map at time *T* (for example respectively 1999 and 2003). We propose to represent in the same grey, on both maps, the territorial entities which were the subject of no change. We then decide to represent, on the first map, in orange the territorial entities which underwent a change other than disappearance and in red the territorial entities having undergone a change of disappearance type. Then, on the second map, we decide to represent the entities differently according to the type of change that allowed their creation: indeed, in the TSN and TSN-Change models applied to NUTS data, an entity that changes its identifier is subject to disappearance. We think that it can be useful to distinguish the new entities resulting from a change of geometries from the other new entities.

We finally think that we should exploit the link that exists in RDF (with the property `tsn:hasNextVersion`) between the entities of two versions, for example to show on the second map which entity or entities result from an entity of the first map. This is an existing possibility in *covikoa-interaction* vocabulary that we have not yet exemplified.

8.3.2 Writing the Derivation Model

We present here some of the elements making the specificity of the Derivation Model written to exploit *TSN* and *TSN-Change* data[1].

We start by defining a `cvkd:DataIntegrationRule` to retrieve the data to be geovisualised (see Listing 8.4). The SPARQL query that composes the core of this rule retrieves the data that corresponds to the NUTS of level 1 for the years 1999 and 2003, the link between them (through the `tsnchange:hasNextVersion` property), as well as the types of changes that allowed the passage from one version to another. It queries all the information needed to search for matches according to the type of change as in the queries presented by Bernard (2019) and shown previously (Listing 8.1, Listing 8.2 and Listing 8.3). These matches will be defined in the Derivation Model using *Property Constraints*.

```
:IntegrateNuts1Data
    a sh:NodeShape , cvkd:DataIntegrationRule ;
    sh:targetNode :UnitVersionGVIR ;
    sh:rule [
        a sh:SPARQLRule ;
        sh:construct """
    PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
    PREFIX tsn: <http://purl.org/net/tsn#>
```

[1]The case study for this chapter is provided in the code repository as "case study #5".

```
    PREFIX nuts: <http://purl.org/steamer/nuts/>
    PREFIX tsnchange: <http://purl.org/net/tsnchange#>
    PREFIX geo: <http://www.opengis.net/ont/geosparql#>

    CONSTRUCT {
        ?TU rdf:type tsn:UnitVersion .
        ?TU tsn:isMemberOf ?level .
        ?TU geo:hasGeometry ?TUGeomIri .
        ?TUGeomIri geo:asWKT ?geom .
        ?TU tsn:hasName ?name .
        ?TU tsn:hasIdentifier ?id .
        ?TU tsnchange:hasNextVersion ?nextVersion .
        ?TU tsnchange:input ?input .
        ?input a ?what .
        ?nextVersion rdf:type tsn:UnitVersion .
        ?nextVersion geo:hasGeometry ?nextVersionGeomIri .
        ?nextVersionGeomIri geo:asWKT ?geomNextVersion .
        ?nextVersion tsn:hasName ?nameNextVersion .
        ?nextVersion tsn:hasIdentifier ?idNextVersion .
    }
    WHERE {
        NOT EXISTS { ?TU tsn:hasIdentifier ?id }
        SERVICE <http://steamerlod.imag.fr/repositories/geochange>
        {
            SELECT ?TU ?geom ?category ?name ?id ?nextVersion ?nameNextVersion ?
    idNextVersion ?geomNextVersion ?TUGeomIri ?nextVersionGeomIri ?what ?input WHERE
     {
                ?TU rdf:type tsn:UnitVersion .
                ?TU tsn:isMemberOf ?level .
                ?level tsn:hasIdentifier "NUTS_V1999_L1" .
                ?TU tsn:hasName ?name .
                ?TU tsn:hasIdentifier ?id .
                ?TU tsnchange:hasNextVersion ?nextVersion .
                ?TU geo:hasGeometry [ geo:asWKT ?geom ] .
                ?nextVersion geo:hasGeometry [ geo:asWKT ?geomNextVersion ] .
                ?nextVersion tsn:hasName ?nameNextVersion .
                ?nextVersion tsn:hasIdentifier ?idNextVersion .
                BIND(iri(CONCAT('urn:', STR(?id), '-geom')) as ?TUGeomIri)
                BIND(iri(CONCAT('urn:', STR(?idNextVersion), '-geom')) as ?
    nextVersionGeomIri)
                OPTIONAL {
                    ?TU tsnchange:input ?input .
                    ?input a ?what .
                }
            }
        }
    } """ ] .
```

Listing 8.4: Data Integration Rule that builds a graph linking the entities of two consecutive versions.

We then need to create a subclass of `gviz:GeoVisualIntermediateRepresentation` for the only concept related to geospatial data, `tsn:UnitVersion`, we call it `:UnitVersionGVIR` . We also define two maps, two legends linked to them, and a popup component. We omit these statements for the sake of brevity but note that these two maps are named `:mapBefore` and `:mapAfter`, and will respectively display NUTS1 in their 1999

version and in their 2003 version.

Next, we define two portrayals in accordance with the proposal of representation we made in the previous subsection, one for each map, and we name them `:PortrayalBefore` (Listing 8.5) and `:PortrayalAfter` (Listing 8.6). We can see that the various property constraint (`cvkd:PropertyConstraint`, cf. Chapter 5) mechanisms needed are directly accessible in *covikoa-derivation* to exploit the way the data is structured. To do so, we use several of the specificities presented in the previous chapters: the possibility to create property constraints composed of a logical operator articulating two or more property constraints (Listing 8.5, line 11 and Listing 8.6, line 11), the possibility to define a constraint meaning the selection of entities when a property is absent or present (Listing 8.5, lines 14 and 18 and Listing 8.6, lines 14 and 18), and the possibility to define a constraint on the RDF type of a targeted individual (Listing 8.5, lines 29 and 39 and Listing 8.6, lines 29 and 39).

```
:PortrayalBefore a gviz:Portrayal ;
    dct:title "NUTS1, 1999, type of change" ;
    cvkd:denotesGVR :UnitVersionGVIR ;
    gviz:onComponent :mapBefore ;
    ion:allowsInteraction :identifyToPopup ;
    gviz:hasPortrayalRule [
        gviz:hasSymbol [
            dct:title "No change in 2003" ;
            symblzr:hasSymbolizer :SymbolizerNutsGrey  ;
          ] ;
        cvkd:hasPropertyConstraint [ cvkd:and (
            [
                cvkd:propertyPath tsnchange:input ;
                cvkd:valueOrObjectIsEqualTo cvkd:absentProperty ;
            ]
            [
                cvkd:propertyPath tsnchange:hasNextVersion ;
                cvkd:valueOrObjectIsEqualTo cvkd:presentProperty  ;
            ]
          )] ;
      ] ;
    gviz:hasPortrayalRule [
        gviz:hasSymbol [
            dct:title "Modified (except for disappearance) in 2003" ;
            symblzr:hasSymbolizer :SymbolizerOrange ;
          ] ;
        cvkd:hasPropertyConstraint [
            cvkd:propertyPath tsnchange:input ;
            cvkd:objectNotOfType tsnchange:Disappearance ;
          ] ;
      ];
    gviz:hasPortrayalRule [
        gviz:hasSymbol [
            dct:title "Disappeared in 2003" ;
            symblzr:hasSymbolizer :SymbolizerPinkRed ;
          ] ;
        cvkd:hasPropertyConstraint [
            cvkd:propertyPath tsnchange:input ;
            cvkd:objectOfType tsnchange:Disappearance ;
          ] ;
```

```
41      ] .
```

Listing 8.5: Portrayal for the territorial unit before change.

```
1  :PortrayalAfter a gviz:Portrayal ;
2      dct:title "NUTS1, 2003" ;
3      cvkd:denotesGVR :UnitVersionGVIR ;
4      gviz:onComponent :mapAfter ;
5      ion:allowsInteraction :identifyToPopup ;
6      gviz:hasPortrayalRule [
7          gviz:hasSymbol [
8              dct:title "Entity unchanged from 1999" ;
9              symblzr:hasSymbolizer :SymbolizerNutsGrey  ;
10           ] ;
11         cvkd:hasPropertyConstraint [ cvkd:and (
12             [
13                 cvkd:propertyPath ([sh:inversePath tsnchange:hasNextVersion]) ;
14                 cvkd:valueOrObjectIsEqualTo cvkd:presentProperty ;
15             ]
16             [
17                 cvkd:propertyPath ([sh:inversePath tsnchange:hasNextVersion]
       tsnchange:input) ;
18                 cvkd:valueOrObjectIsEqualTo cvkd:absentProperty ;
19             ]
20           )] ;
21       ] ;
22     gviz:hasPortrayalRule [
23         gviz:hasSymbol [
24             dct:title "Entity changed from 1999 (Geometry change)" ;
25             symblzr:hasSymbolizer :SymbolizerGreen ;
26           ] ;
27         cvkd:hasPropertyConstraint [
28             cvkd:propertyPath ([sh:inversePath tsnchange:hasNextVersion] tsnchange:
       input) ;
29             cvkd:objectOfType tsnchange:GeometryChange ;
30           ];
31       ];
32     gviz:hasPortrayalRule [
33         gviz:hasSymbol [
34             dct:title "Entity changed from 1999 (other change)" ;
35             symblzr:hasSymbolizer :SymbolizerGreen2 ;
36           ] ;
37         cvkd:hasPropertyConstraint [
38             cvkd:propertyPath ([sh:inversePath tsnchange:hasNextVersion] tsnchange:
       input) ;
39             cvkd:objectNotOfType tsnchange:GeometryChange ;
40           ];
41       ] .
```

Listing 8.6: Portrayal for the territorial unit after change.

As shown in the previous chapters, we also define an interaction (`:identifyToPopup`) to display the properties of the entity that receives a click, both on the materialisations of entities produced by `:PortrayalBefore` and by `:PortrayalAfter`.

One of the possibilities not examined so far is that of describing interactions whose result does not (or not only) apply to the corresponding entities of the entity

with which the interaction occurs. In this case, the result of the interaction is encoded to occur on the entities described by a property (`tsnchange:hasNextVersion`) which is read during the interaction (Listing 8.7).

In Listing 8.7, we define an interaction occurring on hovering (`ion:mouseOver`, line 4) over entities produced by `:PortrayalBefore` (line 1) and having three outcomes (lines 6-8). In order to distinguish these outcomes clearly throughout the discourse of this section, we have defined them as named individuals (`:outcomeNextVersion`, `:outcomeHoveredEntity` and `:outcomeOtherEntity`). These three outcomes are distinguished by different elements: they do not apply to entities of the same portrayal, the entity selection strategy is not the same and the type of outcome is not the same.

```
1  :PortrayalBefore ion:allowsInteraction :identifyNextVersion .
2
3  :identifyNextVersion a ion:Interaction ;
4      ion:isTriggeredBy ion:mouseOver ;
5      ion:hasAnalyticalPurpose ion:identify ;
6      ion:hasTargetOutcome :outcomeNextVersion ;
7      ion:hasTargetOutcome :outcomeHoveredEntity ;
8      ion:hasRestOutcome :outcomeOtherEntity ;
9      ion:hasEnding [ a ion:DuringEvent ] .
10
11 :outcomeNextVersion a ion:Outcome ;
12     ion:hasOutcomeSelectionStrategy [ a ion:FollowPropertyPath ;
13         ion:propertyPath tsnchange:hasNextVersion ;
14         ion:targetsEntitiesFrom :PortrayalAfter ;
15       ] ;
16     ion:hasInteractionSymbolizer [ a symblzr:PolygonSymbolizer ;
17         graphic:hasStroke [ a graphic:Stroke ;
18             graphic:strokeWidth 2 ;
19             graphic:strokeColor "rgba(255,255,0,1)"^^graphic:cssColorLiteral ;
20           ] ;
21         graphic:hasFill [ a graphic:Fill ;
22             graphic:fillColor "rgba(255,192,203,1)"^^graphic:cssColorLiteral ;
23           ] ;
24       ] .
25
26 :outcomeHoveredEntity a ion:Outcome ;
27     ion:hasOutcomeSelectionStrategy [ a ion:SameIndividual ;
28         ion:targetsEntitiesFrom :PortrayalBefore ;
29       ] ;
30     ion:hasSymbolizerModifier [ a ion:SymbolizerModifier ;
31         ion:onProperty (graphic:hasStroke graphic:strokeColor) ;
32         ion:modifiedValue "rgba(255, 255, 0, 1)"^^graphic:cssColorLiteral ;
33       ] ;
34     ion:hasSymbolizerModifier [ a ion:SymbolizerModifier ;
35         ion:onProperty (graphic:hasStroke graphic:strokeWidth) ;
36         ion:modifiedValue 2 ;
37       ] .
38
39 :outcomeOtherEntity a ion:Outcome ;
40     ion:hasOutcomeSelectionStrategy [ a ion:SameIndividual ;
41         ion:targetsEntitiesFrom :PortrayalBefore ;
42       ] ;
43     ion:hasInteractionSymbolizer [ a symblzr:PolygonSymbolizer ;
44         graphic:hasStroke [ a graphic:Stroke ;
```

```
45              graphic:strokeWidth 2 ;
46              graphic:strokeColor "rgba(212,210,210,0.5)"^^graphic:cssColorLiteral ;
47            ] ;
48          graphic:hasFill [ a graphic:Fill ;
49              graphic:fillColor "rgba(237,237,237,0.4)"^^graphic:cssColorLiteral ;
50            ] ;
51        ] .
```

Listing 8.7: Interaction to highlight the next version(s) of an hovered entity as well as the hovered entity itself.

The first outcome (`:outcomeNextVersion`, line 11) allows us to express that we wish to highlight the materialisation(s), produced by the `:PortrayalAfter` portrayal, which correspond to the entities located at the end of the `tsnchange:hasNextVersion` property of the hovered entity. We call `ion:FollowPropertyPath` (line 12) this selection strategy `ion:SelectionStrategy`. The identifiers of these materialisations are requested at interaction time, the result is cached, and these identifiers are used to highlight the appropriate entities.

The second outcome (`:outcomeHoveredEntity`, line 26) concerns the highlighting of the materialisation produced by the `:PortrayalBefore` portrayal (this is also the portrayal on which the interaction takes place) and which corresponds to the hovered entity. This is a case of selection using the `ion:SameIndividual` selection strategy which means that materialisation to highlight is also the one that is hovered. Another difference from the outcomes presented so far is the use of a `ion:SymbolizerModifier` rather than a new `symblzr:Symbolizer` (lines 30 and 34). This enables the property that needs to be modified (in this case the colour and width of the stroke) to be pointed at, while retaining the other properties of the symbolizer (in this case its fill property), allowing the category, expressed by their fill colour, of the entity being hovered over to continue to be read.

These first two outcomes are linked to the interaction through the property `ion :hasTargetOutcome`. This is the most intuitive case where the corresponding entities (either located at the end of the property path that has been read, or corresponding directly to the entity interacted with) are the ones that should receive the interaction.

The third outcome (`:outcomeOtherEntity`, line 40) is related to the interaction through the property `ion:hasRestOutcome`. This means that it is not the materialisation(s) of the entities produced that must be selected, but the inverse of those entities within the specified portrayal. As such, this third outcome also uses the `ion:SameIndividual` strategy selection on the `:PortrayalBefore` portrayal. But, as it is linked to the interaction by the `ion:hasRestOutcome` property, the symbolizer applies to all materialisations in `:PortrayalBefore` except the one the user is interacting with. This allows us to create a representation that slightly obscures the entities we are not interested in, thus implementing a variation of the highligthing techniques proposed by Robinson (2011) that we discussed in Chapter 2.

In other words, the first outcome highlights the entity or entities corresponding

to the next version, on the other map. The second outcome highlights the entity that has been hovered over. And the third outcome slightly hides the colour values of all the entities that are not hovered over (the opposite of the second outcome). The resulting geovisualisation is detailed in the following subsection

8.3.3 The resulting geovisualisation

The resulting geovisualisation (Figure 8.5) displays the two maps side by side, partitioning the window equally in its centre, with the 1999 map on the left and the 2003 map on the right, as defined in the Derivation Model. Each map has its own legend. The views are synchronised between the two maps, so that a crosshair cursor appears as a mirror image on the second map of the cursor on the map overlaid.

