Model-driven Integration of Heterogeneous n-Dimensional Urban Data

Diego Vinasco-Alvarez

N° d'ordre NNT : 2023LYO20092

THÈSE de DOCTORAT DE L'UNIVERSITÉ LUMIÈRE LYON 2

École Doctorale : ED 512

Informatique et Mathématiques

Discipline : Informatique

Soutenue publiquement le 11 décembre 2023 par :

Diego VINASCO-ALVAREZ

Model-driven Integration of Heterogeneous n-Dimensional Urban Data.

Devant le jury composé de :

Christophe NICOLLE, Professeur, Université de Bourgogne, Président

Marlène VILLANOVA-OLIVER, Maîtresse de conférences HDR, Université Grenoble Alpes, Rapporteure

François PINET, Directeur de recherche HDR, INRAE, Rapporteur

Gilles GESQUIÈRE, Professeur, Université Lumière Lyon 2, Directeur de thèse

Sylvie SERVIGNE, Maîtresse de conférences, INSA de Lyon, Co-Directrice de thèse

John SAMUEL, Enseignant chercheur, CPE Lyon, Co-Directeur de thèse

Contrat de diffusion

Model-driven Integration of Heterogeneous n-Dimensional Urban Data

Diego Vinasco-Alvarez
Université de Lyon, Université Lumière Lyon 2, LIRIS UMR-CNRS 5205
2023

A thesis presented for the degree of
Doctor of Computer Science
Doctoral school: ED 512 “Informatique et Mathématiques”

Thesis examiners:
Marlène Villanova-Oliver, Maître de conférences, Université Grenoble Alpes, Rapportrice
François Pinet, Directeur de recherche, INRAE, Rapporteur
Sidonie Christophe, Directrice de recherche, IGN, Examinatrice
Christophe Nicolle, Professeur exceptionnel, Université de Bourgogne I3M, Examinateur
Gilles Gesquière, Professeur exceptionnel, Université Lumière Lyon 2, Directeur de thèse
Sylvie Servigne, Maître de conférences, INSA Lyon, Co-encadrante de thèse
John Samuel, Enseignant-Chercheur, CPE Lyon, Co-encadrant de thèse

Acknowledgements

It's well known that getting a Ph.D. is a lot of work with many challenges and this thesis is no exception. There are many people and organizations that have helped me along the way and without them, this work would not have been possible.

I would first like to thank the University Lumiére Lyon 2 for funding this research project. I am enormously thankful for my thesis supervisors Gilles Gesquère, Sylvie Servigne, and John Samuel for their invaluable guidance, support, and for sharing their wisom and experience with me. These have helped me grow—not only as a researcher and a communicator—but have opened the doors to many opportunities for me, for which I am immensly grateful.

In addition, I would like to thank the members of the LIRIS laboratory and the VCity project, whom have had great pleasure in working with these past years. In particular, I would like to thank Eric Boix, Clément Colin, Corentin Gautier, Valentin Machado, Mathieu Livebardon, Lorenzo Marnat, and Jey Puget Gil for our many fruitful discussions—both technical and theoretical—that have helped throughout my thesis.

I'd also like to thank Victor de Boer his feedback during the 19th ESWC Ph.D. Symposium. Likewise, I'd like to thank all the anonymous reviewers of conference proceedings and journal articles who have provided their insightful feedback. I would also like to express my sincere thanks to the members of my doctoral thesis commitee and the reviewers of this dissertation.

I would like to thank the Open Geospatial Consortium and the members of the CityGML SWG group for introducing me to data standardization and letting me be a part of the many insightful discussions had during the development of the CityGML 3.0 GML encoding. In particular, I would like to thank Tatjana Kutzner, Thomas Kolbe, Claus Nagel, Steve Smith, Volker Coors, Nobuyuki Hashimoto, Peter Parslow, and Dean Hintz.

Also, my Ph.D. work belongs to the particular generation of Ph.D. works that were effectuated during the height of the COVID pandemic. I am tremendously grateful to the Deleuil-Lecoeur family for their immeasurable generosity and hospitality.

I would also like to thank all of my past professors and teachers for their guidance. In particular, I would like to thank Linnea Archer and Philip Olson for encouraging me to be better student and writer.

Finally, I would like to thank my family and friends for their love, care, and support, and who have—in no small part—made me the person that I am today. I would especially like to thank Louise Deleuil for her earnest encouragement and for putting up with me over the years.

Abstract

Urbanization and anthropization are dynamic and multifaceted change processes with strong impacts on our societies. To confront the increasing need for understanding these processes, data-driven approaches such as urban digital twins and smart city applications have become powerful solutions to model, visualize, and navigate the complex urban landscape and lifecycle. Often, these approaches rely on integrating these urban data, which consist of a variety of different information from different actors and organizations. In this context, urban data integration is a process that combines heterogeneous urban data from these information domains to create more complete data views of the urban landscape and its evolution for the user. To provide these views, the integration process must take into consideration both the heterogeneous nature of this data and its n-dimensional (nD) characteristics (i.e., 2D, 3D, time, semantics).

The aim of this thesis is to propose a robust semantic and approach to data integration based on standards and models, where the conceptual models underlying the different data sources are preserved. Instead of using direct data conversion, we propose a model-driven methodology to ensure that, for a given application, all the information that can be represented by a given standard will be available to the application, even if the standards and the data models they define implicitly evolve. In this thesis, we focus on nD urban data—and in particular, on 3D and temporal urban data. Building a robust semantic model is challenging, as choosing an expressive semantic model requires a good understanding of the different types of information that can be represented by the various sources and the standards to which they belong. Also, automated mapping of conceptual model information has a number of limitations. By considering heterogeneous constantly evolving nD urban data standards, our aim is to be able to ensure data interoperability and the possibility of integrating this data with other, related open data. This proposal also aims to improve access to geospatial data on the web, while guaranteeing its integrity and access.

Résumé

L'urbanisation et l'anthropisation sont des processus de changement dynamiques et multiformes qui ont de fortes répercussions sur nos sociétés. Pour répondre au besoin de comprendre ces processus, les approches fondées sur les données—telles que les jumeaux numériques urbains et les applications de villes intelligentes—sont devenues des solutions puissantes pour modéliser, visualiser et naviguer dans le paysage et le cycle de vie urbains complexes. Souvent, ces approches reposent sur l'intégration de ces données urbaines, qui consistent en une variété d'informations provenant de différents acteurs et organisations. Dans ce contexte, l'intégration des données urbaines est un processus qui combine des données urbaines hétérogènes provenant de ces domaines d'information afin de créer des vues de données plus complètes du paysage urbain et de son évolution pour l'utilisateur. Pour fournir ces vues, le processus d'intégration doit prendre en compte à la fois la nature hétérogène de ces données et leurs caractéristiques n-dimensionnelles (nD) (i.e., 2D, 3D, temps, sémantique).

L'objectif de cette thèse est de proposer une approche sémantique évolutive de l'intégration de données basée sur les standards et des modèles, où les modèles conceptuels sous-jacents aux différentes sources de données sont préservés. Au lieu d'utiliser une conversion directe des données, nous proposons une méthodologie fondée sur un modèle sémantique garantissant que, pour une application, toutes les informations pouvant être représentées par une standard donnée seront disponibles pour l'application. Une approche basée modèle permet de garantir que l'application peut utiliser toutes les informations exposées par l'une de ces sources de données, même si les standards et implicitement les modèles évoluent. Nous nous intéressons dans cette thèse aux données urbaines à n-dimensions et en particulier aux données urbaines 3D et temporelles. La construction d'un modèle sémantique évolutif est un défi, car le choix d'un modèle sémantique expressif nécessite une bonne compréhension des différentes informations qui peuvent être représentées par les différentes sources et les standards dont elles relèvent. En outre, la mise en correspondance automatisée des informations du modèle urbain avec le modèle sémantique proposé présente plusieurs limites. Également, l'évolution automatisée du modèle sémantique, avec l'intégration de nouvelles sources de données ainsi que de nouvelles versions des standards de données associées en raison d'évolutions continues nécessite une solution générique. En considérant plusieurs standards de données, en constante évolution, pour représenter des données urbaines en n-dimension souvent hétérogènes, notre objectif est de pouvoir assurer l'interopérabilité des données et la possibilité de les intégrer à d'autres données ouvertes liées. Cette proposition

vise également à améliorer l'accès aux données géospatiales sur le web tout en garantissant leur intégrité et leur accès.

Contents

List of Figures

List of Tables

The following publications in journals, conference proceedings, and technical reports were produced during this thesis.

[Sam+23] John Samuel, Vincent Jaillot, Clément Colin, et al. “UD-SV: Urban data services and visualization framework for sharing multidisciplinary research”. en. In: Transactions in GIS 27.3 (Apr. 2023), pp. 841-858

[Col+23] Clément Colin, Corentin Gautier, Diego Vinasco-Alvarez, et al. “UD-SV : Plateforme d’exploration de données urbaines à n-dimensions — Espace, Temps, Thématiques”. fr. In: Mappemonde. Revue trimestrielle sur l’image géographique et les formes du territoire 135 (Mar. 2023). Number: 135 Publisher: UMR Espace

[Vin22] Diego Vinasco-Alvarez. “Leveraging Standards in Model-Centric Geospatial Knowledge Graph Creation”. en. In: The Semantic Web: ESWC 2022 Satellite Events. Ed. by Paul Groth, Anisa Rula, Jodi Schneider, et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2022, pp. 224-233

[Vin+21a] Diego Vinasco-Alvarez, John Samuel, Sylvie Servigne, and Gilles Gesquière. “Towards Limiting Semantic Data Loss In 4D Urban Data Semantic Graph Generation”. In: ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences VIII-4/W2-2021 (Oct. 2021), pp. 37-44

[Vin+21b] Diego Vinasco-Alvarez, John Samuel, Sylvie Servigne, and Gilles Gesquière. “Towards a semantic web representation from a 3D geospatial urban data model”. In: May 2021

[Vin+20] Diego Vinasco-Alvarez, John Samuel, Sylvie Servigne, and Gilles Gesquière. “From CityGML to OWL”. Technical Report. LIRIS UMR 5205, Sept. 2020

Part I

Introduction

1 Context

Urbanization and anthropization are dynamic and multifaceted change processes with strong impacts on our societies. According to 2018 UN estimates [Uni19], 2007 was the first year in human history when more of the world's population resided in urban regions as opposed to rural ones. The same estimates expect this trend to continue to rise to 68% by 2050. With the increasing availability of open urban data [Bar+14], data-driven approaches such as smart city applications and urban digital twins [Bat18; Jul+18; SH20] have become powerful solutions to help understand urbanization and improve sustainability and quality of life in the urban environment [SKH18].

In this context, the "smart" city is one capable of monitoring various aspects of the city (such as pollution, transportation, waste management, poverty, energy, housing availability, etc. . . .), digitally capturing this information, and using it to digitally advise governments and citizens in decision-making and facilitate automation in these urban areas [RP16; SKH18; DZS21]. Today cities such as London (UK), New York (USA), and Paris (France) are implementing technologies such as Internet of Things (IoT) devices for capturing this information through sensors and transporting it through energy-efficient communication networks [SKH18]. Additionally, cloud-based data centers can be used for data management processes (such as data storage, processing, fusion, and analysis) before being exploited in smart city applications and systems [SKH18]. Recently, the use of urban digital twins have been proposed as approaches to smart city applications [DZS21], for example in Zurich, Switzerland (figure 1.1).

The term *digital twin* originates from product lifecycle management [Gri14] where a digital model (the twin) of a physical product or system is created to be able to monitor, test, and simulate different phenomena over the course of the product's lifecycle [Jon+20]. The urban analog of this concept is the Urban Digital Twin (UDT). In this context, the twin of a physical city or urban space is often composed of a geospatial 2D or 3D model and standardized semantic (or thematic) urban data [KD21; DZS21]. As sensors, surveying teams, administrative entities record changes in the urban landscape, these virtual models are updated (or "retwinned").