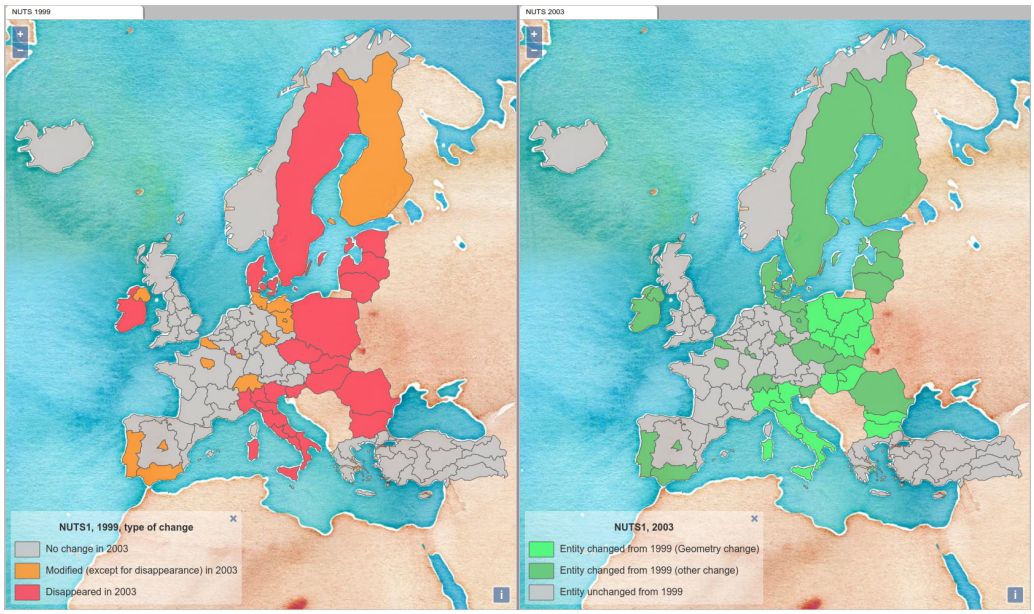

Figure 8.5: Overview of the resulting geovisualisation for the TSN case study.

Thanks to the particular shade of green used on the second map to materialise the entities that have appeared following a change in geometry, we feel that we help identifying these entities which are recognisable at first glance. Using a different shade of green (and not another colour) allows us to maintain a reduced number of colours distinguishing the 3 main elements on these two maps: red for disappearance, orange for modification (input of the changes between the two versions - on the first map) and green for creation (output of the changes between the two versions - on the second map) as well as entities that have not changed between versions, in grey.

Finally, the interaction that we have defined above (Listing 8.7) makes it easy to see which entity (or entities) of the new version 2003, correspond to a particular entity of the first version. This is for example the case for the NUTS1 "PL" of 1999

entity which has notably undergone a restructuring of its geometry (of the Scission type) to give rise in 2003 to several NUTS1: "PL1", "PL2", "PL3", "PL4", "PL5" and "PL6" entities (Figure 8.6).

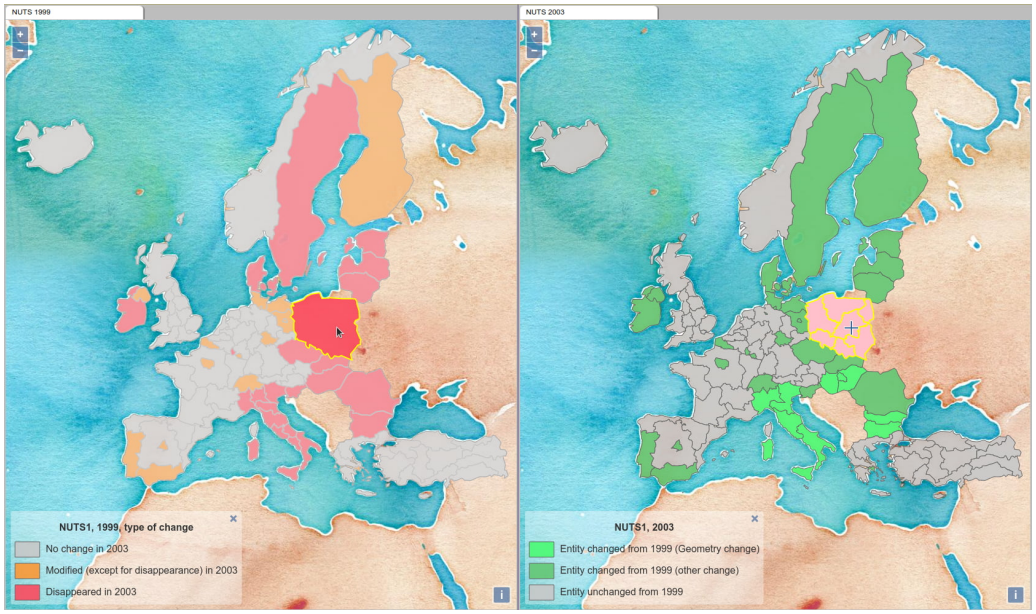

Figure 8.6: Outcomes of the interaction on mouseover on the first map.

A limitation of this interaction is that it does not fully capture the restructuring (of *Merge* type) of geometries as is the case for some of the Italian NUTS1 (Figure 8.7). In order to fully capture this information and render it to the user, it could have been interesting to highlight, on the left-hand map, the other entities that were used to create the NUTS of the next version (the one highlighted on the right-hand map), in a different colour from the NUTS over which the cursor is hovering.

This is possible in CoViKoa by defining different portrayal rules for the disappearing entities, depending on whether the entities are linked to a *Merge* change or not (even if each of these portrayal rules produces the same red symbolizer), then by attaching a different interaction to each of these portrayal rules, and by defining outcomes mixing the selection strategies seen previously (and the fact that our interaction mechanism can target entities produced by a portrayal or by a portrayal rule). However this possibility is not very ergonomic for the geovisualisation designer (compared to having a constraint mechanism directly when defining interactions) and we discuss this point in Section 8.4.

We believe that, although the information represented on these two maps simplifies the reality described in the change graph constituted by Bernard (2019), it captures the essence of the changes undergone thanks to the combination of the information represented on the two maps and a popup displaying information accessible when clicking on entities of the first map. Figure 8.8 for instance displays the information for 1999 NUTS1 of Ireland and Ile-de-France, respectively shown in red and

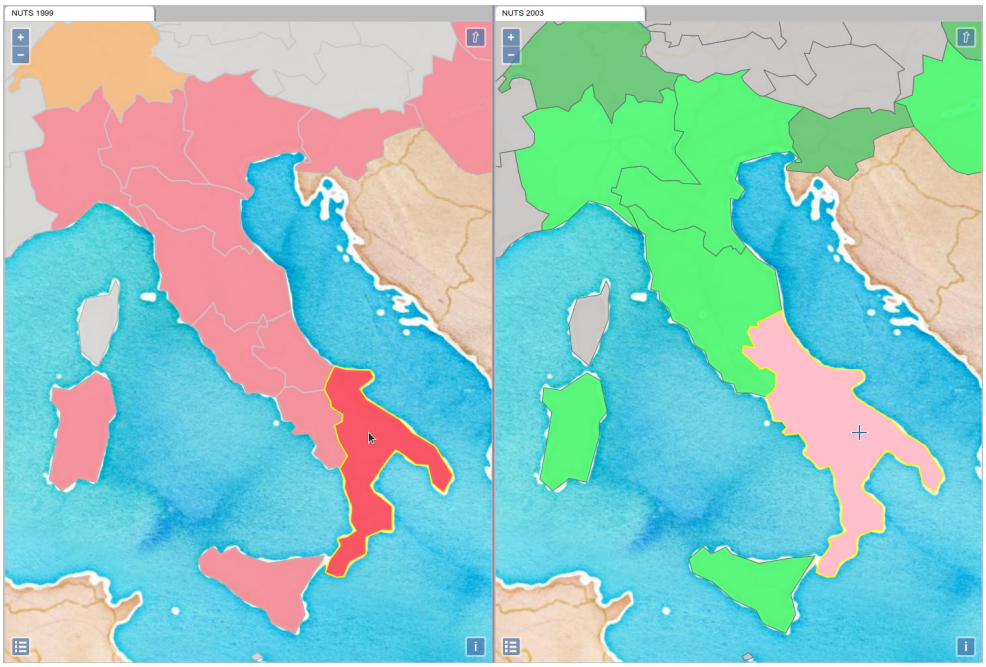

Figure 8.7: Outcomes of the interaction on mouseover on the first map (next version entity is larger that the one hovered on).

orange on the left-hand map.

We also believe that, due to the specific nature of the changes undergone by entities within a TSN, our representation is better than that of a three-colour diff display. Indeed, few geometries (in fact, none in the case of the NUTS1 changes from 1999 to 2003) are modified in the manner of the 4th district of Illinois, i.e. with portions of territory lost, an unchanged core, and portions of territory gained. Moreover, even if this had been the case, it would have been difficult to represent at the scale of the whole NUTS1 level, notably because the overwhelming majority of the entities are spatially adjacent to other entities (thus creating a representation conflict between the parts lost by a specific territory and gained by the adjacent territory).

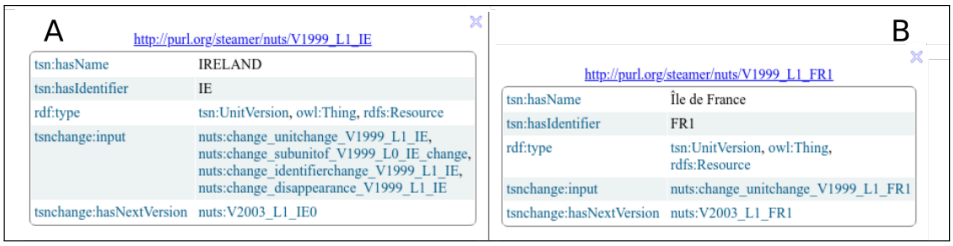

Figure 8.8: Popup to display additional information.

Nevertheless, we can suspect that some changes had repercussions on the entities

of the lower hierarchical levels, in our case on the NUTS2 level (it is for example probably the case for the change of identifier and when new entities have been created). This information is present in the data graphs constituted by Bernard (2019) but it is absent from our current representations. It could have been shown in different ways:

- by adding a specific overlaid portrayal (hatch or punctual symbol for example) on the NUTS1 entities whose change led to a change on the corresponding NUTS2,
- by representing the lower level entities, the NUTS2, on the same map (e.g. by showing their borders in a lighter colour than that of NUTS1),
- by displaying the said NUTS2 entities, for example when clicking on a NUTS1 entity or when a specific zoom level is reached by the user.

All these opportunities are possible with CoViKoa but were not explored in this manuscript due to time constraints.

8.4 Discussions on the limitations

While this case study on TSN change graphs from Bernard (2019) highlights several strengths of our framework, it also reveals some of its limitations.

Among these strengths, we believe that CoViKoa offers a quick way to create a geovisualisation from this type of data. This is due in particular to the fact that CoViKoa allows us to avoid thinking about how to transform the data described in RDF into a more classical formalism for GIS data. Of course, dedicated experiments, such as a comparative evaluation with other geovisualisation techniques for RDF data, have to be conducted to confirm this statement.

Finally, this is also the case because the *covikoa-client* web application implements all the possibilities that can be described with our vocabularies, avoiding any development costs once the Derivation Model is correctly written. Indeed, we can write the different elements related to the retrieval of RDF data and their representation in a single document, the Derivation Model, and obtain the corresponding geovisualisation within seconds of launching CoViKoa.

Conversely, the limitations we have encountered are related to the necessary retrieval of the RDF data we are exploiting and to shortcomings in our modelling of interactions. These two elements are discussed below, and avenues for improvement are presented.

8.4.1 Retrieval of remote data to be displayed

The first limitation is inherent in the way our framework was developed. Indeed, the CoViKoa framework works with the data of the Semantic Data Model to be used in a local graph (more precisely in a Jena model which is linked with the data resulting from CoViKoa's reasoning). In case studies of Chapter 7, these data are contained in turtle files which are loaded when the framework is started. In order to integrate the data to be represented (whether it is the whole online dataset as in this chapter the TSN datasets or additional data as it was the case in Chapter 7 for the reference objects of an external source), we have also seen that the *Data Integration Rule* mechanism exists. In both cases the dataset becomes available locally at the time of the execution of CoViKoa's reasoning.

This limitation could easily be circumvented in future developments because of the mechanism we have created which allows to generate derivation and enrichment rules for a given Derivation Model. Indeed, in the case where the data to be exploited is located in a remote graph exposed behind a SPARQL endpoint (the address of which could be specified in the Derivation Model or when starting the application for example) it would suffice to include the SERVICE clause in all the appropriate places of the generated rules in order to read the data on this endpoint. Our individuals of type `gviz:GeoVisualIntermediateRepresentation` would no longer be linked (through the `gviz:represents` property) to an individual present in our graph but to a resolvable IRI on the remote service, as specified.

8.4.2 Interactions

Various elements related to interactions are not captured by our *covikoa-interaction* vocabulary. These include, for example, the various interactions relating to the movement of the map over a specific entity. Indeed, it is common, and this could be the case in the geovisualisation proposed in this chapter, that an event (e.g. a single click or a double click) leads to a zooming and panning action so that the user's window is adjusted to the size of the entity interacted with. In turn, it is also common for the same event, click or double click, to zoom back to the full extent of the map.

There is currently no option to do this in CoViKoa. We believe, however, that these are essentially new vocabulary elements to be added, and do not call into question the logic adopted when modelling *covikoa-interaction*.

Moreover, we saw in Section 8.3 that we were limited by the selection strategy of individuals that should receive an outcome, which could lead to an unergonomic expression of interactions or to the impossibility of expressing some of them. This limitation could be overcome if a mechanism similar to that of property constraints was available at the level of each outcome of an interaction.

Making these additions should lead to further reflection to identify the different

types of outcomes possible in a geovisualisation and the elements that characterise each of them, and then create the appropriate sub-classes of `ion:Outcome` in *covikoa-interaction* ontology.

8.5 Conclusion

In this chapter, after a brief presentation of the TSN and TSN-Change ontologies (Section 8.2), we have presented how we envision a geovisualisation for the data of these models and how to retrieve the data necessary to prepare a geovisualisation of the changes undergone by a level of a TSN between two versions (Section 8.3). We here acted as a geovisualisation designer who would use our approach, of course making here the hypothesis he or she is aware of the possibilities offered by our framework and initiated to it. We come back to this point in the next chapter.

We have also presented (Section 8.3) how to create the corresponding geovisualisation, based on two maps representing each version and on interactions exploiting the data stored in the graph and in particular the property allowing to link a unit to its following version(s). This case study contributes to demonstrate the applicability of our approach to a data model and a context different from the CHOUCAS project that inspired this work. This is a first step towards a larger validation of the genericity of our proposal.

In Section 8.4, we have discussed some of the choices of representations implemented using CoViKoa for this case study. It is worth noting that the prototyping possibilities offered by our framework have allowed us to test different options whose corresponding rendering have not been exposed in the manuscript. We think that such a flexibility for building geovisualisation is an argument in favour of our approach. Finally, this case makes us meet some limits we have also discussed.

In the next chapter, we conclude this manuscript by providing a summary of the contributions and discussing future research and development directions.

Chapter 9

Conclusion

Semantic Web technologies are gradually transforming the way data is published on the Web. Semantic models are now used to describe geospatial data at a high level (e.g. with GeoSPARQL) but also those of various more specific application domains (maritime, transport, etc.). These changes in practice are accompanied by new needs for portraying the geospatial data that is published according to these new formalisms, different from those traditionally used in GIScience.

While approaches that directly allow the creation of interactive maps from RDF data exist, they are limited by the little control that can be exercised over the portrayals that can be created and do not focus on the exploitation of a specific application domain (León et al., 2012) or they are limited by their complexity of use to create classical portrayals such as choropleths (Huang and Harrie, 2020; Huang et al., 2020). Furthermore, no proposal emphasises bringing all other aspects of geovisualisation to the Semantic Web data (such as the definition of interactions or the juxtaposition of components for a better understanding of a phenomenon).

The main question we have therefore answered in this manuscript is how one can use the formal description of a geovisualisation and the data that appear in it to facilitate the creation of a geovisualisation that supports the user's reasoning, in particular when the data is already in RDF and is described by an ontological model.

To this end, we first reviewed the state of the art of the various techniques used in geovisualisation and then examined the stack of Semantic Web technologies. We subsequently focused our analysis on the existing solutions for cartography or geovisualisation already using Semantic Web technologies. In a second part, we proposed a framework for the geovisualisation designer using Semantic Web technologies and based on an ecosystem of ontologies. Our approach goes further than the simple presentation of data since it allows the use of advanced selection mechanisms, the transformation of the data and the integration of Linked Data for the needs of the visualisation. These data are represented in several components and interactions

between them can be defined. It is these elements that bring us into the realm of geovisualisation and allow us to question the geovisualisation designer on how to effectively present the data to the end-user. As such, our approach presents several original features:

- it leads the geovisualisation designer to consider what is shown, and how to show it,
- this point is reinforced by the use of a purely declarative approach,
- and by the exploitation of the knowledge embedded in the data model.

Therefore, our approach, which relies on the data model to define but also generate these portrayals on the fly, can exploit the knowledge contained in the model or deduced from it. We believe that this encourages the geovisualisation designer to think about the most appropriate portrayals to support a geovisual analysis process that must be conducted on the basis of the data in the model in order to carry out a reasoning process (understand, decide, etc.). This intuition, based at this stage on our own practice and on our knowledge of other geovisualisation techniques for RDF data, will have to be validated by a comparative evaluation. As to our knowledge there are no proposals that are strictly equivalent to CoViKoa, a first type of evaluation could be qualitative, using the "think aloud" method (Jaspers et al., 2004). This has not yet been implemented because it requires developments such as those mentioned subsequently in the perspectives.

In this chapter we make an inventory of our contributions (Section 9.1) and then we highlight the perspectives of future works that could continue our research (Section 9.2).

9.1 Summary of the contributions

We have presented in this manuscript the different aspects of CoViKoa, a framework based on an ontology ecosystem, allowing to describe in RDF a geovisualisation application through different aspects: its components, the way in which the data (extracted from a semantic data model) appear and the interactions that can occur. If all the elements we propose have been developed in this framework, we believe that the ontologies that are proposed can also be used independently.

Firstly, we proposed **an ecosystem of several OWL ontologies to describe various aspects of a geovisualisation**.

- The first ontology we propose is *covikoa-geoviz*, an ontology allowing to describe the components of a geovisualisation and how data appears in them. This first contribution is inspired in particular by the proposal drafted by Villanova-Oliver (2018) of an ontology for geovisualisation, but also integrates elements proposed by Smith (2010).
- The second ontology proposed is *covikoa-interaction*, an ontology for describing the interactions that can take place in the various components of the interface.

The organisation of this ontology is based on the observation we made during our analysis of the state of the art and in particular of Roth (2012), which led us to think that it is possible to formalise an interaction thanks to three central concepts: the element that triggers the interaction, the user's intent (or analytical purpose), and the type of result obtained.

- The third ontology that we propose, *covikoa-context*, allows to describe the initial state of the geovisualisation and the initial visualisation context.
- The fourth ontology we propose, *covikoa-derivation*, is dedicated to expressing constraints and filters which describe how and the reason why a portrayal applies to specific individuals (because of some values or of semantic features). Some of the vocabulary elements of this ontology are familiar to geomaticians since there are similarities with the filters defined in SLD specification (OGC®, 2007). This ontology is strongly linked to the framework proposed hereafter since it is on these filters and constraints that it reasons.
- In addition, we propose a populated ontology to describe discrete colour palettes, *dicopal*, which includes palettes from Okabe and Ito (2002), Brewer et al. (2003), Light and Bartlein (2004), and Crameri (2018) that are classified according to whether they implement a sequential, divergent or qualitative scheme.
- Finally, we propose an ontology dedicated to formalising common cartographic methods: *carto*. This ontology is inspired by various works found in the literature (in particular Iosifescu-Enescu and Hurni, 2007; Smith, 2010; Brus et al., 2010) with the difference that we focus only on the description of common cartographic solutions (by including the various parameters that allow them to be qualified) and on the description of the data that can be used to build these solutions.

The other elements generally formalised in ontologies for cartography are thus here provided by other ontologies that we reuse from the literature. These are the *Scale* ontology, which is proposed by Carral et al. (2013) and taken over by Huang and Harrie (2020), as well as the *Graphic*, *Symbolizer* and *Symbol* ontologies which are proposed by Fellah (2017) and adopted by Huang and Harrie (2020) and Huang et al. (2020). The six ontologies we propose are designed to be compatible with these four ones, thus forming a coherent ecosystem of ontologies.

Secondly, we proposed **a framework to specify and generate a geovisualisation, which takes advantage of the semantics of an existing model to create the corresponding geovisualisation** :*CoViKoa*. The description of the geovisualisation is done through a specification document, the Derivation Model, which describes in RDF, using the aforementioned ecosystem of ontologies, how the data should be represented in the geovisualization components. In this sense, we extend and improve the approach described in Huang and Harrie (2020) and Huang et al. (2020) by making our approach purely declarative. It is thus possible to use the various filters, selections and transformations formalised in *covikoa-derivation* vocabulary. In addition, if the declaration of portrayal and portrayal rules is what allows CoViKoa to reason about the link between data and their portrayals, we also propose to define these portrayals much more succinctly as long as it is one of the classical cartographic representations described in *carto* vocabulary. This succinct

description is transformed into statements that use the *covikoa-geoviz* and *covikoa-derivation* vocabularies. All these elements are rewritten in the form of SHACL rules specific to the Derivation Model at the loading stage of CoViKoa. At runtime, the application of these rules makes it possible to create the link between the data to be visualised and the representations which correspond to them, thus making it possible to obtain a well-formed RDF graph according to the models proposed previously (in particular *covikoa-geoviz*). This data graph is published behind a SPARQL interface. Since our approach fully processes and publishes its presentation data as Semantic Web data, described by shared vocabularies, it is in line with a challenge raised by Pietriga and Lee (2009) when they discuss visualisation of RDF data. We then propose *covikoa-client*, a generic Web client allowing to exploit this RDF graph to create the corresponding geovisualisation in the form of a Web application. All these elements allow us to provide an implementation of *the map as a knowledge base*, a concept that was introduced by Varanka and Usery (2018). In addition, Varanka and Usery (2018) identify an approach to construct the map as a knowledge base using SPARQL queries as well as an approach based on a browsable graph approach. Our proposal is a combination of both, in that:

- it allows us to extract from our knowledge base, through SPARQL queries, the elements necessary for the map as well as the geovisual semantics that correspond to them,
- it makes it possible to integrate data from other sources to enrich the map (using federated queries, both to exploit linked data according to the specificities of the data currently exploited, as shown in Chapter 7, and to integrate the whole of the data to be represented, as shown in Chapter 8).

This proposal also addresses several challenges that Villanova-Oliver (2018) points out such as the necessity of formalising the correspondence between a concept (from an application, business, or domain ontology) and the geovisual properties that can be associated with it and the fact that it may be useful or necessary to opt for another organisation of concepts than the one used in the original ontology.

Finally, **we test the whole approach on two data models through case studies**. At first we propose an OWL ontology formalising the field of victim search in mountain environments (the *Choucas Alert Ontology*) and enabling the implementation of the CoViKoa framework. We also proposed portrayals for the data described by this model. Then we test our framework on another existing model: the *TSN* and *TSN-change* ontologies (Bernard, 2019) that allow to describe the changes undergone by territorial statistical entities. These two case studies highlight the strengths of our approach: the possibility of combining geometric transformations of the data with various filters and constraints to easily use the semantics of the exploited model, as well as the possibility of defining interactions between the data presented in the components, all this being materialised in a modern looking interface with a flexible layout. Since our work heavily uses W3C technologies for the Semantic Web, we have also noted the existence of various use cases[1] identified by The Spatial Data on

[1] https://www.w3.org/TR/sdw-ucr/

the Web Working Group[2]. We are comforted in the results we obtain by the fact that several of them particularly resonate with ours, such as: *using spatial data during emergency response operations*, *consuming geographical data in a Web application* and *combining spatial RDF data for integrated querying in a triplestore*.

Our case studies also highlight several limitations of our work. Indeed, although we have successfully used *carto* on statistical data (the NUTS 2 data presented in Chapter 5 and reused in Chapter 6 were for example very well suited to implement choropleth or proportional symbol representations in a particularly concise way), we note that *dicopal* and *carto* are not deployed in case studies of Chapter 7 and Chapter 8[3]. This is due to the fact that none of the representations we needed were offered by *carto* and it may also be a signal that work remains to be done on this ontology and its operationalisation. In addition, if *dicopal* encodes interesting and easily queryable knowledge, as shown in chapter 5, its automatic use by a rule system is intrinsically linked to reasoning about the types of cartographic representations. Finally, we are aware of the entry cost of our approach, due to the numerous vocabulary elements provided by our ontologies. In the next section, we discuss these limitations and propose future developments to be undertaken as well as research perspectives.

9.2 Future research and development directions

9.2.1 Development of UI components to support the creation of the Derivation Model

Writing the Derivation Model relies on a declarative approach that eases the tasks of a geovisualisation designer. It involves many vocabularies and must conform to some expectations. We already propose SHACL shapes that allow us to validate the Derivation Model and thus announce the various expectations of our framework regarding the RDF declarations to be included in the Derivation Model.

However, we believe that this is not enough to enable a quick and efficient take-up of our framework. In order to improve this point, we believe that the SHACL shapes we propose could be used to create interface components that support the writing of a valid Derivation Model. Indeed, the possibility of using SHACL shapes to create user interface elements is mentioned in the abstract of the recommendation (Knublauch and Kontokostas, 2017) and we are already using some of the non-validating constraints that exist in SHACL to specify more information about the expected data (such as `sh:lessThanOrEquals` when defining the shape for the properties that qualify a `scale:Scale`).

[2] A joint work between the W3C and the OGC.

[3] However, case study #6, which can be found in the code repository accompanying this manuscript, allows one to see them in action.

Methods for creating components have already been proposed[4] and could be adapted for our use case. To be complete, the joint reading of the Semantic Data Model (SDM) could make it possible to propose to the user the whole of the classes which can be used (Figure 9.1 - the four classes listed in the HTML `select` element appear there because they are subclasses of `geo:Feature` in the SDM). Finally, the interface could greatly ease the creations of the various portrayal rules and symbolizers that constitute a portrayal (Figure 9.2). Such an interface could show the RDF code produced by the various statements entered in a live manner (as shown on Figure 9.3 which corresponds to declarations shown on Figure 9.1).

It should be noted that this interface is not implemented nor in the process of being implemented and that we only present mock-ups of each of the successive panels.

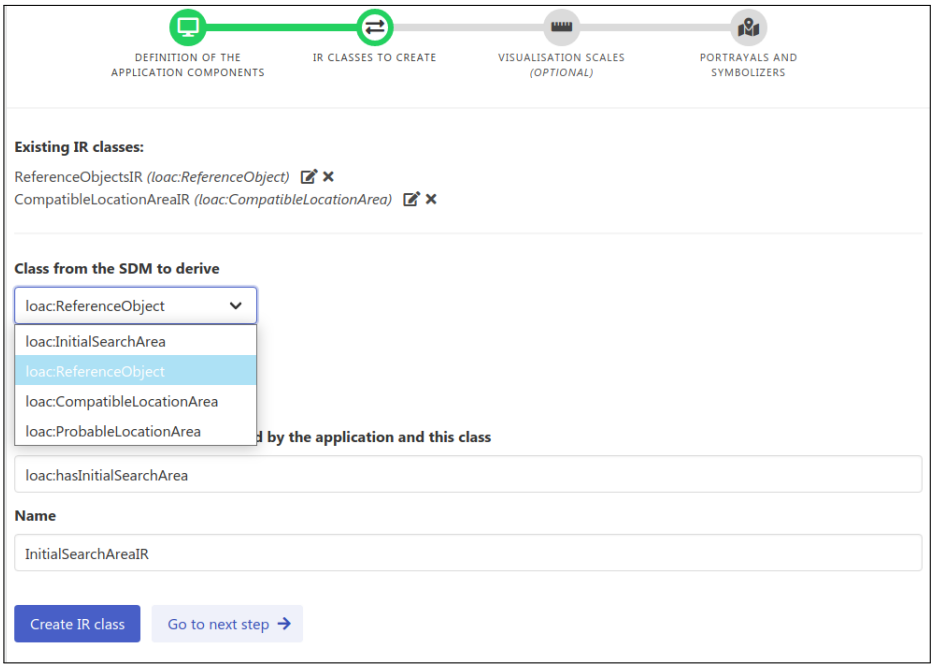

Figure 9.1: Example of user interface for assisting to write the Derivation Model (*IR classes to create* step).

Such a UI will support the creation of the Derivation Model but will also indirectly assist the user in his or her learning of the vocabularies used by CoViKoa. This will be a valuable tool to further validate our proposal as it decreases the entry cost to use our approach and thus could increase its appropriation by geovisualisation designers.

Another perspective could be to work on guidelines on how to use our approach in various contexts. Such a guideline document could for instance contain information about the way an organisation can use CoViKoa in order to publish a geovisualisation

[4] http://datashapes.org/forms.html

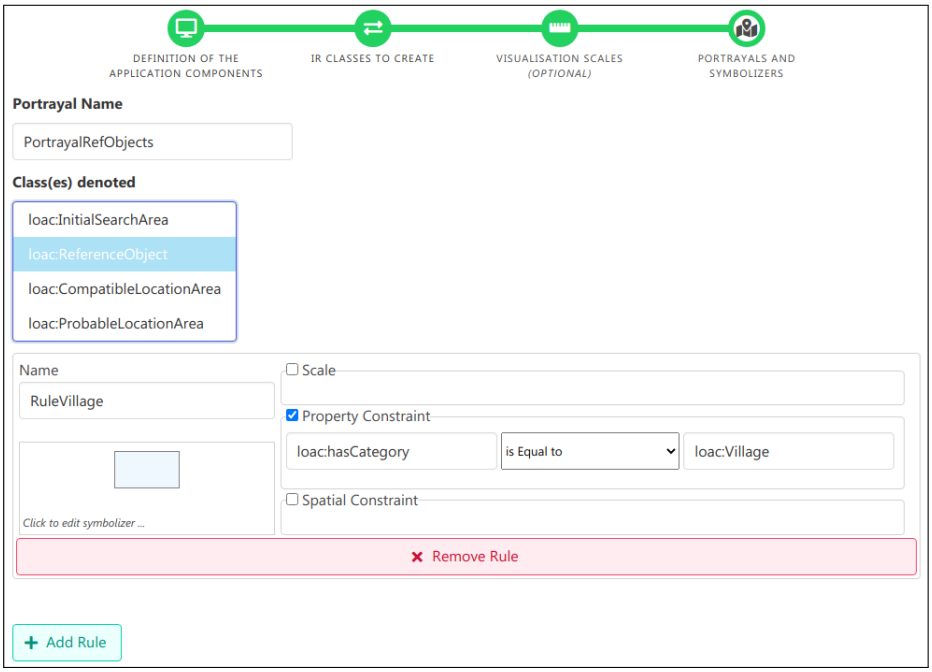

Figure 9.2: Example of user interface for assisting to write the Derivation Model (*Portrayals and symbolizers* step).