Over the evolution of both twins (physical and virtual), virtual processes[1] can be applied to the digital twin to measure or simulate the effects of different urban phenomena [Bat18].

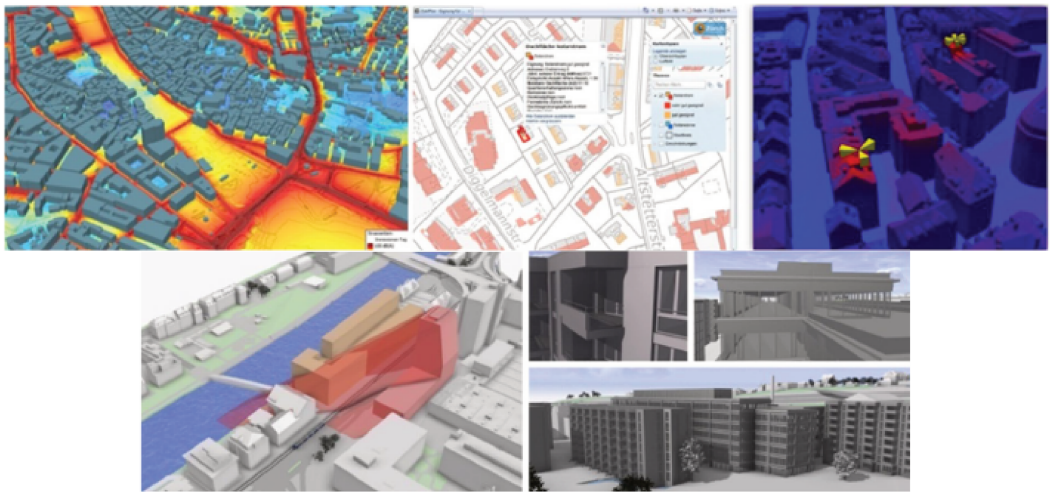

Figure 1.1.: Established applications of the "City of Zurich" Digital Twin: noise propagation (top left), air pollution (top-center), mobile phone radiation (top right), solar potential estimation (lower left), and construction project visualization (bottom-right) from Schrotter and Hürzeler [SH20].

When implementing these data-driven approaches, taking into consideration the underlying n-Dimensional (nD) characteristics of urban data is crucial for creating integrated views of the evolving urban landscape for end-users. For a long time, 2D maps, aerial photographs, and cadastral views have been used to manage the urban lifecycle. With the increasing availability of 3D data, thanks to new data acquisition methods (LIDAR, photogrammetry...), many applications are now possible with 3D urban data [Bil+15]. In addition to spatial information, some 3D urban data models require taking into account a temporal dimension (4D) to represent the changes of cities as they evolve over time [CK19; JSG20; SSG20].

These *spatio-temporal* (4D) models help historians and urban planners in particular to visualize and contextualize the impact of past key projects on the development of a city. Such studies may require the integration of urban models with a corpus of documents consisting of project plans, newspaper articles, archives, social media posts, etc. Integrating urban data stored in these different data formats or structured according to different data standards requires taking into consideration an abstract '*n*-th' dimension. This dimension refers to the semantic or thematic information used to define the meaning and structure of urban data as opposed to spatial and

[1] These processes may also include realizing real-world events such as signaling an update to a traffic light or adjusting the temperature of a building.

temporal information [Jai+21], as shown in figure 1.2. Defining the semantics of urban data is usually done by proposing data standards which may contain common vocabularies and data transfer formats like, for example, the CityGML[2] standard for creating semantic 3D city models [GP12].

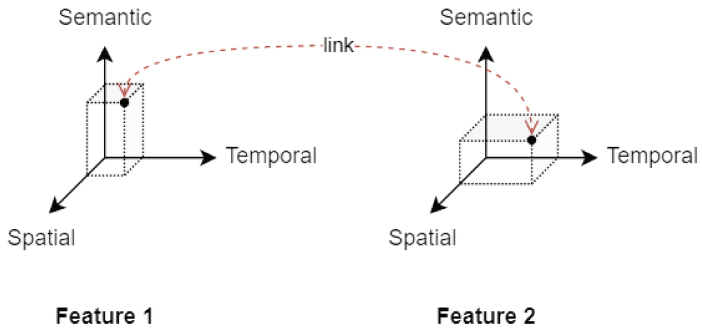

Figure 1.2.: An example of relating two representations of the same geospatial feature (real-world object or phenomena) in nD space (4D spacetime + semantic/thematic space) through a link (adapted from Jaillot et al. [Jai+21]). A real-world phenomenon can have spatial and temporal properties (i.e., where and when the representation exists) as well as semantic or thematic properties (i.e., who or what the representation is).

To ensure that the nD urban data are diffused in interoperable and uniform manners by data providers, different international organizational bodies (OGC[3], ISO[4], W3C[5], OMG[6]...) and research works may propose data standards or extensions of existing standards (for example, those illustrated in figure 1.3). These data standards may be composed of normative and informative *documentation* in natural language. In this context, documentation can define how data conformant to a standard should be structured and the semantic meaning behind different concepts and vocabulary used in the standard [PB11]. In addition, data standards may contain *data models* (for defining the abstract structure of data conformant to the standard) and *data schema* (for formally defining the structure of data in a machine-readable manner) [Kut16]. Data models and data schema can contain or may be supplemented with *domain-specific rules* for constraining data to be conformant to said model or schema [SMJ02]. Also, standards may propose rules to define how formal data schema should be derived from abstract data models as *encoding rules* [Kut16].

[2] https://www.ogc.org/standard/citygml/
[3] https://www.ogc.org/
[4] https://www.iso.org/home.html
[5] https://www.w3.org/
[6] https://www.omg.org/

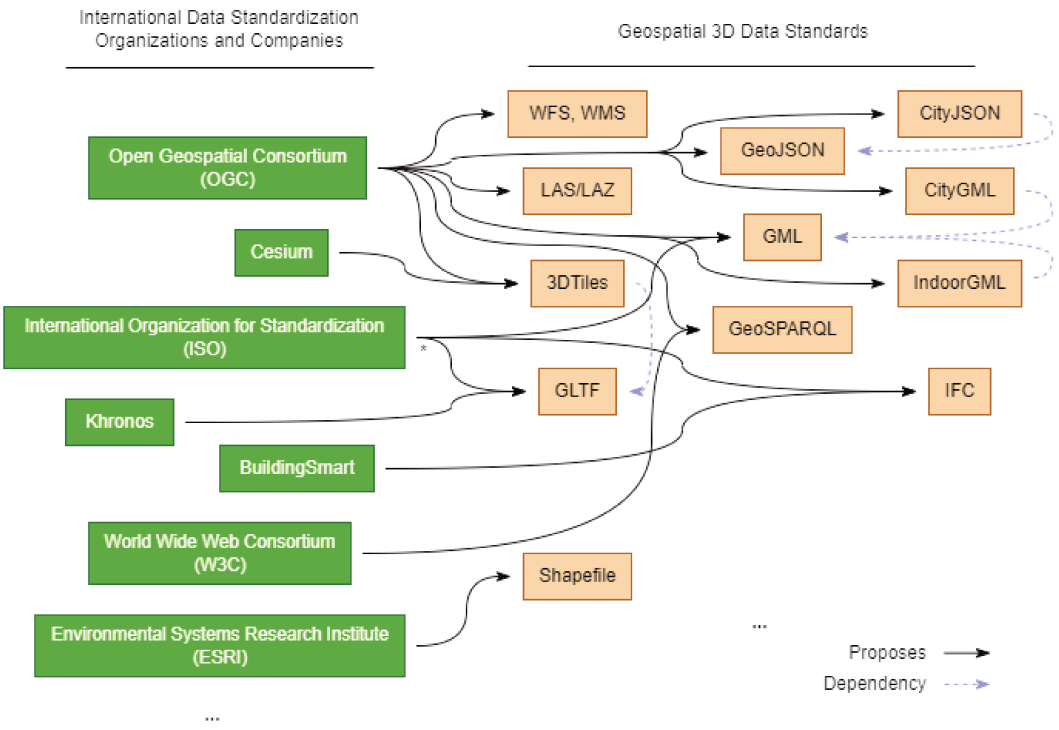

Figure 1.3.: A non-exhaustive illustration of different urban data standards and the international data standardization organizations that propose them.

Although data standards are used to create and share data in interoperable manners, many open urban data producers do not reuse the same data standards or propose their own internal, application-specific data structures and vocabularies [Bar+14]. This lack of standard reuse in open urban data is in part due to a demonstrated lack of cross-domain collaboration between governmental departments and research communities from different domains of urban information and is a cause of the rise of domain-specific *data silos* [Urb18]. These issues may also occur as data standards themselves evolve over time and older datasets are not forward compatible with newer iterations of the same standard. Resolving the interoperability gap between these data silos has led to the creation of international Spatial Data Infrastructures (SDI)—such as the INSPIRE initiative in Europe [TSD12; Bri18]. It is also one of the biggest technical challenges to the creation and maintenance of urban digital twin applications [Lei+23].

In this context, *urban data integration* is a process that combines multisource urban data from these heterogeneous information domains to create more complete views of the urban landscape and its evolution for the end-user [Tra+16]. To provide views in an interoperable manner, the integration process must take into consideration

existing urban data standards and their n-dimensional characteristics. The following section precisely defines the urban data integration problems and research questions addressed by this thesis.

1.1 Problem statement and research questions

In general, data integration requires confronting a wide variety of challenges. In the domain of nD urban data, this section identifies the two overarching problems this thesis proposes to resolve. First, diverse types of heterogeneity may occur amongst different nD urban data sources are identified. Second, data transformation or conversion processes (also referred to as data materialization [Tra+20]) are often utilized for urban data integration. However, these processes may risk semantic data loss as highlighted by Lei et al. [Lei+23]. This section also presents several research questions that arise from the identified problems.

P1: ***Interoperability gaps between heterogeneous (urban) data standards makes reusing these standards difficult.***

As is the case in most domains of urban information, heterogeneity with respect to urban data can take many forms and may require combining different integration approaches to resolve. To define heterogeneity, this thesis refers to Kutzner [Kut16] who synthesizes five different types of information heterogeneity from Leser, Naumann, et al. [LN+07] and the international ISO 19101 standard [ISO14]: semantic, technical, syntactic, structural/schematic, and data model. This thesis focuses on confronting three of these types (illustrated in figure 1.4):

P1.a: Semantic This heterogeneity refers to the meaning or interpretation of information [Kut16]. The intended semantics of a data model are often defined using natural language specification [AK03]. Semantic heterogeneity can appear when different data models define the *same* information using *synonyms* (e.g., two transportation data models that use the word *Road* and *Street* to refer to the same real-world phenomena) or *different* information using *homonyms* (e.g., the word *Pollution* to refer to noise pollution in one data model and air pollution in another).

P1.b: Structural/schematic This heterogeneity can appear when identical information is defined using different data structures from the same data modeling paradigm [Kut16]. e.g., in an object-oriented data model, using a character

string to store the type of geospatial feature as opposed to instantiating some enumeration.

P1.c: Data model This heterogeneity can appear when different data models define identical information [Kut16]. This commonly occurs when data models of the same information are created using different modeling paradigms. e.g., modeling information using classes and associations from an object-oriented modeling paradigm as opposed to tables in a relational database modeling paradigm.