Live output (turtle)

```
:ReferenceObjectIR a owl:Class ;
    rdfs:subClassOf gviz:GeoVisualIntermediateRepresentation ;
    cvkd:represents loac:ReferenceObject ;
    cvkd:pathFromGVA (loac:hasInitialSearchArea loac:contains) .

:CompatibleLocationAreaIR a owl:Class ;
    rdfs:subClassOf gviz:GeoVisualIntermediateRepresentation ;
    cvkd:represents loac:CompatibleLocationArea ;
    cvkd:pathFromGVA (loac:hasClue loac:hasCompatibleLocationArea) .

:InitialSearchAreaIR a owl:Class ;
    rdfs:subClassOf gviz:GeoVisualIntermediateRepresentation ;
    cvkd:represents loac:InitialSearchArea ;
    cvkd:pathFromGVA loac:hasInitialSearchArea .
```

Figure 9.3: Example of live output to inform the user about the RDF statements generated .

of its RDF data.

9.2.2 Improved update actions on the CoViKoa graph

The approach presented in this manuscript and in particular the CoViKoa reasoning presented in Chapter 6 uses the unmodified SHACL API. The latter only allows the execution of SHACL-AF rules of type CONSTRUCT whose data have to be added to a new graph (as per the specification[5]). As part of this specification, it is up to the SHACL-AF user to take care of the new graph.

This is what we have done in our approach by unioning the inference graph with the input data graph and re-running the reasoning using this union as the new data graph, until no new triples are created. To do so, we had to take care to avoid the infinite loops that can occur in this approach with suitable `CONSTRUCT` queries (i.e. queries that check with `NOT EXISTS` clause that the data to be created does not already exist). However, this approach has limitations since it is only possible to create new triples but not to remove triples from the original data graph (thus not allowing to update a value either).

In order to show the interest of our approach with the existing official specifications, we presented here our approach based on the official API, in line with the specification. However, it would be interesting to test the use of a modified SHACL API allowing to operate `INSERT` / `DELETE` "inplace" on the data graph used. One the one hand, such a modification (based on the Java SHACL API of TopQuadrant) is trivial to implement in the code. On the other hand, such a possibility would be interesting for the instrumentation of a system like ours, for example by allowing the deletion of IRs and materialisations that correspond to data described by the SDM that have just been deleted from the data graph, whereas this step is currently performed in Java code.

This could for example be interesting from a performance point of view. This is also in line with the lack of a complete rule language for the Semantic Web.

9.2.3 Improved modelling of the legend

Currently the construction of the legend is highly implicit and is done on the client side. Indeed, the user of our approach only specifies that a legend component is linked to a map component. The semantics that we give to this link amounts to displaying on the legend the different elements of the map to which it is linked. Finally, the client-side code exploits the semantics conferred by the cartographic solutions, when used, to construct legends more appropriate to the type of cartographic representation. This is particularly the case for proportional symbol maps (Figure 9.4). Because we

[5] https://www.w3.org/TR/shacl-af/#rules-execution

know that this is such a graphical representation, we can decide to sample and depict 3 values (Figure 9.4, part *A*) that make sense in terms of the distribution of values whereas in other types of representations it would have been expected to represent all the symbolizers mobilised on the map (Figure 9.4, part *B*).

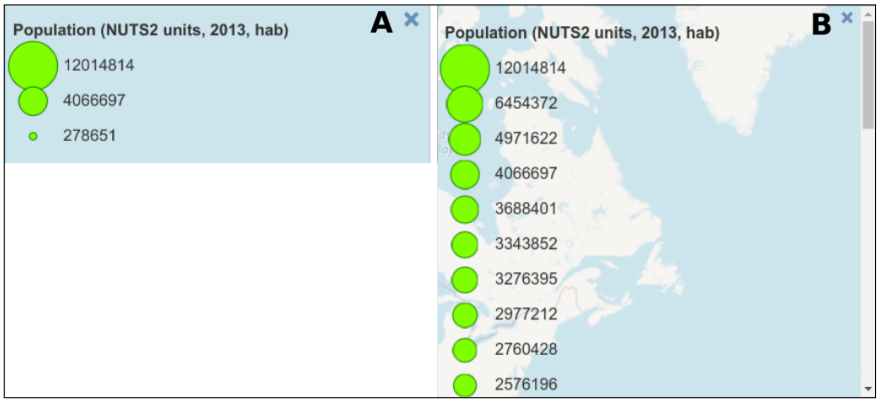

Figure 9.4: Rendering of the legend improved by cartographic knowledge.

Therefore we believe that one of the immediate follow-ups to our work would be to improve the formalisation of the elements of the legend and to automate its construction in the RDF graph so as to avoid its implicit construction on the client side.

This requires several steps. First of all, we need to describe the elements that can appear in a legend using an appropriate ontology. This ontology could be inspired by the work of Fellah (2018) (Figure 9.5). Then it would be necessary to specify in the derivation model which portrayals are intended to appear in the legend. Finally, the creation of a SHACL enrichment rule (using the same mechanism as those already present in the framework) would make it possible to create the various items of the legend. This will make it possible to have different rules depending on the type of representation, making it possible to construct legends using the knowledge specific to the field of cartography. Finally, this SHACL enrichment rule could also establish between each item of the legend and the corresponding materialisation(s) on the map. Doing so could make it possible to easily implement a classical interaction which consists in filtering the elements displayed on the map at the time of a click on one of the data classes presented in the legend.

Another element that is implicit in the legend is its arrangement in the interface. Currently the legend is integrated into the map, with the possibility of reducing it.

A common practice, however, is to display the legend outside the map. This practice allows, for example, to share a legend for several maps. This practice may also be particularly suitable for an interface that may have multiple panels as in our case. Two sub-classes of `gviz:LegendComponent` could thus be created to express this: `gviz:InMapLegendComponent` and `gviz:StandaloneLegendComponent`.

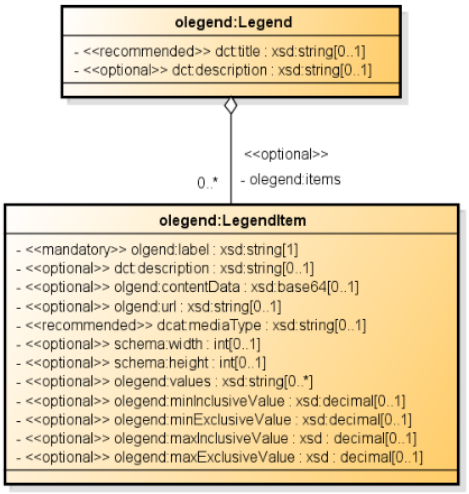

Figure 9.5: Model proposal for describing the legend (Fellah, 2018) . Source: Image from Fellah (2018) .

9.2.4 Allowing the user to choose their portrayals from a catalogue of styles

In the approach that is currently presented, the user of our framework defines the portrayals that will be applied to the data (provided they comply with any specified constraints). This corresponds to the approach that can be used to create thematic maps for example, in the line of what some authors call *rule-based symbolisation* (Iosifescu-Enescu and Hurni, 2007).

However, we have seen in Chapter 4 that proposals exist to describe style catalogues (Fellah, 2017). We think that our framework could mix the two approaches by allowing to describe the possible styles that can be applied to the data and by allowing to create the geovisualisation that would allow the user to choose among the various existing styles, possibly after having specified which one is the default style. The definition of new styles could intervene at different levels, in fact in the case of a choropleth map, one could for example imagine proposing the different appropriate palettes. Since we propose a vocabulary allowing the description of the initial state of the application and which includes the possibility of defining a user profile, it would be possible to filter the styles to be proposed to the user if he or she has a colour perception deficiency (using the fact that this information is stored at the level of the colour palettes which can be used by the representations).

This proposal could be usefully and easily coupled with the one made below.

9.2.5 Towards a recommendation system of appropriate portrayals

In Chapter 5 we proposed *carto*, a vocabulary to describe the different types of cartographic representations, the parameters allowing them to be instantiated, as well as several ways to qualify the data to be represented (according to its spatial implantation, its datatype and its statistical type). We have also established links between the types of cartographic representations and the appropriate data types. However, our system does not exploit this knowledge at any time for the time being.

We believe that we could read the SDM (either the ontological model or the instantiated data) to determine which map representations are possible on a dataset and provide the result as an RDF graph using the *carto* vocabulary. This description could then be transformed, as presented in Chapter 6, into a valid Derivation Model for CoViKoa.

Working on this proposal would have several implications. At first, it would be necessary to formalise new types of map representations in *carto* and to prepare SHACL rules which allow the description of these new representations to be transformed into portrayal rules for CoViKoa. It would then be necessary, and this is in line with the previous subsection, to determine how to describe not one style (for example a choropleth map with 5 classes and the *BuPu* palette from Brewer et al., 2003), but of all the styles that could be appropriate (idem, but with all the palettes suitable for the progression of values in question).

Pursuing such an objective and tackling the issues we have presented above will reinforce the declarative approach we have initiated in this work for an automated and assisted tool that fully exploits knowledge embedded in a semantic data model to infer relevant and adapted geovisualisation of data.

This work was funded by the French National Research Agency
[ANR-16-CE23-0018 CHOUCAS].

Bibliography

Abadie, Nathalie, Ammar Mechouche, and Sébastien Mustière (2010). "OWL-based formalisation of geographic databases specifications". In: *17th International Conference on Knowledge Engineering and Knowledge Management (EKAW'10), Lisbon (Portugal)* (cf. p. 1, 78).

Abi-Zeid, Irène, Oscar Nilo, Stéphane Schvartz, and Michael Morin (2010). "Towards a knowledge-based system prototype for aeronautical Search and Rescue operations". In: *2010 13th International Conference on Information Fusion*. 2010 13th International Conference on Information Fusion (FUSION 2010). Edinburgh: IEEE, pp. 1–8. ISBN: 978-0-9824438-1-1. DOI: 10.1109/ICIF.2010.5712112. URL: http://ieeexplore.ieee.org/document/5712112/ (cf. p. 5, 32, 33).

Allemang, Dean and James A. Hendler (2011). *Semantic Web for the working ontologist: effective modeling in RDFS and OWL*. 2nd ed. OCLC: ocn712780761. Waltham, MA: Morgan Kaufmann/Elsevier. 354 pp. ISBN: 978-0-12-385965-5 (cf. p. 55, 67, 145).

Andrienko, Natalia and Gennady Andrienko (2006). "The complexity challenge to creating useful and usable geovisualization tools". In: *GIScience 4th International Conference on Geographic Information Science (Münster, Germany)* (cf. p. 7).

— (2007). "Intelligent Visualisation and Information Presentation for Civil Crisis Management". In: *Transactions in GIS* 11.6, pp. 889–909. ISSN: 13611682, 14679671. DOI: 10.1111/j.1467-9671.2007.01078.x. URL: http://doi.wiley.com/10.1111/j.1467-9671.2007.01078.x (cf. p. 23).

Andrienko, Natalia, Gennady Andrienko, and Peter Gatalsky (2003). "Exploratory spatio-temporal visualization: an analytical review". In: *Journal of Visual Languages & Computing* 14.6, pp. 503–541. ISSN: 1045926X. DOI: 10.1016/S1045-926X(03)00046-6. URL: https://linkinghub.elsevier.com/retrieve/pii/S1045926X03000466 (cf. p. 108).

Bateman, John A., Joana Hois, Robert Ross, and Thora Tenbrink (2010). "A linguistic ontology of space for natural language processing". In: *Artificial Intelligence* 174.14, pp. 1027–1071. ISSN: 00043702. DOI: 10.1016/j.artint.2010.05.008. URL: https://linkinghub.elsevier.com/retrieve/pii/S0004370210000858 (cf. p. 57).

Beckett, Dave (2004). *RDF/XML Syntax Specification*. W3C Recommendation. World Wide Web Consortium. URL: https://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/ (cf. p. 45).

Beckett, David, Tim Berners-Lee, Eric Prud'hommeaux, and Gavin Carothers (2014). *RDF 1.1 Turtle — Terse RDF Triple Language*. W3C Recommendation. World Wide Web Consortium. URL: https://www.w3.org/TR/turtle/ (cf. p. 49).

Bernard, Camille (2019). "Immersing evolving geographic divisions in the semantic Web". Theses. Université Grenoble Alpes. URL: https://tel.archives-ouvertes.fr/tel-02524361 (cf. p. xii, 93, 157, 189–195, 202, 204, 210).

Bernard, Camille, Jérôme Gensel, Hy Dao, and Marlène Villanova-Oliver (2019). *Territorial Statistical Nomenclature Ontology*. Version 1.1. URL: http://purl.org/net/tsn (cf. p. 190).

Berners-Lee, Tim (July 27, 2006). *Linked Data*. W3C. URL: https://www.w3.org/DesignIssues/LinkedData.html (cf. p. 3, 7).

Berners-Lee, Tim, Dan Connolly, Lalana Kagal, Yosi Scharf, and Jim Hendler (2008). "N3Logic: A logical framework for the World Wide Web". In: *Theory and Practice of Logic Programming* 8.3, pp. 249–269. ISSN: 1471-0684, 1475-3081. DOI: 10.1017/S1471068407003213. URL: https://www.cambridge.org/core/product/identifier/S1471068407003213/type/journal_article (cf. p. 64).

Bertin, Jacques (1967). *Sémiologie graphique : les diagrammes, les réseaux et les cartes*. Paris: Mouton/Gauthier-Villars (cf. p. 15–17).

Bimonte, S. (2014). "A generic geovisualization model for spatial OLAP and its implementation in a standards-based architecture". In: *Ingéniérie des Systèmes d'Information* 5, pp. 97–118. DOI: 10.3166/ISI.19.5.97-118. URL: https://hal.inrae.fr/hal-02600328 (cf. p. 3).

Bojārs, Uldis, Alexandre Passant, JG Breslin, and Stefan Decker (2008). "Social network and data portability using semantic web technologies". In: *Proceedings of the BIS 2008 Workshop on Social Aspects of the Web, Innsbruck, Austria (May 2008)* (cf. p. 1).

Brewer, Cynthia A., Geoffrey W. Hatchard, and Mark A. Harrower (2003). "ColorBrewer in Print: A Catalog of Color Schemes for Maps". In: *Cartography and Geographic Information Science* 30.1, pp. 5–32. ISSN: 1523-0406, 1545-0465. DOI: 10.1559/152304003100010929. URL: http://www.tandfonline.com/doi/abs/10.1559/152304003100010929 (cf. p. 18, 122–124, 209, 217).

Brickley, Dan and R. V. Guha (2004). *RDF Vocabulary Description Language 1.0: RDF Schema*. W3C Recommendation. World Wide Web Consortium. URL: https://www.w3.org/TR/2004/REC-rdf-schema-20040210/ (cf. p. 45).

— (2014). *RDF Schema 1.1*. W3C Recommendation. World Wide Web Consortium. URL: https://www.w3.org/TR/rdf-schema/ (cf. p. 45).

Brink, Linda van den et al. (2018). "Best practices for publishing, retrieving, and using spatial data on the web". In: *Semantic Web* 10.1. Ed. by Pascal Hitzler, pp. 95–114. ISSN: 22104968, 15700844. DOI: 10.3233/SW-180305. URL: https://www.medra.org/servlet/aliasResolver?alias=iospress&doi=10.3233/SW-180305 (cf. p. 1, 2).

Brus, Jan, Dobesova Zdena, Jaromir Kanok, and Vilem Pechanec (2010). "Design of intelligent system in cartography". In: *9th RoEduNet IEEE International Conference*, pp. 112–117 (cf. p. 2, 81, 82, 91, 95, 97, 116, 125, 126, 209).

Bunel, Mattia (2021). "Modélisation et raisonnement spatial flou pour l'aide à la localisation de victimes en montagne". Theses. Université gustave eiffel. URL: https://tel.archives-ouvertes.fr/tel-03298717 (cf. p. 162, 166).

Carral, David, Simon Scheider, Krzysztof Janowicz, Charles Vardeman, Adila A. Krisnadhi, and Pascal Hitzler (2013). "An Ontology Design Pattern for Cartographic Map Scaling". In: *The Semantic Web: Semantics and Big Data*. Ed. by Philipp Cimiano, Oscar Corcho, Valentina Presutti, Laura Hollink, and Sebastian Rudolph. Red. by David Hutchison et al. Vol. 7882. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 76–93. ISBN: 978-3-642-38287-1 978-3-642-38288-8. DOI: 10.1007/978-3-642-38288-8_6. URL: http://link.springer.com/10.1007/978-3-642-38288-8_6 (cf. p. 89, 121, 209).

Cox, Simon and Chris Little (2017). *Time Ontology in OWL*. W3C Recommendation. World Wide Web Consortium. URL: https://www.w3.org/TR/2017/REC-owl-time-20171019/ (cf. p. 57).

Crameri, Fabio (2018). "Geodynamic diagnostics, scientific visualisation and StagLab 3.0". In: *Geoscientific Model Development* 11.6, pp. 2541–2562. ISSN: 1991-9603. DOI: 10.5194/gmd-11-2541-2018. URL: https://www.geosci-model-dev.net/11/2541/2018/ (cf. p. 122, 123, 209).

Crampton, Jeremy W. (2002). "Interactivity Types in Geographic Visualization". In: *Cartography and Geographic Information Science* 29.2, pp. 85–98. ISSN: 1523-0406, 1545-0465. DOI: 10.1559/152304002782053314. URL: http://www.tandfonline.com/doi/abs/10.1559/152304002782053314 (cf. p. 13, 21).

Cyganiak, Richard, David Wood, and Markus Lanthaler (2014). *RDF 1.1 Concepts and Abstract Syntax*. W3C Recommendation. World Wide Web Consortium. URL: https://www.w3.org/TR/rdf11-concepts/ (cf. p. 45).

Daga, Enrico, Albert Meroño Peñuela, and Enrico Motta (2019). "Modelling and querying lists in RDF: A pragmatic study". In: *QuWeDa 2019*. Ed. by Muhammad Saleem, Aidan Hogan, Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, and Ruben Verborgh. CEUR Workshop Proceedings. CEUR-WS, pp. 21–36 (cf. p. 122).

Dean, Mike and Guus Schreiber (2004). *OWL Web Ontology Language Reference*. W3C Recommendation. World Wide Web Consortium. URL: https://www.w3.org/TR/2004/REC-owl-ref-20040210/ (cf. p. 52).

Dent, Borden D., Jeffrey Torguson, and T. W. Hodler (2009). *Cartography: thematic map design*. 6th ed. OCLC: ocn184827987. New York: McGraw-Hill Higher Education. 336 pp. ISBN: 978-0-07-294382-5 (cf. p. 15, 17).

Desimone, Robert and John Duncan (1995). "Neural Mechanisms of Selective Visual Attention". In: *Annual Review of Neuroscience* 18.1, pp. 193–222. ISSN: 0147-006X, 1545-4126. DOI: 10.1146/annurev.ne.18.030195.001205. URL: http://www.annualreviews.org/doi/10.1146/annurev.ne.18.030195.001205 (cf. p. 27).

Dong, Weihua, Jiping Liu, and Qingsheng Guo (2008). "Using ontology for map personalization visualization application". In: *Proceedings of the International Conference on Earth Observation Data Processing and Analysis (ICEODPA)*. International Conference on Earth Observation Data Processing and Analysis. Ed. by Deren Li, Jianya Gong, and Huayi Wu. Vol. 7285. Wuhan, China, 72853P. DOI: 10.1117/12.815832. URL: http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.815832 (cf. p. 83, 84).

Durkee, George and Vanessa Glynn-Linaris (2012). *Using GIS for Wildland Search and Rescue*. ESRI Press. Redcands, CA, USA. 200 pp. (cf. p. 32).

Dykes, Jason, Alan M. MacEachren, and Menno-Jan Kraak, eds. (2007). *Exploring geovisualization*. Reprinted. OCLC: 836882461. Amsterdam: Elsevier. 710 pp. ISBN: 978-0-08-044531-1 978-0-08-044533-5 (cf. p. 13, 19, 23, 112).

Fabrikant, Sara (2001). "Building Task-Ontologies for GeoVisualization". In: *Pre-Conference Workshop on Geovisualization on the Web*. 20th International Cartographic Conference, ICA Commission on Visualization and Virtual Environments. Beijing, China. URL: https://www.researchgate.net/publication/2381464_Building_Task-Ontologies_for_GeoVisualization (cf. p. 2, 78, 93).

Fellah, Stephane (2017). *OGC Testbed-12 Semantic Portrayal, Registry and Mediation*. OGC 16-059. Open Geospatial Consortium. URL: http://docs.opengeospatial.org/per/16-059.html (cf. p. 85, 86, 89, 95–97, 101, 107, 115, 116, 209, 216).

— (2018). *OGC Testbed-13 Portrayal Engineering Report*. OGC 17-045. Open Geospatial Consortium. URL: http://docs.opengeospatial.org/per/16-059.html (cf. p. 215, 216).

Gabbard, J.L., D. Hix, and J.E. Swan (1999). "User-centered design and evaluation of virtual environments". In: *IEEE Computer Graphics and Applications* 19.6, pp. 51–59. ISSN: 02721716. DOI: 10.1109/38.799740. URL: http://ieeexplore.ieee.org/document/799740/ (cf. p. 38).

Gandon, Fabien and Guus Schreiber (2014). *RDF 1.1 XML Syntax*. W3C Recommendation. World Wide Web Consortium. URL: https://www.w3.org/TR/rdf-syntax-grammar/ (cf. p. 45).

Garbis, George, Kostis Kyzirakos, and Manolis Koubarakis (2013). "Geographica: A Benchmark for Geospatial RDF Stores (Long Version)". In: *Advanced Information Systems Engineering*. Ed. by Camille Salinesi, Moira C. Norrie, and Óscar Pastor. Red. by David Hutchison et al. Vol. 7908. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 343–359. ISBN: 978-3-642-38708-1 978-3-642-38709-8. DOI: 10.1007/978-3-642-41338-4_22. URL: http://link.springer.com/10.1007/978-3-642-41338-4_22 (cf. p. 67).

Glimm, Birte, Aidan Hogan, Markus Krötzsch, and Axel Polleres (2012). "OWL: Yet to arrive on the Web of Data?" In: *CEUR Workshop Proceedings* 937 (cf. p. 55).

Grant, Jan and Dave Beckett (2004). *RDF Test Cases*. W3C Recommendation. World Wide Web Consortium. URL: https://www.w3.org/TR/2004/REC-rdf-testcases-20040210/ (cf. p. 45, 49).

Gruber, Thomas R. (1995). "Toward principles for the design of ontologies used for knowledge sharing?" In: *International Journal of Human-Computer Studies* 43.5, pp. 907–928. ISSN: 10715819. DOI: 10.1006/ijhc.1995.1081. URL: https://linkinghub.elsevier.com/retrieve/pii/S1071581985710816 (cf. p. 44).

Halik, Łukasz (2012). "The analysis of visual variables for use in the cartographic design of point symbols for mobile Augmented Reality applications". In: *Geodesy and Cartography* 61.1, pp. 19–30. ISSN: 2080-6736. DOI: 10.2478/v10277-012-0019-4. URL: http://journals.pan.pl/dlibra/publication/113245/edition/98326/content (cf. p. 17).

Harris, Steve and Andy Seaborne (2013). *SPARQL 1.1 Query Language*. W3C Recommendation. World Wide Web Consortium. URL: https://www.w3.org/TR/sparql11-query/ (cf. p. 59).

Hayes, Patrick (2004). *RDF Semantics*. W3C Recommendation. World Wide Web Consortium. URL: http://www.w3.org/TR/2004/REC-rdf-mt-20040210 (cf. p. 45, 53).

Hayes, Patrick J. and Peter F. Patel-Schneider (2014). *RDF 1.1 Semantics*. W3C Recommendation. World Wide Web Consortium. URL: https://www.w3.org/TR/rdf11-mt/ (cf. p. 45).

Hebeler, John, ed. (2009). *Semantic web programming*. Timely. Practical. Reliable. OCLC: 261175439. Indianapolis, Ind: Wiley. 616 pp. ISBN: 978-0-470-41801-7 (cf. p. 67, 75).

Heer, Jeffrey and Maneesh Agrawala (2006). "Software Design Patterns for Information Visualization". In: *IEEE Transactions on Visualization and Computer Graphics* 12.5, pp. 853–860. ISSN: 1077-2626. DOI: 10.1109/TVCG.2006.178. URL: http://ieeexplore.ieee.org/document/4015439/ (cf. p. 131, 132).

Hitzler, Pascal, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, and Sebastian Rudolph (2012). *OWL 2 Web Ontology Language: Primer (Second Edition)*. W3C Recommendation. World Wide Web Consortium. URL: https://www.w3.org/TR/2012/REC-owl2-primer-20121211/ (cf. p. 52, 54).

Hogan, Aidan (2020). *The Web of Data*. Cham: Springer International Publishing. ISBN: 978-3-030-51579-9 978-3-030-51580-5. DOI: 10.1007/978-3-030-51580-5. URL: http://link.springer.com/10.1007/978-3-030-51580-5 (cf. p. 44, 46, 48).

Horrocks et al. (2004). "SWRL: A Semantic Web rule language combining OWL and RuleML". In: *W3C Subm* 21 (cf. p. 64).

Houda, Mnasser, Maha Khemaja, Kathia Oliveira, and Mourad Abed (2010). "A public transportation ontology to support user travel planning". In: *2010 Fourth International Conference on Research Challenges in Information Science (RCIS)*. 2010 Fourth International Conference on Research Challenges in Information Science (RCIS). Nice, France: IEEE, pp. 127–136. ISBN: 978-1-4244-4839-5. DOI: 10.1109/RCIS.2010.5507372. URL: http://ieeexplore.ieee.org/document/5507372/ (cf. p. 1).

Hu, Yingjie (2018). "Geospatial Semantics". In: *Comprehensive Geographic Information Systems*. Elsevier, pp. 80–94. ISBN: 978-0-12-804793-4. DOI: 10.1016/B978-0-12-409548-9.09597-X. URL: https://linkinghub.elsevier.com/retrieve/pii/B978012409548909597X (cf. p. 2, 78).

Huang, Weiming and Lars Harrie (2020). "Towards knowledge-based geovisualisation using Semantic Web technologies: a knowledge representation approach coupling ontologies and rules". In: *International Journal of Digital Earth* 13.9, pp. 976–997. ISSN: 1753-8947. DOI: 10.1080/17538947.2019.1604835. URL: https://doi.org/10.1080/17538947.2019.1604835 (cf. p. 2, 3, 89–91, 95–97, 101, 102, 105, 107, 114, 115, 121, 207, 209).

Huang, Weiming, Khashayar Kazemzadeh, Ali Mansourian, and Lars Harrie (2020). "Towards Knowledge-Based Geospatial Data Integration and Visualization: A Case of Visualizing Urban Bicycling Suitability". In: *IEEE Access* 8, pp. 85473–85489. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2992023. URL: https://ieeexplore.ieee.org/document/9085369/ (cf. p. 2, 3, 91, 92, 96, 97, 123, 125, 207, 209).

Huang, Weiming, Ali Mansourian, Ehsan Abdolmajidi, Haiqi Xu, and Lars Harrie (2018). "Synchronising geometric representations for map mashups using relative positioning and Linked Data". In: *International Journal of Geographical Information Science* 32.6, pp. 1117–1137. ISSN: 1365-8816, 1362-3087. DOI: 10.1080/13658816.2018.1441416. URL: https://www.tandfonline.com/doi/full/10.1080/13658816.2018.1441416 (cf. p. 31, 84, 89).

Huang, Weiming, Syed Amir Raza, Oleg Mirzov, and Lars Harrie (2019). "Assessment and Benchmarking of Spatially Enabled RDF Stores for the Next Generation of Spatial Data Infrastructure". In: *ISPRS International Journal of Geo-Information* 8.7, p. 310. ISSN: 2220-9964. DOI: 10.3390/ijgi8070310. URL: https://www.mdpi.com/2220-9964/8/7/310 (cf. p. 67, 75).

Iosifescu-Enescu, Ionuț and Lorenz Hurni (2007). "Towards cartographic ontologies or" how computers learn cartography"". In: *Proceedings 23rd International Cartographic Conference*, pp. 4–10. URL: https://icaci.org/files/documents/ICC_proceedings/ICC2007/documents/doc/THEME%201/oral%205/1.5.3%20TOWARDS%20CARTOGRAPHIC%20ONTOLOGIES%20OR%20%E2%80%9CHOW%20COMPUTERS%20LEAR.doc (cf. p. 2, 79, 80, 83, 95–97, 114, 125, 126, 156, 209, 216).

Janowicz, Krzysztof, Simon Scheider, and Benjamin Adams (2013). "A Geo-semantics Flyby". In: *Reasoning Web. Semantic Technologies for Intelligent Data Access: 9th International Summer School 2013, Mannheim, Germany, July 30 – August 2, 2013. Proceedings*. Ed. by Sebastian Rudolph, Georg Gottlob, Ian Horrocks, and Frank van Harmelen. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 230–250. ISBN: 978-3-642-39784-4. DOI: 10.1007/978-3-642-39784-4_6. URL: https://doi.org/10.1007/978-3-642-39784-4_6 (cf. p. 2, 78).

Jaspers, M, T Steen, C Bos, and M Geenen (2004). "The think aloud method: a guide to user interface design". In: *International Journal of Medical Informatics* 73.11, pp. 781–795. ISSN: 13865056. DOI: 10.1016/j.ijmedinf.2004.08.003. URL: https://linkinghub.elsevier.com/retrieve/pii/S1386505604001820 (cf. p. 208).

Johnson, Lanny (2004). "An introduction to mountain search and rescue". In: *Emergency Medicine Clinics of North America* 22.2, pp. 511–524. ISSN: 07338627. DOI: 10.1016/j.emc.2004.01.010. URL: https://linkinghub.elsevier.com/retrieve/pii/S0733862704000112 (cf. p. 4).

Kellogg, Gregg and Markus Lanthaler (2014). *RDF 1.1 Test Cases*. W3C Working Group Note. World Wide Web Consortium. URL: https://www.w3.org/TR/rdf11-testcases/ (cf. p. 45).

Klyne, Graham and Jeremy J. Carroll (2004). *Resource Description Framework (RDF): Concepts and Abstract Syntax*. W3C Recommendation. World Wide Web Consortium. URL: http://www.w3.org/TR/2004/REC-rdf-concepts-20040210 (cf. p. 45).

Knublauch, Holger, Dean Allemang, and Simon Steyskal (2017). *SHACL Advanced Features*. W3C Working Group Note. World Wide Web Consortium. URL: https://www.w3.org/TR/shacl-af/ (cf. p. 65).

Knublauch, Holger, James A. Hendler, and Kingsley Idehen (2011). *SPIN - Overview and Motivation*. W3C Member Submission. Published: W3C Member Submission. World Wide Web Consortium. URL: http://www.w3.org/Submission/2011/SUBM-spin-overview-20110222/ (cf. p. 65).

Knublauch, Holger and Dimitris Kontokostas (2017). *Shapes Constraint Language (SHACL)*. W3C Recommendation. World Wide Web Consortium. URL: https://www.w3.org/TR/shacl/ (cf. p. 62, 211).

Knublauch, Holger and Pano Maria (2017). *SHACL JavaScript Extensions*. W3C Working Group Note. World Wide Web Consortium. URL: https://www.w3.org/TR/shacl-js/ (cf. p. 65).