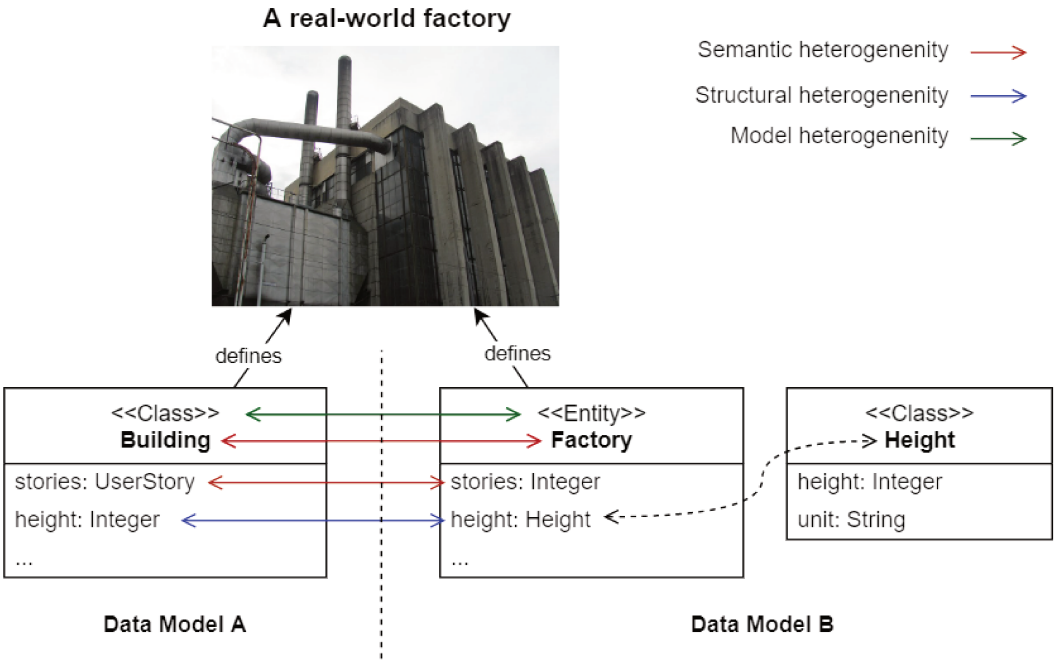

Figure 1.4.: An example of semantic, structural/schematic, and model heterogeneity between two models of the same real-world concept expressed with UML. Both models feature semantic heterogeneity twofold: through their use of the words *Building* and *Factory* to describe the same concept, and through *stories* to refer to two different concepts—a story of some type "UserStory" (left) and the number of physical building stories as an integer (right). Structural heterogeneity is exemplified through how *height* data is stored. The left model uses an attribute with an integer value, while the right implements an attribute referencing a class with its own height attribute. The last and most esoteric example is of data model heterogeneity. The left model defines the concept Building, with the UML-native *class* stereotype. The right model uses the stereotype, *entity*, a modeling concept that originates from different languages such as EXPRESS and is used similarly to UML classes.

These types of heterogeneity may occur for several reasons and as previously stated, can even occur between different versions of the same data model. Different domains of urban information use different vocabularies and different interpretations of urban

phenomena. Even time itself can be structured differently depending on the needs of a data integration use case [SCI18]. Each different type of heterogeneity can be viewed as a sub-problem of heterogeneous data integration. To understand how these sub-problems can be resolved in the context of nD urban data integration, different aspects of these data models and the data that conform to them must be taken into consideration, such as data modeling paradigm, language, model structure, level of model abstraction, data transfer formats, model semantics, etc. These concepts and their impact on nD urban data integration are explained in further detail in chapter 2.

P2: ***Data transformation between data standards may risk data loss, limiting the reusability of the transformed data.***

Transforming or converting data from one standard to another is a common approach in urban data integration within the context of creating and maintaining urban digital twins [Lei+23]. This is sometimes referred to as ETL (Extract-Transform-Load), a term that originates from data warehouse integration [Vas09]. Regarding (geo)spatial urban data, these approaches are sometimes referred to as spatial or semantic ETL [Dre+20; Pat+14; Kut16]. Many of the types of heterogeneity highlighted in the previous problem also occur when applying transformation approaches, and furthermore, the loss of *semantic* data (**P1.a**) during transformation has been identified as a barrier to the adoption of urban digital twin applications [Lei+23]. For this reason, data transformations must also consider the underlying data model (and data format) of the data to be transformed. This can reduce data loss [BA05] and consequently retain interoperability with existing standards after transformation and improve the reusability of transformed data.

Research questions

The problems from section 1.1 give rise to the following research questions that direct the work presented in this thesis:

- **RQ1:** *How can nD urban data integration approaches ensure that urban data standards can be easily reused, even as these standards evolve?*
- **RQ2:** *How can data loss be limited when transforming data between heterogeneous nD data formats?*

- **RQ3:** *How are model-driven transformation approaches affected by models at different levels of abstraction?*

1.2 Research context

Within this greater context, this thesis is funded by the Université Lumière Lyon 2 and takes place within the VCity research project[7] which explores scientific bottlenecks in the domain of 3D urban data integration, city evolution, digital city representations, and tools for visualizing and analyzing the evolving urban landscape. The VCity project and the research effectuated during this thesis also endeavors to collaborate with standardization organizations such as the Open Geospatial Consortium to further creation of urban data standards to meet the demands of experts from different urban data domains in academia, industry, and government. In addition, the project develops open-source tools within the UD-SV (Urban Data Services and Visualization) framework [Sam+23] to provide 3D urban data analysis tools for researchers and industry.

The objective of this thesis is to propose a *standards-based model-driven approach for integrating nD urban data*, where the conceptual models underlying the heterogeneous urban data sources are used to preserve the interoperability of data during transformation. This approach endeavors to be reusable for a wide variety of existing urban data standards, even as new standards are proposed, and these standards evolve over time. In addition, this thesis investigates how urban knowledge graphs can provide interoperable data formats to provide a basis for interoperability in the integration of n-Dimensional urban data. The final objective of this thesis is to ensure that urban data integrated through this approach can be easily reused by urban data-driven applications such as urban digital twins to provide users with more complete views of the evolving urban landscape (figure 1.5). The realized contributions of this work are denoted in the following section.

[7] https://projet.liris.cnrs.fr/vcity/

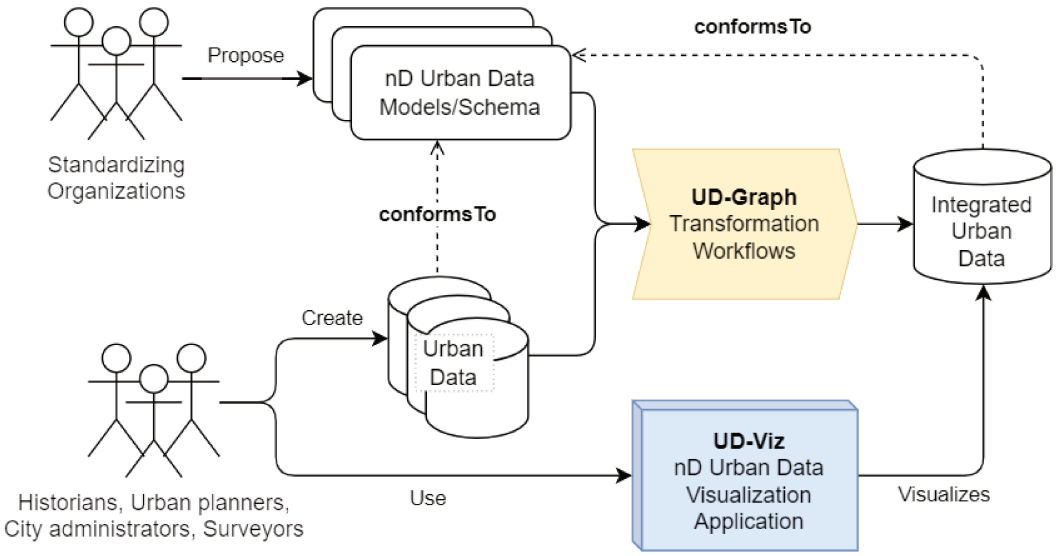

Figure 1.5.: An illustration of the different users and actors involved in the production and utilization of standards and standardized multisource nD urban data. In addition, this diagram shows an overview of how these standards and data are used by the proposed nD urban data transformation and visualization components (UD-Graph and UD-Viz, components of the UD-SV framework) to provide these users with integrated views of these data.

1.3 Contributions

Over the course of this thesis, the following scientific and technical contributions have been realized:

C1: Formalization of nD urban data models from evolving standards

To partially respond to **RQ1** and **RQ3**, several ontological nD urban data models are formalized based on the CityGML[8] standard (a widely adopted OGC standard for creating 3D semantic city models [GP12]). At the time of writing this work, version 3.0 of the CityGML standard has been published by the OGC including an GML specific encoding. This work proposes two model-driven transformations (contribution **C5** below) of the different modules of the CityGML 2.0 and 3.0 conceptual and schematic data models towards expressive, formal knowledge graph languages and formats for formalizing new interoperable representations of these evolving standardized data models. The first data model (contribution **C1.a**) is

[8] https://www.ogc.org/standards/citygml

created from less abstract XML Schema while the second data model (contribution **C1.b**) is based on geospatial UML to OWL transformation standards (ISO 19150-2[9]). The formalized data models reuse Semantic Web standards such as SKOS[10], OWL-Time[11], and GeoSPARQL[12] to increase their interoperability across information domains.

C2: Formalization of data model extensions for navigating concurrent scenarios of urban evolution

Data model extensions are often proposed when existing standards are insufficient for structuring data in specific urban use cases or information domains [BKN18]. To complement contribution **C1**, and better understand how this integration approach can be applied to extensions of data models, this work proposes model-driven transformations of these extensions that can be executed in parallel with the proposed transformations in **C1**. Ontological extensions to the proposed nD urban data models are also formalized using this method. In particular, an ontological model based on the Workspace CityGML Application Domain Extension (ADE) [SSG20] is integrated using this approach for representing concurrent and parallel scenarios of urban evolution.

C3: Formalization of rules for validating scenarios of evolution of the urban landscape

To complement the formalized data models (**C1**) and their extensions (**C2**), and respond to **RQ2**, logical domain rules for structuring and implementing data conformant to these models are also formalized. When integrating standardized data, these rules can be used to validate that data integrated by transformation and conversion techniques or existing data are conformant to their original data standards. For this, rules are proposed for validating basic temporal relations (contribution **C3.a**) between temporal entities such as instants and intervals of time. In addition, an existing ruleset from the CityGML Workspace ADE [SSG20] is reformalized for validating concurrent viewpoints of urban evolution (contribution **C3.b**). To be able to exploit these rules alongside the formalized ontological data models they

[9] https://www.iso.org/standard/57466.html
[10] https://www.w3.org/TR/skos-reference/
[11] https://www.w3.org/TR/owl-time/
[12] https://www.ogc.org/standard/geosparql/

constrain, rules are formalized in machine-readable Semantic Web formats such as SWRL[13].

C4: Development of a model-driven data transformation workflows

To respond to **RQ2** and **RQ3**, several modular data transformation workflows are proposed through **UD-Graph**, a data integration tool (as a component of the UD-SV framework). UD-Graph contains a set of transformation and nD urban data validation tools. These tools can be used to orchestrate transformation workflows (or data transformation pipelines) for geospatial data from XML formats to RDF formats and validate the transformed data. The transformations effectuated by UD-Graph take into consideration XML and RDF geospatial data standards during transformation (GML, and GeoSPARQL) (contribution **C4.a**). These model-driven workflows can also use OWL ontologies created from UML to OWL transformations to enable data integration based on more abstract UML models (contribution **C4.b**). After transformation, constraints and rules from the data model can be used to validate the conformity of the transformed data to their original data models. To effectuate this validation activity in a reproducible manner, a proof-of-concept test suite was added as a component of UD-Graph (contribution **C4.c**). These tools and workflows are used to produce contributions **C1–C2**.