Konstantopoulos, Stasinos, Georgios Paliouras, and Symeon Chatzinotas (2006). "SHARE-ODS: An Ontology Data Service for Search and Rescue Operations". In: *Advances in Artificial Intelligence*. Ed. by Grigoris Antoniou, George Potamias, Costas Spyropoulos, and Dimitris Plexousakis. Vol. 3955. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 525–528. ISBN: 978-3-540-34117-8 978-3-540-34118-5. DOI: 10.1007/11752912_60. URL: http://link.springer.com/10.1007/11752912_60 (cf. p. 5).

Konstantopoulos, Stasinos, Jens Pottebaum, Jochen Schon, Daniel Schneider, Thomas Winkler, Georgios Paliouras, and Rainer Koch (2009). "Ontology-Based Rescue Operation Management". In: *Mobile Response*. Ed. by Jobst Löffler and Markus Klann. Vol. 5424. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 112–121. ISBN: 978-3-642-00439-1 978-3-642-00440-7. DOI: 10.1007/978-3-642-00440-7_11. URL: http://link.springer.com/10.1007/978-3-642-00440-7_11 (cf. p. 5).

Kraak, M. J. and Ferjan Ormeling (2010). *Cartography: visualization of geospatial data*. 3rd ed. OCLC: ocn477062041. Harlow ; New York: Prentice Hall. 198 pp. ISBN: 978-0-273-72279-3 (cf. p. 15, 17, 19, 23).

Krygier, John and Denis Wood (2005). *Making maps: a visual guide to map design for GIS*. New York: Guilford Press. 303 pp. ISBN: 978-1-59385-200-9 (cf. p. 15, 17).

Lambert, Nicolas and Christine Zanin (2020). *Practical handbook of thematic cartography: principles, methods, and applications*. OCLC: 1164619833. ISBN: 978-0-367-26129-0 (cf. p. 14, 15, 17, 19, 20, 125).

Lamy, Jean-Baptiste (2017). "Représentation, iconisation et visualisation des connaissances : Principes et applications à l'aide à la décision médicale". Habilitation à diriger des recherches. Normandie Université, Université de Rouen. URL: https://hal.archives-ouvertes.fr/tel-01849832 (cf. p. 1).

Lassila, Ora and Ralph R. Swick (1999). *Resource Description Framework (RDF) Model and Syntax Specification*. W3C Recommendation. World Wide Web Consortium. URL: https://www.w3.org/TR/1999/REC-rdf-syntax-19990222/ (cf. p. 45).

Lee, Jae-Gil and Minseo Kang (2015). "Geospatial Big Data: Challenges and Opportunities". In: *Big Data Research* 2.2, pp. 74–81. ISSN: 22145796. DOI: 10.1016/j.bdr.2015.01.003. URL: https://linkinghub.elsevier.com/retrieve/pii/S2214579615000040 (cf. p. 1).

León, Alexander de, Filip Wisniewki, Boris Villazón-Terrazas, and Oscar Corcho (2012). "Map4rdf - Faceted Browser for Geospatial Datasets". In: *PMOD workshop* (cf. p. 2, 87, 95, 96, 207).

Lieberman, Joshua, Raj Singh, and Chris Goad (2007). *W3C Geospatial Vocabulary*. W3C Incubator Group Report. World Wide Web Consortium. URL: https://www.w3.org/2005/Incubator/geo/XGR-geo-20071023/ (cf. p. 56).

Light, Adam and Patrick J. Bartlein (2004). "The end of the rainbow? Color schemes for improved data graphics". In: *Eos, Transactions American Geophysical Union* 85.40, p. 385. ISSN: 0096-3941. DOI: 10.1029/2004EO400002. URL: http://doi.wiley.com/10.1029/2004EO400002 (cf. p. 18, 122, 209).

Liu, Zhen, Huadong Guo, and Changlin Wang (2016). "Considerations on Geospatial Big Data". In: *IOP Conference Series: Earth and Environmental Science* 46, p. 012058. ISSN: 1755-1307, 1755-1315. DOI: 10.1088/1755-1315/46/1/012058. URL: https://iopscience.iop.org/article/10.1088/1755-1315/46/1/012058 (cf. p. 1).

Lohmann, Steffen, Stefan Negru, Florian Haag, and Thomas Ertl (2016). "Visualizing Ontologies with VOWL". In: *Semantic Web* 7.4, pp. 399–419. DOI: 10.3233/SW-150200. URL: http://dx.doi.org/10.3233/SW-150200 (cf. p. 53).

Löffler, Jobst, V Hernández Ernst, Jochen Schon, Jens Pottebaum, and Rainer Koch (2007). "Intelligent use of geospatial information for emergency operation management". In: *Proceedings of the fourth international conference on information systems for crisis management, ISCRAM*. Delft, the Netherlands (cf. p. 29, 30).

MacEachren, Alan M. (1995). "How maps work: representation, visualization and design". In: *How maps work*. Guilford Press, NY, distributed by Longman outside N. America. ISBN: 0-89862-589-0 (cf. p. 15, 17).

Maceachren, Alan M., Monica Wachowicz, Robert Edsall, Daniel Haug, and Raymon Masters (1999). "Constructing knowledge from multivariate spatiotemporal data: integrating geographical visualization with knowledge discovery in database methods". In: *International Journal of Geographical Information Science* 13.4, pp. 311–334. ISSN: 1365-8816, 1362-3087. DOI: 10.1080/136588199241229. URL: http://www.tandfonline.com/doi/abs/10.1080/136588199241229 (cf. p. 108).

MacEachren, A.M., M. Gahegan, W. Pike, I. Brewer, G. Cai, E. Lengerich, and F. Hardistry (2004). "Geovisualization for knowledge construction and decision support". In: *IEEE Computer Graphics and Applications* 24.1, pp. 13–17. ISSN: 0272-1716. DOI: 10.1109/MCG.2004.1255801. URL: http://ieeexplore.ieee.org/document/1255801/ (cf. p. 2).

Manola, Frank and Eric Miller (2004). *RDF Primer*. W3C Recommendation. World Wide Web Consortium. URL: http://www.w3.org/TR/2004/REC-rdf-primer-20040210 (cf. p. 45).

Marven, Cindy A., Rosaline R. Canessa, and Peter Keller (2007). "Exploratory Spatial Data Analysis to Support Maritime Search and Rescue Planning". In: *Geomatics Solutions for Disaster Management*. Ed. by Jonathan Li, Sisi Zlatanova, and Andrea G. Fabbri. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 271–288. ISBN: 978-3-540-72108-6. DOI: 10.1007/978-3-540-72108-6_18. URL: https://doi.org/10.1007/978-3-540-72108-6_18 (cf. p. 5).

Maué, Patrick and Sven Schade (2009). "Data Integration in the Geospatial Semantic Web:" in: *Journal of Cases on Information Technology* 11.4, pp. 100–122. ISSN: 1548-7717, 1548-7725. DOI: 10.4018/jcit.2009072105. URL: http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/jcit.2009072105 (cf. p. 1, 78).

Menin, Aline (2020). "eSTIMe: a visualization framework for assisting a multi-perspective analysis of daily mobility data". Theses. Grenoble UGA - Université Grenoble Alpes. URL: https://hal.archives-ouvertes.fr/tel-03081054 (cf. p. 26).

Mericskay, Boris (2016). "La cartographie à l'heure du Géoweb : Retour sur les nouveaux modes de représentation spatiale des données numériques". In: *Cartes & géomatique*. La sémiologie dans tous les sens – Temps, Art & Cartographie 229-230, pp. 37–50. URL: https://halshs.archives-ouvertes.fr/halshs-01468314 (cf. p. 20, 21, 29).

Michal, Wysokinski and Marcjan Robert (2015). "DECISION SUPPORT SYSTEM FOR SEARCH & RESCUE OPERATIONS". In: *Computer Science* 16.3, p. 281. ISSN: 1508-2806. DOI: 10.7494/csci.2015.16.3.281. URL: http://journals.agh.edu.pl/csci/article/view/1344 (cf. p. 5, 32).

Miller, Harvey J. and Michael F. Goodchild (2015). "Data-driven geography". In: *GeoJournal* 80.4, pp. 449–461. ISSN: 0343-2521, 1572-9893. DOI: 10.1007/s10708-014-9602-6. URL: http://link.springer.com/10.1007/s10708-014-9602-6 (cf. p. 1).

Moisuc, Bogdan (2007). "Conception et mise en œuvre de systèmes d'information spatio-temporelle adaptatifs : le framework ASTIS". PhD thesis. Grenoble: Université Joseph Fourier (cf. p. 3).

Monmonier, Mark (2007). "Cartography: the multidisciplinary pluralism of cartographic art, geospatial technology, and empirical scholarship". In: *Progress in Human Geography* 31.3, pp. 371–379. ISSN: 0309-1325, 1477-0288. DOI: 10.1177/0309132507077089. URL: http://journals.sagepub.com/doi/10.1177/0309132507077089 (cf. p. 20).

Montenegro, Nuno, Jorge C. Gomes, Paulo Urbano, and José P. Duarte (2012). "A Land Use Planning Ontology: LBCS". In: *Future Internet* 4.1, pp. 65–82. ISSN: 1999-5903. DOI: 10.3390/fi4010065. URL: http://www.mdpi.com/1999-5903/4/1/65 (cf. p. 1).

Morrison, Joel L (1974). "A theoretical framework for cartographic generalization with the emphasis on the process of symbolization". In: *International Yearbook of Cartography* 14.1974, pp. 115–27 (cf. p. 15, 17).

Motik, Boris, Peter F. Patel-Schneider, and Bernardo Cuenca Grau (2012). *OWL 2 Web Ontology Language: Direct Semantics (Second Edition)*. W3C Recommendation. World Wide Web Consortium. URL: https://www.w3.org/TR/2012/REC-owl2-direct-semantics-20121211/ (cf. p. 52).

Nakajima, Miho, L.Ian Schmitt, and Michael M. Halassa (2019). "Prefrontal Cortex Regulates Sensory Filtering through a Basal Ganglia-to-Thalamus Pathway". In: *Neuron* 103.3, 445–458.e10. ISSN: 08966273. DOI: 10.1016/j.neuron.2019.05.026. URL: https://linkinghub.elsevier.com/retrieve/pii/S0896627319304799 (cf. p. 27).

Neuville, R., F. Poux, P. Hallot, and R. Billen (2016). "TOWARDS A NORMALISED 3D GEOVISUALISATION:THE VIEWPOINT MANAGEMENT". In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* IV-2/W1, pp. 179–186. ISSN: 2194-9050. DOI: 10.5194/isprs-annals-IV-2-W1-179-2016. URL: https://www.isprs-ann-photogramm-remote-sens-spatial-inf-sci.net/IV-2-W1/179/2016/ (cf. p. 17).

Noël, David, Marlène Villanova-Oliver, Jérôme Gensel, and Pierre Le Quéau (2017). "Design Patterns for Modelling Life Trajectories in the Semantic Web". In: *Web and Wireless Geographical Information Systems*. Ed. by David Brosset, Christophe Claramunt, Xiang Li, and Tianzhen Wang. Vol. 10181. Cham: Springer Inter-

national Publishing, pp. 51–65. ISBN: 978-3-319-55997-1 978-3-319-55998-8. DOI: 10.1007/978-3-319-55998-8_4. URL: http://link.springer.com/10.1007/978-3-319-55998-8_4 (cf. p. 1).

NRC, National Research Council (2007). *Successful Response Starts with a Map: Improving Geospatial Support for Disaster Management*. Washington, D.C.: National Academies Press. ISBN: 978-0-309-10340-4. DOI: 10.17226/11793. URL: http://www.nap.edu/catalog/11793 (cf. p. 5).

OGC® (2007). *OpenGIS Styled Layer Descriptor Profile of the Web Map Service Implementation Specification 1.1.0*. OGC® Implementation Specification 05-078r4. URL: https://www.ogc.org/standards/sld (cf. p. 3, 41, 209).

— (2012). *OGC GeoSPARQL - A Geographic Query Language for RDF Data*. OGC® Standard OGC 11-052r4. Open Geospatial Consortium. URL: https://www.ogc.org/standards/geosparql (cf. p. 1, 56).

Okabe, Masataka and Kei Ito (Nov. 20, 2002). *Color Universal Design (CUD) - How to make figures and presentations that are friendly to Colorblind people*. URL: https://jfly.uni-koeln.de/color/ (cf. p. 122, 123, 209).

Olteanu-Raimond, Ana-Maria, Sébastien Mustière, Mattia Bunel, Catherine Domingues, Cécile Duchêne, Laurence Jolivet, and Marie-Dominique Van-Damme (2020). *Spatial reference objects ontology*. Version 1.0.1. URL: http://purl.org/choucas.ign.fr/oor (cf. p. 94).

Olteanu-Raimond, Ana-Maria et al. (2017). "Projet CHOUCAS : Intégration de données hétérogènes et raisonnement spatial pour l'aide à la localisation des victimes en montagne". In: *Spatial Analysis and GEOmatics 2017*. Rouen, France: INSA de rouen. URL: https://hal.archives-ouvertes.fr/hal-01649156 (cf. p. 3, 159).

Peirce, Charles Sanders (1907). *Collected Papers of Charles Sanders Peirce - Pragmatism*. The Essential Peirce Reprinted (cf. p. 15).

Penaz, Tomas, Radek Dostal, Isik Yilmaz, and Marian Marschalko (2014). "Design and construction of knowledge ontology for thematic cartography domain". In: *Episodes* 37.1, pp. 48–58. ISSN: 0705-3797, 2586-1298. DOI: 10.18814/epiiugs/2014/v37i1/006. URL: http://www.episodes.org/journal/view.html?doi=10.18814/epiiugs/2014/v37i1/006 (cf. p. 2, 81, 83, 95, 125, 126, 142).

Pietriga, Emmanuel, Chris Bizer, David Karger, and Ryan Lee (2006). "Fresnel - A Browser-Independent Presentation Vocabulary for RDF". In: *Lecture Notes in Computer Science (LNCS 4273), Proceedings of the 5th International Semantic Web Conference (ISWC 2006)*. Springer, pp. 158–171. DOI: http://dx.doi.org/10.1007/11926078_12 (cf. p. 131, 132).

Pietriga, Emmanuel and Ryan Lee (2009). "Langages et outils pour la visualisation et la manipulation de données du web sémantique". In: *Techniques et sciences informatiques* 28.2, pp. 173–197. ISSN: 07524072. DOI: 10.3166/tsi.28.173-197. URL: http://tsi.revuesonline.com/article.jsp?articleId=12995 (cf. p. 2, 210).

Prudhomme, Claire, Timo Homburg, Jean-Jacques Ponciano, Frank Boochs, Christophe Cruz, and Ana-Maria Roxin (2020). "Interpretation and automatic integration of geospatial data into the Semantic Web: Towards a process of automatic geospatial data interpretation, classification and integration using semantic technologies". In: *Computing* 102.2, pp. 365–391. ISSN: 0010-485X, 1436-5057. DOI: 10.1007/

s00607-019-00701-y. URL: http://link.springer.com/10.1007/s00607-019-00701-y (cf. p. 1, 78).

Prud'hommeaux, Eric, Iovka Boneva, Jose Emilio Labra Gayo, and Gregg Kellogg (2019). *Shape Expressions Language 2.1*. Final Community Group Report. World Wide Web Consortium. URL: http://shex.io/shex-semantics/ (cf. p. 63).

Prud'hommeaux, Eric and Carlos Buil-Aranda (2013). *SPARQL 1.1 Federated Query*. W3C Recommendation. World Wide Web Consortium. URL: http://www.w3.org/TR/2013/REC-sparql11-federated-query-20130321/ (cf. p. 59).

Prud'hommeaux, Eric and Andy Seaborne (2008). *SPARQL Query Language for RDF*. W3C Recommendation. World Wide Web Consortium. URL: https://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/ (cf. p. 59).

Rafes, Karima and Cécile Germain (2015). "A platform for scientific data sharing". In: *BDA2015 - Bases de Données Avancées*. Île de Porquerolles, France. URL: https://hal.inria.fr/hal-01168496 (cf. p. 1).