C5: Production of nD urban datasets from real-world open data

Additionally, to respond to **RQ1-3**, open CityGML data from the metropole of Lyon, France was transformed using the formalized CityGML 2.0 and 3.0 ontological models and UD-Graph transformation workflows. This data captures snapshots of the different districts of the Lyon metropole at different instances of time (2009, 2012, 2015, and 2018). Using the constraints of the ontological model and logical rules proposed in **C1–3**, these data were validated as conformant and logically consistent to the CityGML model and its relevant extensions. In particular, several datasets were transformed including 3D city models of the 1st arrondisement of Lyon (contribution **C5.a**), the campus of 'La Doua' of Villeurbanne (contribution **C5.b**), and the 'Gratte-Ciel' neighborhood of Villeurbanne (contribution **C5.c**).

[13] https://www.w3.org/Submission/SWRL/

C6: Demonstration of nD urban integration applications for navigating amongst scenarios of evolution

To further respond to **RQ1-3** (but especially **RQ3**) and demonstrate how this integration approach can be implemented in conjunction with other data materialization approaches and in different nD urban data integration contexts, several nD urban data web applications are developed. These applications provide users with methods to visualize and navigate the ontological data models, their extensions, and their conformant data integrated and validated using the previous contributions **C1–5**. To integrate this nD urban data in a manner that facilitates the exploitation of its spatial, temporal, and semantic components multiple transformation workflows are utilized based on the needs of each application. These workflows include transformations from UD-Graph in concert with other transformation approaches such as those proposed by Pédrinis et al. [PMG15] and Marnat et al. [Mar+22]. The proof of concept demonstrations are proposed as extensions of the **UD-Viz** (Urban Data Visualization) framework for creating web applications for visualizing and interacting with geospatial 3D urban data [JSG20; Col+22]. Additionally, the demonstrations are publicly available as well.

C7: Reproducibility

Containerization technologies (e.g., Docker containers) and code archives such as Software Heritage are used to ensure the works proposed during this thesis are accessible and reproducible. Containers are used to encapsulate each step of the proposed nD urban data integration workflows (UD-Graph) and proof of concept demonstrations (UD-Viz).

C8: Contributions to the CityGML 3.0 GML Encoding

As a part of this research, contributions were also made to the CityGML 3.0 GML Encoding[14] Sub-Working Group (SWG) of the OGC. These contributions take the form of participation in discussions and editing of the encoding documentation and the creation of example datasets for demonstrating how evolving urban data can be structured according to the standard.

[14] https://github.com/opengeospatial/CityGML3.0-GML-Encoding

1.4 Outline

The rest of this work is structured as follows:

Chapter 2 introduces key concepts in data modeling and their use in the domain of geospatial and spatio-temporal urban data.

Chapter 3 presents a literature review of the various kinds of nD urban data considered in this domain, applications of nD urban data integration, different approaches in integrating these data, and what knowledge or information gaps exist in the state of the art.

Chapter 4 presents the proposed approach used to integrate 3D urban data models and datasets through model-driven transformation based on physical data models.

Chapter 5 presents the proposed data integration methodology—complementary to the work proposed in chapter 4—relying on model-driven transformations based on conceptual data models. In addition, this chapter presents basic rules for validating temporal relations of urban data.

Chapter 6 presents several technical contributions developed during the course of this thesis including applications of the proposed integration methodology and data standardization contributions to the CityGML data standard from the Open Geospatial Consortium.

Finally, chapter 7 concludes the research proposed during this thesis with a synthesis of the contributions and a discussion of the future research directions of this research.

2 N-dimensional modeling of the urban environment

As introduced in section 1, nD urban data is structured according to a variety of data standards for digitally representing urban phenomena through the spatial, temporal, and semantic dimensions. To understand how data standards and research works define the structure and semantics of the data that conforms to the standard, this chapter introduces and describes key concepts in geospatial and temporal data modeling and data standardization, and provides a literature review of their application in modeling the evolution of the urban environment. The chapter is structured as follows: section 2.1 defines what data models and metamodels are, their characteristics, and how they are generally used in the geospatial research domains to define the meaning and structure of data; section 2.2 introduces the fundamental concepts of ontological modeling in developing interoperable urban data models; section 2.3 introduces different approaches in modeling changes over time in geospatial and urban data contexts; section 3.4 briefly synthesizes the content presented in this chapter.

This chapter and the rest of this dissertation uses the namespace prefixes listed in appendix C.1.

2.1 Data modeling and data models

Models are used in computer science in both academia and industry as methods for abstracting and decomposing complex problems and real-world systems. Born out of mathematical models like set theory, data models have seen widespread adoption for these purposes since the early 1970s—such as the relational model from Codd [Cod70], or the entity-relational model from Chen [Che76]. Originally, these models were implemented in the domain of database development. More recently, these models are used as a basic practice in data standardization to define the structure and meaning of data (i.e., semi-structured and structured data) in addition to real-world phenomena.

In geospatial research domains (in addition to many other fields in philosophy and mathematics [Kut16]) what is being modeled—for example, a specific system or some broader domain of information—is sometimes referred to as the universe of discourse, domain of discourse, or domain of interpretation [GOS09; ISO14]. The ISO 19101 standard for modeling geographic information defines the term **Universe of Discourse** (UoD) as a "view of the real or hypothetical world that includes everything of interest" [ISO14]. The UoD is typically considered something that cannot be modeled in a universal manner as there are potentially infinite interpretations of a UoD [GOS09]. Accordingly, data models define the structure and semantics of data instances (or data), which in turn are used to represent real-world objects and phenomena from a UoD as shown in figure 2.1.

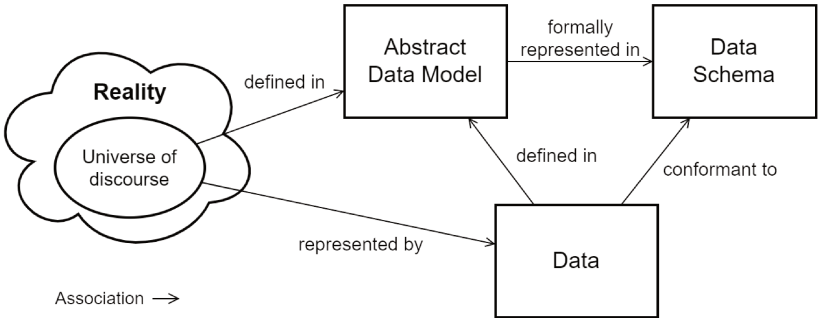

Figure 2.1.: The relationships between a universe of discourse, an abstract data model, and data (adapted from [Kut16]).

Furthermore, data models can describe or define data at different levels of abstraction. Typically, three levels of abstraction are used in database modeling: conceptual, logical, and physical [Avi91]. At the conceptual level, data is abstractly defined with a *conceptual* data model without taking into consideration physical hardware, data formats, or database architectures. The logical level uses less abstract *logical* data models that take into consideration the intended information systems data are stored in (e.g., a relational database) [Avi91]. Logical data models are often constructed by refining an existing conceptual data model [Avi91]. At the physical level, description languages can be used to define, *physical* data models (or data schema as shown in figure 2.1) which take into consideration specific data formats or database architectures (i.e., SQL definitions of tables).

Kühne [Küh06] notes that the word 'model' can be applied to data models as well as data. For example, in geospatial and urban information domains, digital 2D and 3D city models are composed of data (see figure 2.2). Conversely, a building data model may classify the different concepts relevant to the structure of buildings (see figure 2.2). To distinguish between these two implementations of the word 'model',

this dissertation uses the term 2D/3D model to refer only to data. Additionally, references to 3D, 4D, or nD data models or schema do not refer to data instances.

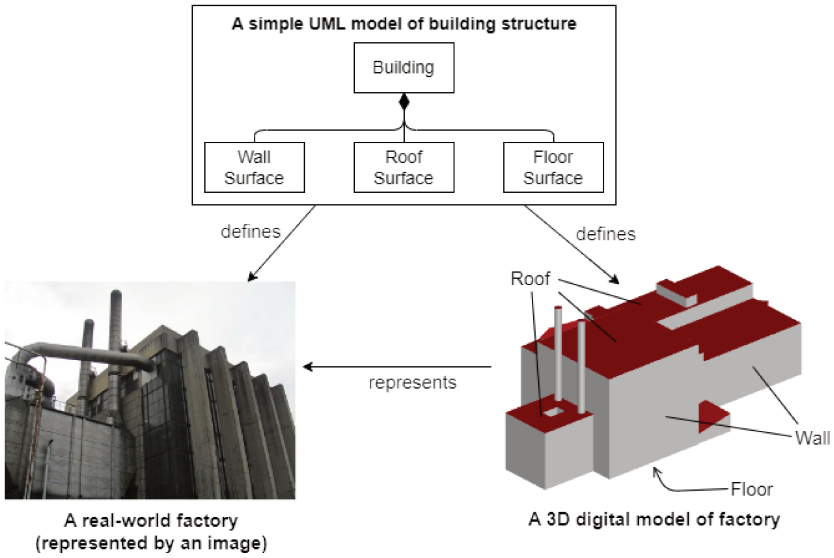

Figure 2.2.: Example of different "models" of a real-world factory. In this context, a 3D digital *model* of the factory (bottom right) represents the factory (bottom left). The abstract building hierarchy (top) is a data *model* of both the factory and the 3D model of the factory. The factory image is taken from Commons [Com22].

The following section discusses how data are modeled and/or described using modeling languages and description languages.

2.1.1 Language, formality, and syntax

In data modeling, there exist two types of languages: formal and informal languages [Kut16]. *Formal* languages are used to define unambiguous, *machine-readable* models (like physical data models or schema). An example of *informal* language is natural language (e.g., English, Japanese, etc.). Much like how natural language can be used to communicate a model orally, formal languages can be used to digitally store and share models. Atkinson and Kühne [AK03] states that a language is composed of 4 elements:

Abstract syntax The concepts from which a model can be created. For instance, a UML model is defined (abstractly) using the metaclasses from the UML model.

Concrete syntax A concrete rendering of these concepts. For example, a UML model can be written down using UML notation [AK03; ISO14]. Concrete

syntaxes can be *graphical* (as is the case with UML notation) or *lexical*, e.g., XMI (XML Metadata Interchange) is an XML data format used to store UML models. Data transfer formats may provide languages a concrete syntax.

Well-formedness *Rules* (or *constraints*) for applying concepts from the language. In the case of UML, Object Constraint Language (OCL) is often proposed to formally define logical rules on concepts defined in UML. Typically, logical languages like first order logic can be used for this purpose.

Semantics Description of *meaning* in a model. Atkinson and Kühne [AK03] propose declaring the semantics of a model informally through natural language specification.

Additionally, every modeling or description language exhibits some modeling paradigm with different paradigms being suited for different applications in data modeling as stated by Kutzner [Kut16]. Some of the earliest paradigms are the relational and entity-relational (ER) paradigms as previously mentioned. This dissertation discusses three paradigms of data modeling or data description languages in particular that are often used in modeling and describing geospatial information.

Object-oriented The *Object-oriented* paradigm is one of the most widely used modeling paradigms at different levels of abstraction. As discussed in [Kut16], this paradigm introduces the concepts of *classes* to define types of entities and *objects*, the instances of classes. *Associations* define the relationships between classes while *attributes* are the characteristics of a class. This paradigm also introduces modeling concepts such as *inheritance* and *cardinality*.

The object-oriented paradigm is widely adopted by the OGC in geospatial data specification through data modeling languages such as UML [Ope22]. Standardizing organizations such as the OMG have utilized the object-oriented modeling paradigm to propose data modeling frameworks (discussed in section 2.1.2) and model-driven transformation approaches (discussed in section 3.2.1).

XML XML (eXtensible Markup Language) is a description language for data on the web [Zis00]. This paradigm introduces the *element* concept. In XML, an element is a data instance which may contain *attributes* and other elements [BA05]. These elements can be combined into hierarchical, tree-like data structures. Additionally, each element is identified by a *tag* that is composed of a *namespace* URL (that may be represented by a prefix) and a *local name*. Listing 2.1 shows is an example of XML elements representing building information. An element with the tag `ns:City` (composed of the prefix `ns`

representing some URL namespace and the local name `City`) contains an attribute `id` and child elements containing information regarding a building and the building's height.

```
<ns:City id="Sydney">
  <ns:Building id="building_3">
    <ns:metricHeight>6</ns:metricHeight>
  </ns:Building>
</ns:City>
```

Listing 2.1: Example XML data defining representing a city and a building within the city.