Rice, James, Adam Farquhar, Philippe Piernot, and Thomas Gruber (1996). "Using the Web instead of a window system". In: *Proceedings of the SIGCHI conference on Human factors in computing systems common ground - CHI '96*. the SIGCHI conference. Vancouver, British Columbia, Canada: ACM Press, pp. 103–110. ISBN: 978-0-89791-777-3. DOI: 10.1145/238386.238442. URL: http://portal.acm.org/citation.cfm?doid=238386.238442 (cf. p. 37).

Riveiro, Maria (2016). "Visually supported reasoning under uncertain conditions: Effects of domain expertise on air traffic risk assessment". In: *Spatial Cognition & Computation* 16.2, pp. 133–153. ISSN: 1387-5868, 1542-7633. DOI: 10.1080/13875868.2015.1137576. URL: http://www.tandfonline.com/doi/full/10.1080/13875868.2015.1137576 (cf. p. 5).

Robinson, Anthony C. (2011). "Highlighting in Geovisualization". In: *Cartography and Geographic Information Science* 38.4, pp. 373–383. ISSN: 1523-0406, 1545-0465. DOI: 10.1559/15230406384373. URL: http://www.tandfonline.com/doi/abs/10.1559/15230406384373 (cf. p. 26, 28, 112, 200).

Robinson, Anthony C., Jin Chen, Eugene J. Lengerich, Hans G. Meyer, and Alan M. MacEachren (2005). "Combining Usability Techniques to Design Geovisualization Tools for Epidemiology". In: *Cartography and Geographic Information Science* 32.4, pp. 243–255. ISSN: 1523-0406, 1545-0465. DOI: 10.1559/152304005775194700. URL: http://www.tandfonline.com/doi/abs/10.1559/152304005775194700 (cf. p. 38, 132).

Robinson, Anthony C., Urška Demšar, Antoni B. Moore, Aileen Buckley, Bin Jiang, Kenneth Field, Menno-Jan Kraak, Silvana P. Camboim, and Claudia R. Sluter (2017). "Geospatial big data and cartography: research challenges and opportunities for making maps that matter". In: *International Journal of Cartography* 3 (sup1), pp. 32–60. ISSN: 2372-9333, 2372-9341. DOI: 10.1080/23729333.2016.1278151. URL: https://www.tandfonline.com/doi/full/10.1080/23729333.2016.1278151 (cf. p. 23).

Roth, Robert (2011). "Interacting With Maps: The Science And Practice Of Cartographic Interaction". PhD thesis. URL: https://etda.libraries.psu.edu/catalog/12433 (cf. p. 21).

Roth, Robert E (2012). "Cartographic Interaction Primitives: Framework and Synthesis". In: *The Cartographic Journal* 49.4, pp. 376–395. ISSN: 0008-7041, 1743-2774.

DOI: 10.1179/1743277412Y.0000000019. URL: http://www.tandfonline.com/doi/full/10.1179/1743277412Y.0000000019 (cf. p. 22, 108, 112, 209).

Roth, Robert E., Richard G. Donohue, Carl M. Sack, Timothy R. Wallace, and Tanya M. A. Buckingham (2015). "A Process for Keeping Pace with Evolving Web Mapping Technologies". In: *Cartographic Perspectives* 78, pp. 25–52. ISSN: 1048-9053. DOI: 10.14714/CP78.1273. URL: https://cartographicperspectives.org/index.php/journal/article/view/cp78-roth-et-al (cf. p. 37).

Ruzicka, Jan, Katerina Ruzickova, and Radek Dostal (2013). "Expert System for Cartography Based on Ontology". In: *2013 Fourth Global Congress on Intelligent Systems*. 2013 Fourth Global Congress on Intelligent Systems (GCIS). Hong Kong: IEEE, pp. 159–163. ISBN: 978-1-4799-2885-9 978-1-4799-2886-6. DOI: 10.1109/GCIS.2013.31. URL: http://ieeexplore.ieee.org/document/6805928/ (cf. p. 2, 81, 83, 95, 96, 125).

Saint-Marc, Cécile (2017). "Formalization and geovisualization of historical natural risk events to understand spatial dynamics". Theses. Université Grenoble-Alpes. URL: https://hal.archives-ouvertes.fr/tel-01572326 (cf. p. 92).

San Martín, Mauro and Claudio Gutierrez (2009). "Representing, querying and transforming social networks with RDF/SPARQL". In: *European semantic web conference*. Springer, pp. 293–307 (cf. p. 1).

Sanchez, Alejandro and Leo Ordinez (2020). "Towards an ontology for designing cycling routes in the city of Puerto Madryn". In: *2020 Seventh International Conference on eDemocracy & eGovernment (ICEDEG)*. 2020 Seventh International Conference on eDemocracy & eGovernment (ICEDEG). Buenos Aires, Argentina: IEEE, pp. 143–150. ISBN: 978-1-72815-882-2. DOI: 10.1109/ICEDEG48599.2020.9096701. URL: https://ieeexplore.ieee.org/document/9096701/ (cf. p. 1).

Scheider, S., A. Ballatore, and R. Lemmens (2019). "Finding and sharing GIS methods based on the questions they answer". In: *International Journal of Digital Earth* 12.5, pp. 594–613. ISSN: 1753-8947, 1753-8955. DOI: 10.1080/17538947.2018.1470688. URL: https://www.tandfonline.com/doi/full/10.1080/17538947.2018.1470688 (cf. p. 78).

Schneider, Michael (2012). *OWL 2 Web Ontology Language: RDF-Based Semantics (Second Edition)*. W3C Recommendation. World Wide Web Consortium. URL: https://www.w3.org/TR/2012/REC-owl2-rdf-based-semantics-20121211/ (cf. p. 53).

Schreiber, Guus and Yves Raimond (2014). *RDF 1.1 Primer*. W3C Recommendation. World Wide Web Consortium. URL: https://www.w3.org/TR/rdf11-primer/ (cf. p. 45).

Shneiderman, Ben (1996). "The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations". In: *Proceedings of the 1996 IEEE Symposium on Visual Languages*. VL '96. USA: IEEE Computer Society, p. 336. ISBN: 0-8186-7508-X (cf. p. 21).

Slocum, Terry A., Daniel C. Cliburn, Johannes J. Feddema, and James R. Miller (2003). "Evaluating the Usability of a Tool for Visualizing the Uncertainty of the Future Global Water Balance". In: *Cartography and Geographic Information Science* 30.4, pp. 299–317. ISSN: 1523-0406, 1545-0465. DOI: 10.1559/152304003322606210. URL: http://www.tandfonline.com/doi/abs/10.1559/152304003322606210 (cf. p. 38).

Slocum, Terry A., Robert B. MacMaster, Fritz C. Kessler, and Hugh H. Howard (2009). *Thematic cartography and geovisualization, 3rd Edition.* Pearson Education. I–X, 1–561. ISBN: 978-0-13-229834-6 (cf. p. 15, 17).

Smith, Duncan A. (2016). "Online interactive thematic mapping: Applications and techniques for socio-economic research". In: *Computers, Environment and Urban Systems* 57, pp. 106–117. ISSN: 01989715. DOI: 10.1016/j.compenvurbsys.2016.01.002. URL: https://linkinghub.elsevier.com/retrieve/pii/S0198971516300023 (cf. p. 37).

Smith, Richard (2010). "Designing a Cartographic Ontology for Use with Expert Systems". In: *Proceedings of the ASPRS/CaGIS 2010 Specialty Conference.* Orlando, Florida. URL: http://www.asprs.org/a/publications/proceedings/orlando2010/files/Smith.pdf (cf. p. 2, 79, 80, 83, 95–97, 105, 125, 208, 209).

Solbrig, Harold and Eric Prud'hommeaux (2014). *Shape Expressions 1.0 Definition.* W3C Member Submission. World Wide Web Consortium. URL: https://www.w3.org/Submission/2014/SUBM-shex-defn-20140602/ (cf. p. 63).

Soman, Ranjith K (2019). "Modelling construction scheduling constraints using shapes constraint language (SHACL)". In: 2019 European Conference on Computing in Construction, pp. 351–358. DOI: 10.35490/EC3.2019.170. URL: https://ec-3.org/publications/conferences/2019/paper/?id=170 (cf. p. 63).

Sporny, Manu, Dave Longley, Gregg Kellogg, Markus Lanthaler, Pierre-Antoine Champin, and Niklas Lindström (2020). *JSON-LD 1.1 - A JSON-based Serialization for Linked Data.* W3C Recommendation. World Wide Web Consortium. URL: https://www.w3.org/TR/json-ld11/ (cf. p. 49).

Sporny, Manu, Dave Longley, Gregg Kellogg, Markus Lanthaler, and Niklas Lindström (2014). *JSON-LD 1.0 - A JSON-based Serialization for Linked Data.* W3C Recommendation. World Wide Web Consortium. URL: https://www.w3.org/TR/2014/REC-json-ld-20140116/ (cf. p. 49).

Sreeves, Matthew (2018). *Managing Uncertainty in Mountain Search and Rescue.* MSc Internship report. Université Grenoble Alpes, p. 33 (cf. p. 33, 34).

Steinberg, Jean (2000). "L'apport de la sémiologie graphique de Jacques Bertin à la cartographie pour l'aménagement et l'urbanisme". In: *Cybergeo.* ISSN: 1278-3366. DOI: 10.4000/cybergeo.497. URL: http://journals.openedition.org/cybergeo/497 (cf. p. 17).

Subercaze, Julien, Christophe Gravier, Jules Chevalier, and Frédérique Laforest (2016). "Inferray: fast in-memory RDF inference". In: 9.6, pp. 468–479 (cf. p. 55).

Tai, Wei, John Keeney, and Declan O'Sullivan (2011). "RESP: A Computer Aided OWL REasoner Selection Process". In: *2011 IEEE Fifth International Conference on Semantic Computing.* 2011 IEEE Fifth International Conference on Semantic Computing (ICSC). Palo Alto, CA, USA: IEEE, pp. 27–34. ISBN: 978-1-4577-1648-5. DOI: 10.1109/ICSC.2011.17. URL: http://ieeexplore.ieee.org/document/6061432/ (cf. p. 73).

Trillos Ujueta, Jaime Manuel, Ana Cristina Trillos Ujueta, and Luis Daniel Fernandes Rotger (2018). "Transforming RDF data into maps: Rdf2map library". In: pp. 1–10. URL: http://cscubs.cs.uni-bonn.de/2018/proceedings/paper_2.pdf (cf. p. 88).

Tyner, Judith A. (2010). *Principles of map design.* OCLC: ocn437300476. New York: Guilford Press. 259 pp. ISBN: 978-1-60623-544-7 (cf. p. 15, 17).

Vandecasteele, Arnaud and Aldo Napoli (2012). "An enhanced spatial reasoning ontology for maritime anomaly detection". In: *2012 7th International Conference on System of Systems Engineering (SoSE)*. 2012 7th International Conference on System of Systems Engineering (SoSE). Genova: IEEE, pp. 1–6. ISBN: 978-1-4673-2975-0 978-1-4673-2974-3 978-1-4673-2973-6. DOI: 10.1109/SYSoSE.2012.6384120. URL: http://ieeexplore.ieee.org/document/6384120/ (cf. p. 1).

Varanka, Dalia E. and E. Lynn Usery (2018). "The map as knowledge base". In: *International Journal of Cartography* 4.2, pp. 201–223. ISSN: 2372-9333, 2372-9341. DOI: 10.1080/23729333.2017.1421004. URL: https://www.tandfonline.com/doi/full/10.1080/23729333.2017.1421004 (cf. p. 2, 88, 89, 95, 97, 114, 210).

Vatin, Gabriel (2014). "Formalization of a Geovisual Analytics Support Environment". Theses. Ecole Nationale Supérieure des Mines de Paris ; CRC - Centre de recherche sur les Risques et les Crises. URL: https://tel.archives-ouvertes.fr/tel-01253688 (cf. p. 83).

Villanova-Oliver, Marlène (2018). "Représentations de connaissances spatiales évolutives : des ontologies aux géovisualisations". Habilitation à diriger des recherches. Communauté Université Grenoble Alpes. URL: https://hal.archives-ouvertes.fr/tel-01935490 (cf. p. 92, 93, 95–97, 105, 106, 131, 208, 210).

Viry, Matthieu and Marlène Villanova-Oliver (2019). "L'Ontologie d'Alerte Choucas : de la modélisation des connaissances à un outil support d'un raisonnement géovisuel - Application à la recherche de victime en haute-montagne". In: SAGEO'19. Clermont-Ferrand, France (cf. p. 33).

— (2020). "Ontologie d'Alerte Choucas : de la modélisation des connaissances à un outil support d'un raisonnement géovisuel". In: *Geomatica* 73.3, pp. 87–103. ISSN: 1195-1036, 1925-4296. DOI: 10.1139/geomat-2020-0005. URL: http://www.nrcresearchpress.com/doi/10.1139/geomat-2020-0005 (cf. p. 33, 164, 166).

Viry, Matthieu, Marlène Villanova-Oliver, Jacques Gautier, Matthew Sreeves, and Paule-Annick Davoine (2019). "Improving the search for victims in mountain environments with geovisualization and competing hypotheses management". In: *Proceedings of the ICA* 2, pp. 1–8. ISSN: 2570-2092. DOI: 10.5194/ica-proc-2-138-2019. URL: https://www.proc-int-cartogr-assoc.net/2/138/2019/ (cf. p. 33, 166).

W3C SPARQL WG, The W3C SPARQL Working Group (2013). *SPARQL 1.1 Overview*. W3C Recommendation. World Wide Web Consortium. URL: https://www.w3.org/TR/2013/REC-sparql11-overview-20130321/ (cf. p. 59).

Wagner, Matthew, Dalia E. Varanka, and E. Lynn Usery (2020). *A system design for implementing advanced feature descriptions for a map knowledge base*. Report 2019-5148. Reston, VA, p. 38. DOI: 10.3133/sir20195148. URL: http://pubs.er.usgs.gov/publication/sir20195148 (cf. p. 89).

Wehrend, S. and C. Lewis (1990). "A problem-oriented classification of visualization techniques". In: *Proceedings of the First IEEE Conference on Visualization: Visualization '90*. First IEEE Conference on Visualization: Visualization '90. San Francisco, CA, USA: IEEE Comput. Soc. Press, pp. 139–143, ISBN: 978-0-8186-2083-6. DOI: 10.1109/VISUAL.1990.146375. URL: http://ieeexplore.ieee.org/document/146375/ (cf. p. 108).

Weinlich, Michael, Peter Kurz, Melissa B. Blau, Felix Walcher, and Stefan Piatek (2018). "Significant acceleration of emergency response using smartphone geolocation data and a worldwide emergency call support system". In: *PLOS ONE* 13.5. Ed. by Yu Ru Kou, e0196336. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0196336. URL: http://dx.plos.org/10.1371/journal.pone.0196336 (cf. p. 4).

Wiegand, Nancy and Cassandra García (2007). "A Task-Based Ontology Approach to Automate Geospatial Data Retrieval". In: *Transactions in GIS* 11.3, pp. 355–376. ISSN: 1361-1682, 1467-9671. DOI: 10.1111/j.1467-9671.2007.01050.x. URL: https://onlinelibrary.wiley.com/doi/10.1111/j.1467-9671.2007.01050.x (cf. p. 1).

Wilkening, Jan and Sara Irina Fabrikant (2011). "How Do Decision Time and Realism Affect Map-Based Decision Making?" In: *Spatial Information Theory*. Ed. by Max Egenhofer, Nicholas Giudice, Reinhard Moratz, and Michael Worboys. Red. by David Hutchison et al. Vol. 6899. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 1–19. ISBN: 978-3-642-23195-7 978-3-642-23196-4. DOI: 10.1007/978-3-642-23196-4_1. URL: http://link.springer.com/10.1007/978-3-642-23196-4_1 (cf. p. 31).