XSD (XML Schema Definition) is the current language used to precise the structure of XML data (replacing Document Type Definition (DTD)). XSD uses 24 different concepts—sometimes referred to as 'constructs'—with *complexType*, *simpleType*, and being among the most commonly used to define the concepts and hierarchical structure of XML data [Bed+11]. A complex type defines an element that may contain purely textual information or other elements, while simple types may only carry text. *Groups* and *AttributeGroups* are used to refer to declare predefined content of elements or attributes to be used in complex and simple type definitions. *SubstitutionGroup* specifies elements that can be used interchangeably and are generally considered to function similar to inheritance in an object-oriented modeling paradigm [FZT04; Bed+11]. Also, *extensions* and *restrictions* can be declared to specify what valid content an elements or XML datatype may contain. *Compositors* like `xs:all`, `xs:sequence`, and `xs:choice` can declare what child elements an XML element may contain. Finally, the semantic meaning of data can be stored in natural language using the *annotation* construct [Bed+11].

Listing 2.2 contains an example of an XSD definition of the building XML data in the previous example. The `City` element is declared as a complex type (line 1) that contains the attribute group named `"hasID"` and the element `"Building"`. The `"hasID"` attribute group defines the attribute `id` (line 32) and the `Building` element, which is a complex type that also contain the `"hasID"` attribute group. The semantic meaning of the `Building` element is defined in natural language by the `xs:annotation` element as documentation (line 11). Additionally, the `Building` element can contain either a `"metricHeight"` (line 20) or a `"imperialHeight"` (line 25) child element (but not both) through a `xs:choice` construct. These child elements are defined as simple types that can contain only integer values as text.

```
<xs:City>
  <xs:complexType>
```

```
    <xs:attributeGroup ref="hasID"/>
    <xs:all>
      <xs:element ref="Building"/>
      ...
    </xs:all>
  </xs:complexType>
</xs:City>
<xs:Building type="BuildingType">
  <xs:annotation>
    <xs:documentation>
      Used to store building information
    </xs:documentation>
  </xs:annotation>
</city:Building>
<xs:complexType name="BuildingType">
  <xs:attributeGroup ref="hasID"/>
  <xs:choice>
    <xs:element name="metricHeight">
      <xs:simpleType>
        <xs:restriction base="xs:integer"/>
      </xs:simpleType>
    </xs:element>
    <xs:element name="imperialHeight">
      <xs:simpleType>
        <xs:restriction base="xs:integer"/>
      </xs:simpleType>
    </xs:element>
  </xs:choice>
</xs:complexType>
<xs:attributeGroup name="hasID">
  <xs:attribute name="id" type="xs:string"/>
</xs:attributeGroup>
```

Listing 2.2: An example XML Schema defining the structure and semantics of building data within a city.

XML is also used as a concrete syntax for representing other data and data models from other modeling paradigms (e.g., XMI, the data transfer format for storing UML models). Many geospatial ISO and OGC data standards rely on the XML modeling paradigm such as GML, KML, CityGML, IndoorGML, etc.

RDF The RDF (Resource Description Framework) paradigm is based on a directed edge-labeled graph formalism [Hog+21]. This paradigm introduces the concept of a *triple*, that composed of a *subject*, *predicate*, and *object* where the predicate denotes a unidirectional relationship between the subject and object (shown in figure 2.3). Identifiers in RDF are composed of a namespace and an

optional *fragment* similar to XML. These identifiers are referred to as *IRI*s or *URI*s (International/Universal Resource Identifier) as they are intended to be internationally unique.

The RDF abstract syntax is defined by RDFS (RDF Schema). This language introduces three concepts: *class*, *property*, and *datatype*. Properties are used to define predicates between other classes, instances of classes, and primitive datatype values. Both classes and properties also may use inheritance. Additionally, what entities a property may relate to and from can be declared as the properties *domain* and *range* respectively.

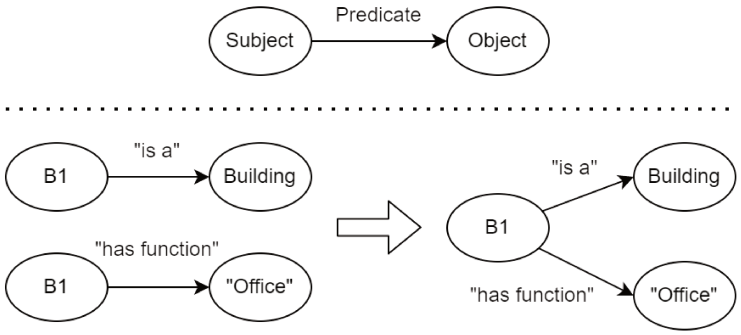

Figure 2.3.: A basic RDF triple (top) and an example of combining triples to form a graph (bottom).

2.1.2 Metamodeling

Metamodeling is an approach to modeling promoted by the Object Management Group's[1] (OMG) as the four-layer metamodeling framework. This framework used to organize the relationships between real-world phenomena and models, and modeling language [AK03]. Level M_0 contains phenomena from the real world, i.e., what is being modeled [Sei03]. Data and code itself can be placed at this level, represented as artifacts like digital documents on a file system, database, or in memory. Models at level M_1 are used as type models of phenomena from the real world and data at level M_0, i.e., data models, data schema, and computational ontologies. Models at level M_2 are used to define the abstract syntax of modeling languages that can be used to define models at level M_1 [Küh06]. Models at this level are considered *metamodels* as they are models used to define models. Models at level M_3 are used to define the language of modeling languages. This is the most

[1] https://www.omg.org/index.htm

abstract level of models, as by definition a model of a language that can define a language has the capability of recursively defining itself.

Concepts from a model can be used to *instantiate* elements of another model in the same level or in a lower level. Kühne [Küh06] referred to intra-level instantiation as *ontological* instantiation and inter-level as *linguistic* instantiation. Note that in UML, these relationships are referred to as "snapshot" and "instanceOf" respectively. Today, the OMG's Meta-Object Facility[2] (MOF) is used as a meta-metamodel for defining metamodels such as UML. Figure 2.4 illustrates these metamodeling layers in relation to the real-world using examples from MOF and UML.

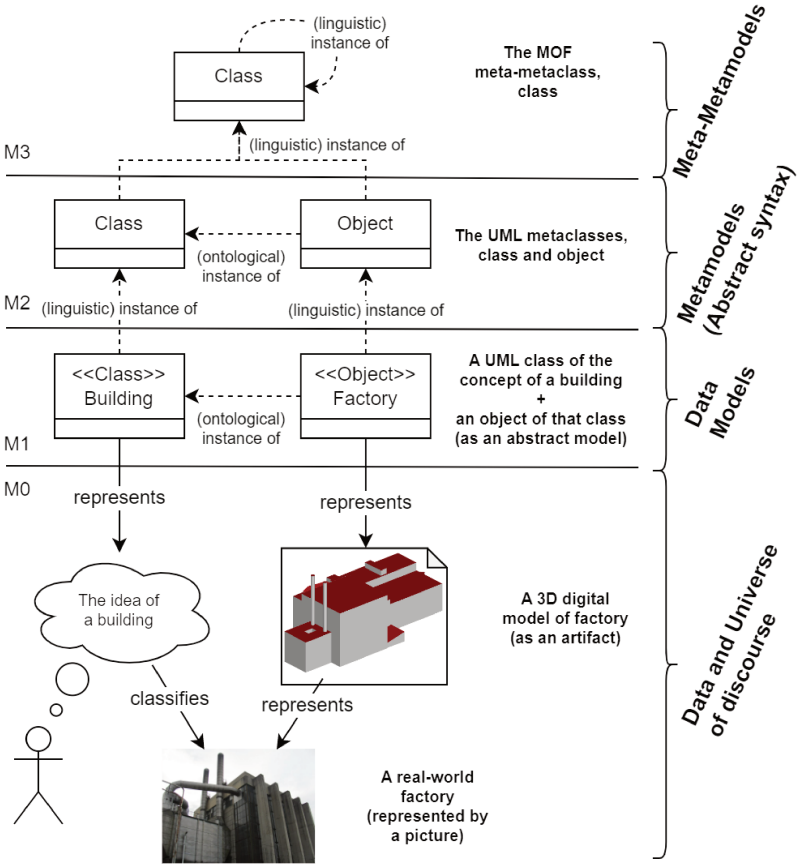

Figure 2.4.: The OMG four-layer metamodeling framework adapted from Atkinson and Kühne [AK03] and Kühne [Küh06]. Note that in the M_0 layer, intangible data artifacts and ideas both "exist" alongside real-world phenomena while also being representations of real-world phenomena themselves.

Among GIS standards, the General Feature Model (GFM) from ISO 19109 is an often used metamodel for creating geospatial data models proposed by the ISO

[2] https://www.omg.org/spec/MOF

Technical Committee 211 (ISO/TC 211) [PB11; HRB20]. The GFM provides abstract geospatial metaclasses for concepts such as *features* (a common term in the geospatial community for an "abstraction of real-world phenomena" [ISO15b]). This metamodel is sometimes paired with ISO 19103, which extends UML with modeling concepts such as `<<CodeList>>`—defined as "a flexible enumeration that uses string values for expressing a list of potential values"—and `<<Union>>`—defined as "a type consisting of one and only one of several alternatives (listed as member attributes)" [ISO15a].

These metaclasses are extensions of the UML modeling language and are provided as UML profiles (figure 2.5) to provide abstract interoperable vocabularies between the geospatial standards that employ them. Additionally, the ISO 191xx series proposes several *Abstract Schemas* as UML models that provide abstract classes for representing geospatial geometry (ISO 19107) and time (ISO 19108) [HRB20]. These abstract schema are used by more domain-specific *Application Schemas* (which are still formalized using modeling languages like UML) such as the CityGML 3.0 and Land Infrastructure (InfraGML) data models proposed by the OGC (also illustrated in figure 2.5) [JOH19]. Encoding rules or transformation rules are also proposed as a part of the ISO 191xx series of standards to represent the abstract metaclasses and classes from these models in technology-specific data formats. For example, ISO 19136 proposes an encoding of the abstract geometric classes from ISO 19107 to the geometric types proposed by the GML standard in the XML syntax defined by XML Schema[3]. These encoding and transformation rules are further explored in chapter 3.

[3] `https://schemas.opengis.net/gml/`

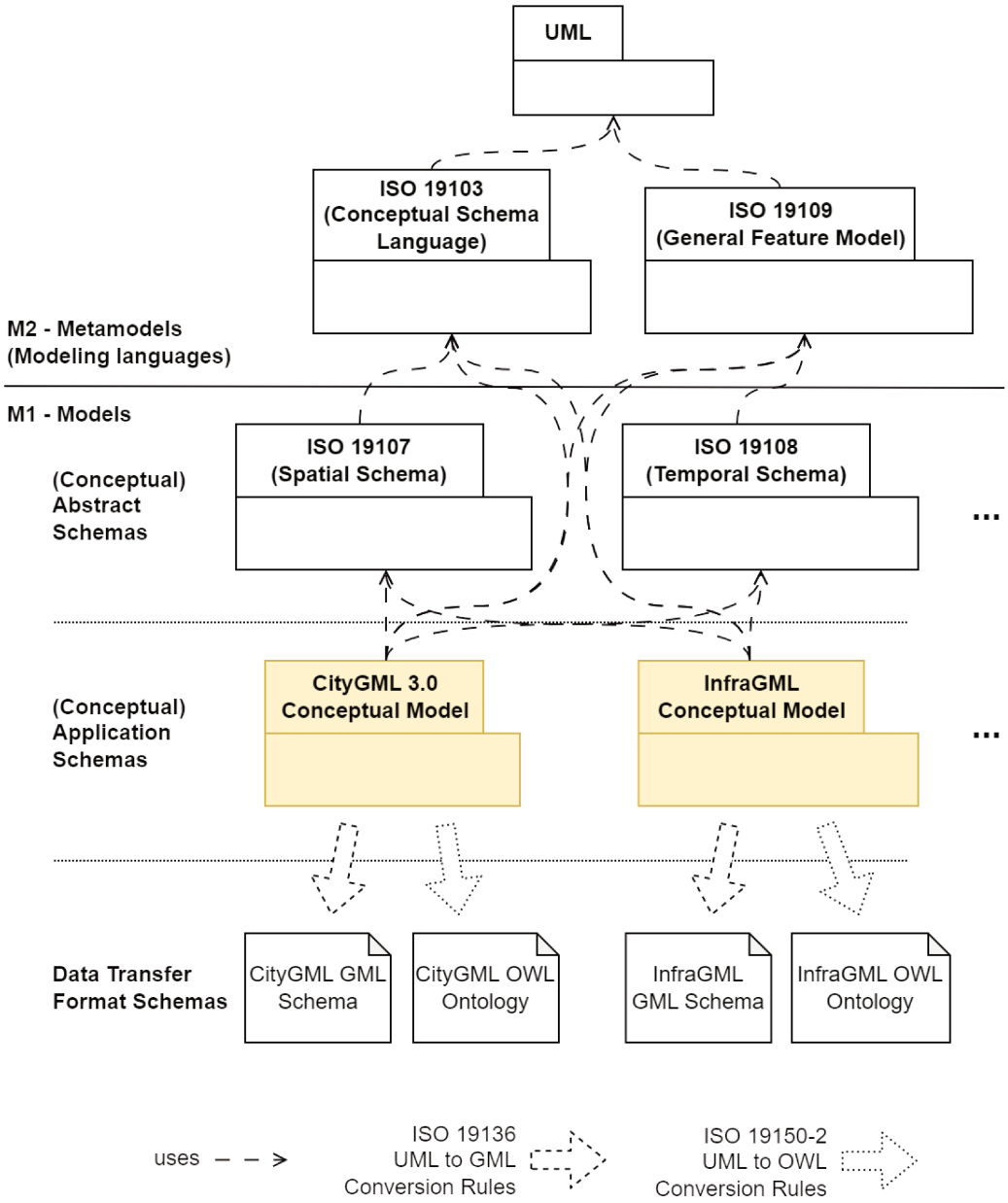

Figure 2.5.: Illustration of relationships between the General Feature Model, the CityGML UML model, and possible encodings of these UML models in GML and OWL based on [JOH19].

2.2 Ontologies and knowledge representation

Ontologies are used in the geospatial research domains to provide a basis for exchanging data across geospatial information domains in industry and academia [Cla20]. Guarino et al. [GOS09] and Uschold and Gruninger [UG04] specify the term *concep-*

tualization as an abstract, simplified model or view of a state of a UoD. In this context, a computational *ontology* is defined as "*a formal, explicit specification of a shared conceptualization*" [SBF98]. Like the definition of well-formedness [AK03], logical languages are typically considered sufficiently formal for representing ontologies (figure 2.6).

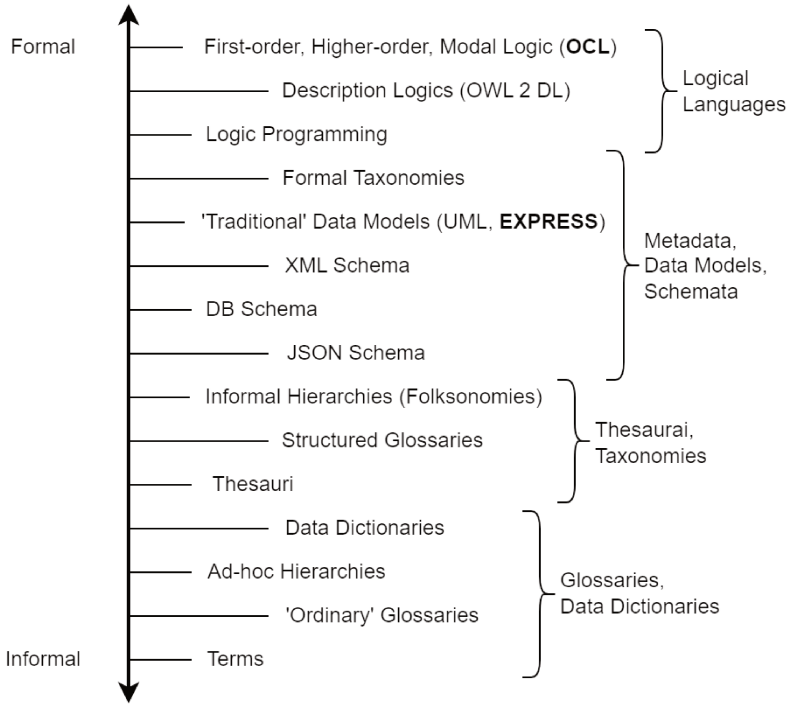

Figure 2.6.: Different language approaches to formalizing ontologies ordered by formality according to Guarino et al. [GOS09] based on Uschold and Gruninger [UG04]. The bottom features relatively simple languages where that can only specify ontologies as sets of terms while the top provides the means to define ontologies as 'rigorously formalized logical theories' [UG04]. New language examples are proposed in bold.

Logical languages are often used to infer new information within an ontology through *entailment* and automated *reasoning* [UG04; Hog+21]. For example, languages that permit defining horn-like rules can define logical 'if this then that' axioms in the ontology. These rules are composed of an *antecedent* and a *consequent* where the antecedent and the consequent are (potentially empty) sets of logical statements such that if the conjunction of all the atomic statements of the antecedent holds true, then the conjunction of all the atomic statements of the consequent must hold true [Hor+04] (equation 2.1).

$$antecedent \rightarrow consequent \tag{2.1}$$

This can be used to define a statement such as *every building that neighbors another building is symmetrically neighbored by that building*. For example, using the basic description logic language ($\mathcal{AL}$) provided in appendix C.2 (that is used throughout this dissertation), we state that given the concept descriptions C and D, the concept $Building$, and the role $neighbors$:

$$\forall C \forall D(Building(C) \wedge Building(D) \wedge neighbors(C, D)) \rightarrow neighbors(D, C) \tag{2.2}$$

In this example, a reasoner would infer that for every statement in the ontology where the antecedent holds true, the consequent is inferred as true.

Often Description Logic (DL) languages (subsets of first-order logic) are used to formalize ontologies required in inferencing applications as they have a good balance of *expressivity* versus computational *efficiency* [GOS09]. Expressivity or expressive power refers to the capabilities offered by the language such as restriction of cardinality, negation, etc. Languages that are more expressive are often more computationally inefficient for reasoning. Languages that are too expressive such that there is no finite algorithm for entailment are considered *undecidable*. Logical languages have also been widely used in knowledge representation and Knowledge Base (KB) and knowledge graph creation where knowledge is divided into a **TBox** and an **ABox** respectively representing the *terminology* or vocabulary of some UoD and the *assertions* or instances of these concepts as shown in figure 2.7.

The W3C has proposed the Semantic Web technology stack of data standards for representing these kinds of data including RDF and RDFS (figure 2.8). Within these standards is OWL[4] (Web Ontology Language), a widely adopted language for defining ontologies on the web. As an extension of RDFS, OWL ontologies can be represented as RDF graphs. Here, OWL specializes the concept `rdfs:Property` with `owl:ObjectProperty` and `owl:DatatypeProperty`. Object properties represent relationships between individuals (asserted class instances) while a datatype properties represents relationships between individuals and primitive datatype values. Also, OWL DL (a subset of OWL) is often used for reasoning applications as it is designed to provide expressive power while still being decidable [HHP12].

[4] https://www.w3.org/TR/owl2-primer/

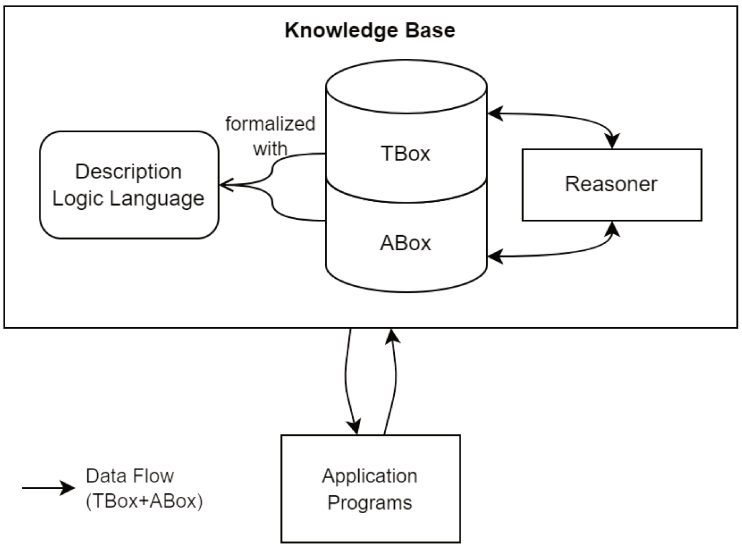

Figure 2.7.: Illustration of a description logic-based knowledge representation system (e.g., a knowledge graph or knowledge base) adapted from [BN03]. In this diagram data flow encompasses the TBox (which represents the data model of the data) since it is accessed in a functionally equivalent manner to data instances (ABox).

In addition, SPARQL[5] is a query language and protocol for accessing updating data stored in RDF databases (sometimes referred to as 'RDF-stores', 'Triple-stores', or 'SPARQL endpoints') [Bui+]. In SPARQL, `INSERT`, `SELECT`, `UPDATE`, and `DELETE` statements can be used to perform basic database operations. These operations identify the RDF data to be retrieved or manipulated through *graph patterns* which are described in an RDF-like triple syntax. `CONSTRUCT` queries are used to retrieve data in an RDF graphs format instead of a tabular format like SQL. Reasoning can be applied to RDF graphs to infer new information [Tra+16]. For example, listing 2.3 presents an example SPARQL query that could be used to reformalize the rule previously proposed in equation 2.2.

```
INSERT {
  ?D :neighbors ?C .
} WHERE {
  ?C a :Building ;
     :neighbors ?D .
  ?D a :Building .
}
```

Listing 2.3: A query for inferring a symmetrical property 'neighbors' between asserted building individuals.

[5] https://www.w3.org/TR/sparql11-query/

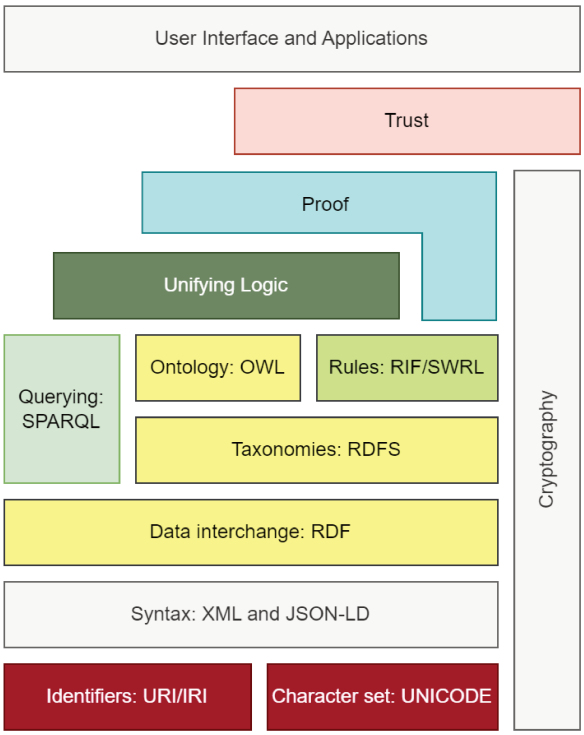

Figure 2.8.: The semantic web technology stack from [BHL01] (updated with the JSON-LD syntax).