Zaki, Chamseddine, Elyes Zekri, Myriam Servières, Guillaume Moreau, and Gérard Hégron (2011). "Generic modeling of application and spatiotemporal data: Application to the study of pedestrian behavior". In: *Intelligent Decision Technologies* 5.4, pp. 297–307. ISSN: 18758843, 18724981. DOI: 10.3233/IDT-2011-0113. URL: https://www.medra.org/servlet/aliasResolver?alias=iospress&doi=10.3233/IDT-2011-0113 (cf. p. 3).

Zhang, Chuanrong, Tian Zhao, and Weidong Li (2015). *Geospatial Semantic Web*. OCLC: 1127150759. ISBN: 978-3-319-17801-1. URL: http://www.vlebooks.com/vleweb/product/openreader?id=none&isbn=9783319178011 (cf. p. 1, 78).

Zhang, Chuanrong, Tian Zhao, Weidong Li, and Jeffrey P. Osleeb (2010). "Towards logic-based geospatial feature discovery and integration using web feature service and geospatial semantic web". In: *International Journal of Geographical Information Science* 24.6, pp. 903–923. ISSN: 1365-8816, 1362-3087. DOI: 10.1080/13658810903240687. URL: http://www.tandfonline.com/doi/abs/10.1080/13658810903240687 (cf. p. 1, 78).

Çöltekin, Arzu, Susanne Bleisch, Gennady Andrienko, and Jason Dykes (2017). "Persistent challenges in geovisualization – a community perspective". In: *International Journal of Cartography* 3 (sup1), pp. 115–139. ISSN: 2372-9333, 2372-9341. DOI: 10.1080/23729333.2017.1302910. URL: https://www.tandfonline.com/doi/full/10.1080/23729333.2017.1302910 (cf. p. 3, 7).

Çöltekin, Arzu, Halldór Janetzko, and Sara Fabrikant (2018). "Geovisualization". In: *Geographic Information Science & Technology Body of Knowledge* 2018 (Q2). ISSN: 25772848. DOI: 10.22224/gistbok/2018.2.6. URL: https://gistbok.ucgis.org/bok-topics/geovisualization (cf. p. 13).

Appendix A

Extract from the model describing NUTS2 entities

```
@prefix ex: <http://example.com/example-nuts#> .
@prefix geo: <http://www.opengis.net/ont/geosparql#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://example.com/example-nuts#> a owl:Ontology .

ex:grossDomesticProduct a owl:DatatypeProperty ;
    rdfs:label "Gross Domestic Product"@en ;
    rdfs:domain ex:NutsUnit ;
    rdfs:range xsd:integer .

ex:id a owl:DatatypeProperty ;
    rdfs:label "Id"@en ;
    rdfs:domain ex:NutsUnit ;
    rdfs:range xsd:string .

ex:name a owl:DatatypeProperty ;
    rdfs:label "Name"@en ;
    rdfs:domain ex:NutsUnit ;
    rdfs:range xsd:string .

ex:population a owl:DatatypeProperty ;
    rdfs:label "Population"@en ;
    rdfs:domain ex:NutsUnit ;
    rdfs:range xsd:integer .

ex:unemploymentRate a owl:DatatypeProperty ;
    rdfs:label "Unemployment Rate"@en ;
    rdfs:domain ex:NutsUnit ;
    rdfs:range xsd:decimal .

ex:hasCountry a owl:ObjectProperty ;
    rdfs:label "Has Country"@en ;
```

```
    rdfs:domain ex:NutsUnit ;
    rdfs:range ex:Country .

ex:Country a owl:Class ;
    rdfs:label "Country"@en .

ex:NutsUnit a owl:Class ;
    rdfs:label "Nuts Unit"@en ;
    rdfs:subClassOf geo:Feature .

# Example of individuals
ex:NUTS2_2013_TR51 a ex:NutsUnit ;
    ex:hasCountry ex:Turkey ;
    ex:id "TR51"^^xsd:string ;
    ex:name "Ankara"^^xsd:string ;
    ex:population 5045083 ;
    ex:unemploymentRate 11.5 ;
    geo:hasGeometry [ a geo:Geometry ;
        geo:asWKT "POLYGON ((...))"^^geo:wktLiteral ] .

ex:Turkey a ex:Country ;
    rdfs:label "Turkey"@en .
```

Listing A.1: Extract from the model describing NUTS2 entities.

Appendix B

Example of a Derivation Model validation report

The following validation report (Listing B.1) contains a `sh:Warning` (lines 19-27) about the missing title (because no title is defined, neither on the `gviz:GeoVisualApplication` nor on the only `gviz:Map2dComponent`) and a `sh:Violation` (lines 28-35) about a missing property (`graphic:hasStroke`) on a `symblzr:LineSymbolizer`, which requires it. What is raised at the *violation* level prevents the framework from working properly or is not consistent with its expectations (for instance here with the invalid `symblzr:LineSymbolizer`) as opposed to what is raised at the *warning* level.

```
1  @prefix :         <http://example.com/cvk-ir#> .
2  @prefix cvkd:     <http://lig-tdcge.imag.fr/steamer/covikoa/derivation#> .
3  @prefix geof:     <http://www.opengis.net/def/function/geosparql/> .
4  @prefix cvkc:     <http://lig-tdcge.imag.fr/steamer/covikoa/context#> .
5  @prefix owl:      <http://www.w3.org/2002/07/owl#> .
6  @prefix xsd:      <http://www.w3.org/2001/XMLSchema#> .
7  @prefix rdfs:     <http://www.w3.org/2000/01/rdf-schema#> .
8  @prefix geo:      <http://www.opengis.net/ont/geosparql#> .
9  @prefix gviz:     <http://lig-tdcge.imag.fr/steamer/covikoa/geoviz#> .
10 @prefix rdf:      <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
11 @prefix sh:       <http://www.w3.org/ns/shacl#> .
12 @prefix xml:      <http://www.w3.org/XML/1998/namespace> .
13 @prefix loac:     <http://purl.org/loac#> .
14 @prefix symblzr:  <https://gis.lu.se/ont/data_portrayal/symbolizer#> .
15 @prefix graphic:  <https://gis.lu.se/ont/data_portrayal/graphic#> .
16
17 [] a sh:ValidationReport ;
18     sh:conforms "false"^^xsd:boolean ;
19     sh:result [ a sh:ValidationResult ;
20         sh:focusNode :HypothesisMap ;
21         sh:resultMessage "Each gviz:Map2dComponent (or the parent gviz:
       GeoVisualApplication) should define a title with 'dct:title'."@en ;
22         sh:resultPath <http://purl.org/dc/terms/title> ;
23         sh:resultSeverity sh:Warning ;
24         sh:sourceConstraint [] ;
```

```
25          sh:sourceConstraintComponent sh:SPARQLConstraintComponent ;
26          sh:sourceShape <http://example.com/covikoa-derivation-model-shape#
        MapOrApplicationHasATitle>
27        ] ;
28      sh:result [ a sh:ValidationResult ;
29          sh:focusNode :SymbolizerReferenceFeatureHighlightLine ;
30          sh:resultMessage "Property needs to have at least 1 values, but found 0" ;
31          sh:resultPath graphic:hasStroke ;
32          sh:resultSeverity sh:Violation ;
33          sh:sourceConstraintComponent sh:MinCountConstraintComponent ;
34          sh:sourceShape <http://example.com/covikoa-derivation-model-shape#
        StrokePropertyShape>
35        ] .
```

Listing B.1: Example of a Derivation Model validation report produced by CoViKoa.

Appendix C

Example of the rewriting of the Derivation Model when using the cartographic vocabulary

```
############################################################
# The user defines these components:
############################################################
:appChoroNutsUnemp a gviz:GeoVisualApplication ;  # A geovisual application ..
    gviz:hasGeoVisualComponent :nutsMap , :nutsMapLegend ; # with a map and a legend
     .

:nutsMap a gviz:Map2dComponent ;
    cvkd:hasDefaultExtent ex:NutsUnit ;
    cvkc:widthPx 1200 ;
    cvkc:heightPx 900 ;
    cvkc:hasBaseMap cvkc:stamenWatercolorBaseMap ;
    cvkc:hasMappingLibrary cvkc:openlayers .

:nutsMapLegend a gviz:LegendComponent ;
    gviz:linkedTo :nutsMap .

############################################################
# The user instantiates an existing cartographic solution:
############################################################
:ChoroplethUnempNuts a carto:QuantitativeChoropleth ;
    dct:title "Unemployment Rate (NUTS2 units, 2013, %)" ;
    carto:targetsSpatialFeature ex:NutsUnit ;
    carto:targetsProperty ex:unemploymentRate ;
    carto:ofData [ a carto:Area , carto:RelativeQuantitativeData , carto:Number ] ;
    carto:hasDataBreaks (0.0 4.0 10.4 16.6 24.2 36) ;
    carto:rendersNoData "true"^^xsd:boolean ;
    carto:hasPaletteScheme dcpal:Blues .
```

Listing C.1: Derivation Model provided by the user for a choropleth portrayal defined with the *carto* vocabulary.

```
############################################################
# The user obtains the declaration of the IR class
# that will be used by CoViKoa:
############################################################
<urn:id:covikoa-ir:NutsUnit-GVR> a owl:Class ;
    rdfs:subClassOf gviz:GeoVisualIntermediateRepresentation ;
    cvkd:represents ex:NutsUnit .

############################################################
# The user obtains the Portrayal and the Portrayal Rules
# that will allow to link data from the SDM to the
# appropriate symbolizers:
############################################################
<urn:id:covikoa-ir:ChoroplethUnempNuts-Portrayal>
    a gviz:Portrayal , carto:QuantitativeChoropleth  ;
    dct:title "Unemployment Rate (NUTS2 units, 2013, %)" ;
    cvkd:denotesGVR <urn:id:covikoa-ir:NutsUnit-GVR> ;
    carto:targetsSpatialFeature ex:NutsUnit ;
    carto:targetsProperty ex:unemploymentRate ;
    carto:ofData [ a carto:Area , carto:RelativeQuantitativeData , carto:Number ] ;
    carto:hasDataBreaks (0.0 4.0 10.4 16.6 24.2 36) ;
    carto:rendersNoData "true"^^xsd:boolean ;
    carto:hasPaletteScheme dcpal:Blues ;
    gviz:hasPortrayalRule [
        gviz:hasSymbol [ a symb:Symbol ;
            dct:title "]0.0,4.0]" ;
            symblzr:hasSymbolizer <urn:id:symbolizer:Area_Blues_5_1> ;
          ] ;
        cvkd:hasPropertyConstraint [
            cvkd:propertyPath ex:unemploymentRate ;
            cvkd:valueIsLessThanOrEqualTo 4.0 ;
          ] ;
      ] ;
    gviz:hasPortrayalRule [
        gviz:hasSymbol [ a symb:Symbol ;
            dct:title "]4.0,10.4]" ;
            symblzr:hasSymbolizer <urn:id:symbolizer:Area_Blues_5_2> ;
          ] ;
        cvkd:hasPropertyConstraint [
            cvkd:propertyPath ex:unemploymentRate ;
            cvkd:valueIsLessThanOrEqualTo 10.4 ;
            cvkd:valueIsGreaterThan 4.0 ;
          ] ;
      ] ;
    gviz:hasPortrayalRule [
        gviz:hasSymbol [ a symb:Symbol ;
            dct:title "]10.4,16.6]" ;
            symblzr:hasSymbolizer <urn:id:symbolizer:Area_Blues_5_3> ;
          ] ;
        cvkd:hasPropertyConstraint [
            cvkd:propertyPath ex:unemploymentRate ;
            cvkd:valueIsLessThanOrEqualTo 16.6 ;
            cvkd:valueIsGreaterThan 10.4 ;
          ] ;
      ] ;
    gviz:hasPortrayalRule [
        gviz:hasSymbol [ a symb:Symbol ;
            dct:title "]16.6,24.2]" ;
```

```
            symblzr:hasSymbolizer <urn:id:symbolizer:Area_Blues_5_4> ;
          ] ;
        cvkd:hasPropertyConstraint [
            cvkd:propertyPath ex:unemploymentRate ;
            cvkd:valueIsLessThanOrEqualTo 24.2 ;
            cvkd:valueIsGreaterThan 16.6 ;
          ] ;
      ] ;
    gviz:hasPortrayalRule [
        gviz:hasSymbol [ a symb:Symbol ;
            dct:title "]24.2,34.8]" ;
            symblzr:hasSymbolizer <urn:id:symbolizer:Area_Blues_5_5> ;
          ] ;
        cvkd:hasPropertyConstraint [
            cvkd:propertyPath ex:unemploymentRate ;
            cvkd:valueIsLessThanOrEqualTo 36 ;
            cvkd:valueIsGreaterThan 24.2 ;
          ] ;
      ] ;
    gviz:hasPortrayalRule [
        gviz:hasSymbol [ a symb:Symbol ;
            dct:title "No data" ;
            symblzr:hasSymbolizer <urn:id:symbolizer:Area_Blues_NoData>  ;
          ] ;
        cvkd:hasPropertyConstraint [
            cvkd:propertyPath ex:unemploymentRate ;
            cvkd:valueOrObjectIsEqualTo cvkd:absentProperty ;
          ] ;
      ] .

############################################################
# And all the necessary symbolizers:
############################################################
<urn:id:symbolizer:Area_Blues_5_1> a symblzr:PolygonSymbolizer ;
    graphic:hasFill [ a graphic:Fill ;
        graphic:fillColor "rgb(239,243,255)"^^graphic:cssColorLiteral
      ] ;
    graphic:hasStroke [ a graphic:Stroke ;
        graphic:strokeColor "rgba(99,99,99,0.7)"^^graphic:cssColorLiteral
      ] .

<urn:id:symbolizer:Area_Blues_5_2> a symblzr:PolygonSymbolizer ;
    graphic:hasFill [ a graphic:Fill ;
        graphic:fillColor "rgb(189,215,231)"^^graphic:cssColorLiteral
      ] ;
    graphic:hasStroke [ a graphic:Stroke ;
        graphic:strokeColor "rgba(99,99,99,0.7)"^^graphic:cssColorLiteral
      ] .

<urn:id:symbolizer:Area_Blues_5_3> a symblzr:PolygonSymbolizer ;
    graphic:hasFill [ a graphic:Fill ;
        graphic:fillColor "rgb(107,174,214)"^^graphic:cssColorLiteral
      ] ;
    graphic:hasStroke [ a graphic:Stroke ;
        graphic:strokeColor "rgba(99,99,99,0.7)"^^graphic:cssColorLiteral
      ] .

<urn:id:symbolizer:Area_Blues_5_4> a symblzr:PolygonSymbolizer ;
```

```
    graphic:hasFill [ a graphic:Fill ;
        graphic:fillColor "rgb(49,130,189)"^^graphic:cssColorLiteral
      ] ;
    graphic:hasStroke [ a graphic:Stroke ;
        graphic:strokeColor "rgba(99,99,99,0.7)"^^graphic:cssColorLiteral
      ] .

<urn:id:symbolizer:Area_Blues_5_5> a symblzr:PolygonSymbolizer ;
    graphic:hasFill [ a graphic:Fill ;
        graphic:fillColor "rgb(8,81,156)"^^graphic:cssColorLiteral
      ] ;
    graphic:hasStroke [ a graphic:Stroke ;
        graphic:strokeColor "rgba(99,99,99,0.7)"^^graphic:cssColorLiteral
      ] .

<urn:id:symbolizer:Area_Blues_NoData> a symblzr:PolygonSymbolizer ;
    graphic:hasFill [ a graphic:Fill ;
        graphic:fillColor "rgba(127,127,127,1)"^^graphic:cssColorLiteral
      ] ;
    graphic:hasStroke [ a graphic:Stroke ;
        graphic:strokeColor "rgba(99,99,99,0.7)"^^graphic:cssColorLiteral
      ] .
```

Listing C.2: Declarations generated and injected into the Derivation Model.