Also, as a part of the Semantic Web technology stack, SWRL[6] is proposed as a DL language based on several sublanguages from OWL and Rule Markup Language that permits the definition of horn-like rules in OWL [Hor+04]. Several SWRL syntaxes have been proposed—such as RDF, XML, and OwlReady2[7]—for defining SWRL rules. Following the previous example, the rule proposed in equation 2.2 could be reformalized in SWRL as shown in listing 2.4.

```
1 :Building(?C), :Building(?D), :neighbors(?C, ?D)
2   -> :neighbors(?D, ?C)
```

Listing 2.4: A SWRL rule for inferring a symmetrical property 'neighbors' between asserted building individuals with the OwlReady2 syntax. Note that in this syntax conjunctions are denoted by commas.

Additionally, some ontologies are created using either a Closed-World Assumption (CWA)—which assumes that what is not explicitly stated is assumed false—or an Open-World Assumption (OWA)—which assumes that what is not explicitly stated is unknown [Hog+21; Bri+14]. OWL ontologies are typically defined using OWA

[6] https://www.w3.org/Submission/SWRL/
[7] https://owlready2.readthedocs.io/en/latest/rule.html

whereas frame-based languages such as UML models are always defined using CWA. For example, in UML a class may only be instantiated with the attributes and associations stated in the UML model where it is defined. However, individuals of an OWL ontology may be asserted with any property unless stated otherwise.

In the urban geospatial domain, several spatial ontologies exist such as the BOT (Building Topology Ontology) [Ras+20] used in BIM for defining the topological relationships between 3D spaces in buildings and the OGC's GeoSPARQL standard. The GeoSPARQL standard serves two-fold as a geospatial extension[8] to the SPARQL query language of the Semantic Web technology stack and a vocabulary for geospatial concepts[9] [BK12]. The GeoSPARQL ontology provides the classes `geo:Feature` and `geo:Geometry` typically used as abstract superclasses for representing geospatial features and geospatial geometry (figure 2.9). Asserted individuals of these classes can be related though the object property `geo:hasGeometry`. Integration of the GeoSPARQL ontology with a more domain-specific ontology is typically done by asserting classes representing real-world phenomena (from the domain-specific ontology) as subclasses of the `geo:Feature` and likewise for spatial or geometric classes and the `geo:hasGeometry` class (also shown in figure 2.9). For example, the GML standard has proposed an ontology[10] that extends the GeoSPARQL ontology to represent GML-specific geometric concepts.

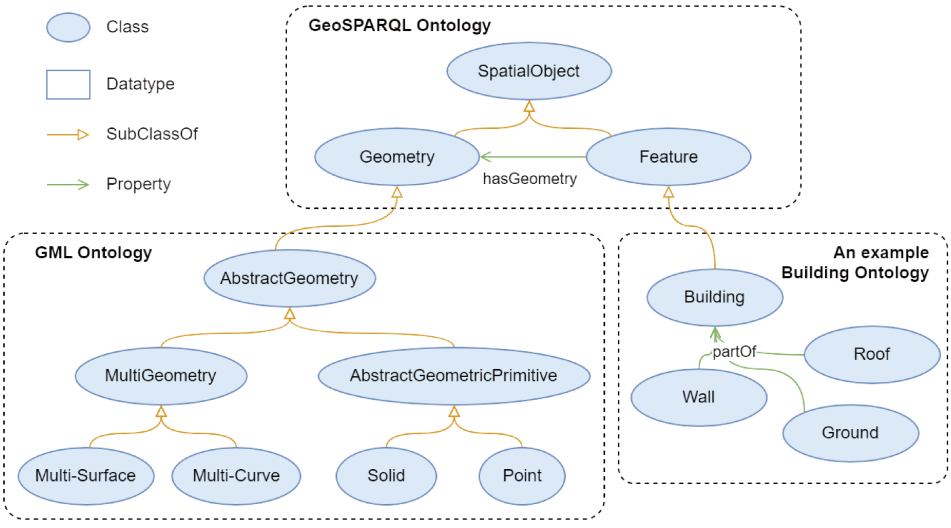

Figure 2.9.: The GeoSPARQL and GML ontologies integrated with an example building ontology.

[8] https://www.ogc.org/standard/geosparql/
[9] http://www.opengis.net/ont/geosparql
[10] https://schemas.opengis.net/gml/3.2.1/gml_32_geometries.rdf

These key concepts in data modeling and description, and knowledge representation provide a solid basis for data standardizing organizations and research works to define data models as conceptualizations of real-world phenomena.

2.3 Modeling the temporal dimension

Concerning geospatial urban information, digital representations of the changes of city objects over time are often based on the *identity* of (urban) objects as the object undergoes changes over time [Ber19]. In this context, identity is the characteristic that distinguishes one object from another and changes to the object may affect an object's identity [HE00; Ste+08; Van+15]. In general, geospatial features may undergo spatial and semantic changes over time. An example of geometric or spatial changes may be the splitting of buildings into two or more buildings or their fusion by constructing a physical extension connecting two buildings [Ren00; Ste+08] (see figure 2.10).

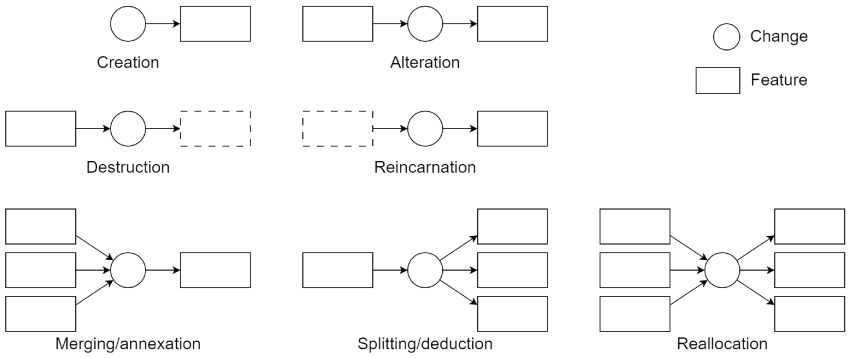

Figure 2.10.: Seven basic types of identity changes proposed in [Ren00].

In addition to spatial and structural changes to geospatial features, semantic changes can also occur over time, e.g., changes to data that describes, categorizes, or contextualizes geospatial features and their relationships to other features. For example, figure 2.11 illustrates several semantic and spatial changes between 3 versions of a 3D city model of a building. Between version 1 and version 2 of the 3D model, building "B1020" changes its *function* attribute from "Office" to "Saddle" (an example of a semantic change). The roof of the building is replaced between *Version 2* and *Version 3* denoting a semantic change of the "roofType" attribute in addition to a change in geometry. These changes are often motivated by external administrative changes, updates to documentation, and changes to the data

standards used to represent the non-spatial attributes of geospatial features among other catalysts [NK20].

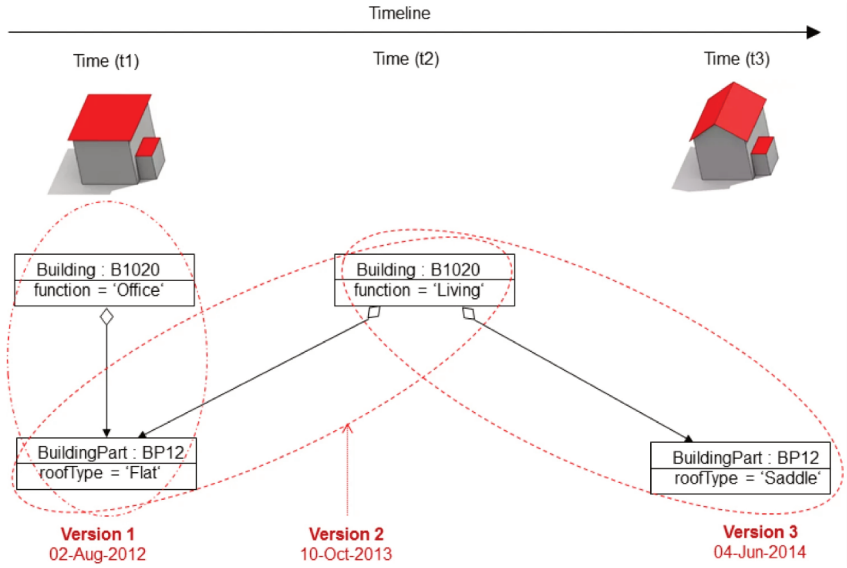

Figure 2.11.: An example of spatial and semantic identity changes from [KCK20].

4D (3D + Time) city models often measure when these changes occur, and for how long, using bi-temporal timestamps that indicate the beginning and end of some temporal *interval* (or a temporal *instant* in the case both timestamps are temporally coincident). Approaches, such as those that follow the INSPIRE model[11], often use two types of temporal intervals or instants for representing the evolution of cities and city objects:

- **Existence time** (or validity time) for representing when urban data and urban objects exist or are relevant in the real-world
- **Transaction time** for storing when the data is recorded, updated, or deleted from a database

Additionally, to be able to compare temporal instants and intervals [All84] proposed a set of temporal relations (figure 2.12). These relations have also been formalized as ontologies in data standards such as OWL-Time. Some works like stRDF (spatio-temporal RDF) [KKK12] provide a notation for defining time intervals with *inclusive* and *exclusive* bi-temporal timestamps. Using an inclusive notation defines interval timestamps as a part of the interval, i.e. two intervals that start and end at the same instant T would 'overlap' one another. Using an exclusive notation defines interval

[11] https://inspire.ec.europa.eu/documents/inspire-generic-conceptual-model

timestamps as outside the interval, i.e. two intervals that start and end at the same instant T would 'meet' one another.

Relation Name		Instant-Instant Relations	Interval-Interval Relations
A disjoint B B disjoint A	A before B B after A		
	A meets B B metBy A		
	A overlaps B B overlappedBy A		
A in B	A finishes B B finishedBy A		
	A during B B contains A		
	A starts B B startedBy A		
	A equals B B equals A		

Figure 2.12.: A synthesis of the temporal relations as defined in OWL-Time according to Allen [All84].

At the city scale, changes are often recorded at different frequencies. Batty [Bat18] notes that in the application of urban digital twins as approaches to model and analyze the evolution of the urban landscape, *high-frequency* cities require updating the digital representation of the city in shorter time intervals at near real-time speed, while *low-frequency* cities may require updates on the time-scales of years and decades (or centuries and millennia in the case of creating city models for archeological and built cultural heritage research domains [Lóp+18; Rod+17]). A lower frequency city may provide a temporally static 'snapshot' of the city every few years as an update.

In order to provide data structures for integrating and storing digital city models produced at different frequencies, research works have proposed graphs and versioning approaches [Ren00; NK21; Ber19; JSG20; CK19]. In particular, the Versioning module proposed in the CityGML 3.0 standard [KCK20] and extension such as the Workspace data model proposed in [SSG20] provide useful concepts for modeling scenarios of urban evolution and for modeling multiple concurrent evolutions of urban data. The term *workspace* originates in Version Management Systems (VMS) such as GIT[12] and Apache Subversion[13] and refers to different versions of data can be used to isolate 'working' copies of data from production-ready versions [Cha+17]. These works use several concepts—some of which also originate from VMS—to model urban evolution and concurrent viewpoints of urban evolution (also illustrated in figure 2.13):

[12] https://git-scm.com/
[13] https://subversion.apache.org/

Feature with a lifespan Also referred to as a “versioned" feature, this is a representation of a real-world phenomenon with a specific existence time.

Version A specialization of a feature with a lifespan used to contain a set of versioned *city object* features that are temporally coincident (these city objects are defined by the other thematic module of CityGML, e.g., buildings, bridges, vegetation, etc. . . .). The version’s existence time is *equal* to the overlap of all the existence times of city object members [SSG20]. This way the version also represents the interval of time when a city model is “stable” and no recorded changes are occurring.

Version Transition A set of **Transactions** or individual changes between the corresponding versioned city objects of two consecutive Versions (e.g., whether a feature is added, modified, or removed from the Version). These transactions are analogous to the previously discussed identity changes. The existence time of a Version Transition occurs between the existence time of its corresponding Versions.

Scenario A string of sequential versions and version transitions can be aggregated into a **Scenario** for representing a specific evolution of the city, hypothetical or otherwise.

Space Scenarios exist within either a **Consensus Space** or **Proposition Space** In consensus space, a single scenario is declared that contains all the versions and transitions that represent an agreed upon real-world evolution of a city (or district). Proposition Space contains all other scenarios including hypothetical or imagined evolutions of the city. This includes scenarios that could belong in consensus space but are currently disputed, unproven, or undocumented.

Workspace A composition of a single proposition and consensus space that can be used to store the relevant representations of city evolution for a specific application or community of users.

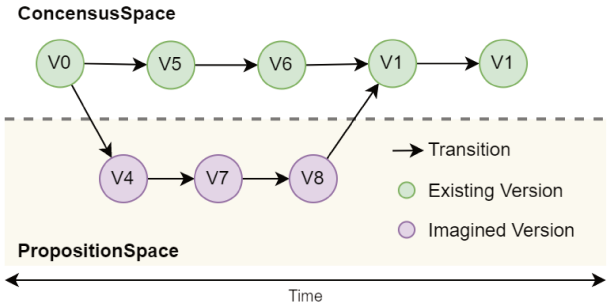

Figure 2.13.: Illustration of a Workspace composed of 2 spaces (a consensus space containing a real-world scenario of urban evolution, and a proposition space containing 0 or more hypothetical scenarios of evolution). These scenarios are themselves composed of Versions or “snapshots" of an urban area and Transitions that contain the changes between each successive Version [SSG20].

2.4 Conclusion

As discussed in this chapter, there are a wide variety of proposed approaches for constructing urban data models with spatial, temporal, and semantic components. These data models are essential for standardizing organizations and research works to define and share the structure and semantics of data being proposed. However, the wide variety of existing urban data standards have been proposed and resolving the interoperability gaps between these standards may be required for applications that rely on heterogeneous urban data sources (problem **P1** defined in section 1.1).

For this purpose, this thesis proposes using *ontological languages* and *knowledge graph formats*. As previously discussed, they sufficiently fulfill these needs by providing a basis for exchanging data across urban information domains and providing rich data structures for storing and combining heterogeneous 3D semantic data [Mal+20; Cla20]. In addition, existing geospatial data standards have been proposed with ontological models, such as the GeoSPARQL ontology and the General Feature (Meta)Model, thus providing well-known vocabularies that can be reused to improve the interoperability of the data transformed by the proposed approach.

To accommodate these needs, chapter 3 will discuss existing approaches in nD urban data integration with regards to the languages and standards discussed in this chapter.

3 N-dimensional urban data integration

As introduced in chapter 1, the urban landscape is a complex environment that encompasses many domains of information, including many areas of built infrastructure such as transportation systems, buildings, bridges, etc. Chapter 2 discusses how data models can be used to define the structure and semantics of nD urban data as it conforms to urban data standards. However, data integration is still required to overcome interoperability gaps *between* data standards to produce more complete views of the urban landscape and its evolution. This chapter introduces and describes key concepts in geospatial and temporal data materialization approaches to data integration, and provides a literature review of their application in creating unified views of the evolution of the urban environment. The chapter is structured as follows: section 3.1 introduces several areas of application of nD urban data integration and the characteristics of the data being integrated; section 3.2 discusses model-driven and linking approaches to data integration, their limitations, and how these approaches are generally applied to resolve heterogeneous data integration problems; section 3.3 introduces data and data model validation techniques and metrics for measuring the quality of formal models; finally, section 3.4 briefly concludes the chapter.

3.1 Urban data integration applications

Data-driven approaches such as the construction of virtual environments such as urban digital twins and smart city applications [Bat18; Jul+18; SH20] can help visualize and simulate urban events and dynamics, both real and imagined, over time [Bil+15; CK19; JSG20; SSG20].

In this domain, one of the biggest areas of research is the integration of Building Information Models (BIM) data, Geographic Information Systems (GIS), and City Information Models (CIM) data, otherwise referred to as BIM-CIM or BIM-GIS integration [Cla20; HRB20; UJS21]. In BIM, the word building refers to not just buildings themselves but the process of building construction including infrastructure

and landscape projects [HRB20]. BIM also includes the design, tools, methods and processes implemented for representing the built environment. IFC[1] (Industry Foundation Classes) is an ISO standard proposed by BuildingSmart that contains a conceptual data model for modeling BIM information. The IFC conceptual model is formalized using EXPRESS and IFC data is stored in the STEP (Standard for the Exchange of Product model data) format [HRB20].

In the urban context, BIM information is catered to model micro-level city information at smaller scales such as construction sites and building interiors [WPL19]. Conversely, GIS and CIM information focuses on macro-level information on the scale of districts and metropolitan agglomerations and take into consideration the position of these phenomena on the Earth's surface [WPL19]. Here, GIS refers to digital systems for collecting, manipulating, visualizing, and storing cartographic, geographic, and remote sensing data [Mag91]. Because of this both GIS and CIM standards are complementary with BIM standards, making them a useful target for data integration.

Amongst GIS and CIM standards, the CityGML standard proposed by the OGC is often used to facilitate the storage and exchange of 3D geospatial semantic city models. The standard is divided into several thematic modules for representing different types of urban information including buildings, transportation networks, and urban vegetation. In this standard, all city objects can be represented in up to five different, well-defined levels of detail (LoD0 to LoD4) with increasing accuracy and structural complexity (e.g., a building is composed of walls, roofs, etc.) [Kol09] (figure 3.1). It defines the three-dimensional geometry, topology, semantics, and appearance of the most relevant topographic objects in the urban context [GP12].

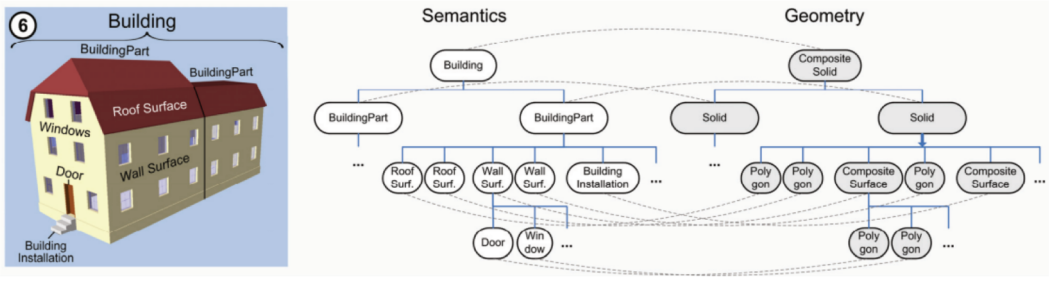

Figure 3.1.: An example of the thematic and geometric classes from CityGML and their relations from Stadler and Kolbe [SK07].

In its most recent version CityGML 3.0 has included BIM-like concepts such as construction classes and logical spaces to improve semantic interoperability with

[1] https://www.buildingsmart.org/standards/bsi-standards/industry-foundation-classes/

IFC [KCK20]. In addition, it has introduced new modules for representing the complex data requirements of 4D (3D + time) semantic city models [CK19]. The OGC has also provided a readily available abstract data model for CityGML 3.0 formalized with UML, referred to as the CityGML Conceptual Model based on the General Feature Model (as introduced in section 2.1.2) [KCK20].

As categorized in Hbeich et al. [HRB20] and Liu et al. [Liu+17] BIM-GIS integration is applied in several urban use-cases for improving city administration, urban planning and construction, and emergency response services. These applications include, the enhancement of *indoor/outdoor navigation* and location-based services, the development of *3D cadastres*, and the development of *urban environment analysis* and *energy management* tools [HRB20]. These applications are impacted by a wide variety of stakeholders including government, construction, and architectural agencies, urban data producers, urban planners, developers, and citizens [NK21].

Another large area of research that often requires nD urban data integration is the domain of Heritage (or historic) Building Information Modeling [Col+20; Lóp+18] (H-BIM) for understanding built cultural heritage (BCH) [NCM19; Col+20; ZG20]. In this domain, cultural heritage institutions and governing bodies use digital data repositories to store, disseminate, and study culturally significant built constructions from the past such as historical monuments and archeological sites [NCM19]. These repositories are especially important for the digital conservation of damaged or at-risk built cultural heritage constructions [Col+20]. Like in the urban digital twin domain, H-BIM and GIS-based BCH approaches may propose BIM-GIS integration to provide more complete views of built cultural heritage objects [Col+20; Lóp+18]. In addition, standards like CIDOC-CRM (International Committee for Documentation-Conceptual Reference Model) and CRMgeo are sometimes integrated to supplement BIM and/or GIS standards with a common vocabulary for modeling non-building related information (e.g., administrative or historical information) [Mes+18; ZG20; NCM19].

With data capture methods such as using laser scanning technologies, aerial photography, and photogrammetry, 3D digital models can be created at different scales for these purposes [Lóp+18; Rod+17] (figure 3.2). Rodríguez-Gonzálvez et al. [Rod+17] notes that time-dependent 3D (or 4D) analysis of cultural heritage information solely based on this type of survey data is most commonly relevant on timescales from the late 19th century until today. In addition, 4D analysis on these and earlier timescales often requires the integration of historical documents such as antique or ancient maps, drawings, paintings, written texts, etc. ... [Rod+17; Sam+16; Jai20; Mün+20]

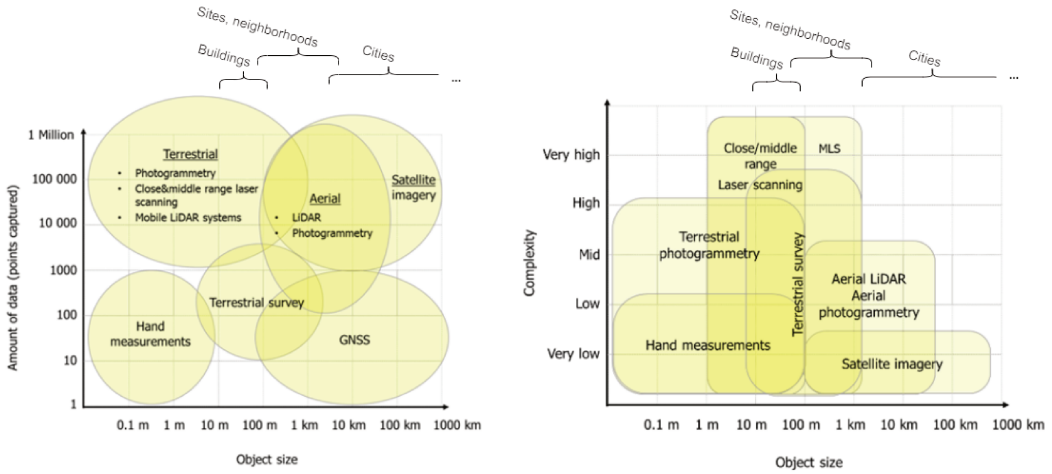

Figure 3.2.: A comparison of object scale in 3D data capture techniques by data size (left) and data complexity (right) from Rodríguez-Gonzálvez et al. [Rod+17]. Techniques include Airborne Laser Scanning (ALS), Terrestrial Laser Scanning (TLS), Mobile LiDAR Systems (MLS), Global Navigation Satellite Systems (GNSS), and 3D photogrammetry (both aerial and terrestrial). Annotations for approximate building, construction site, neighborhood, and city scales have been added.

Despite the fact that data-driven applications for understanding the evolving urban landscape have seen much advancement, there are still limitations, challenges, and information gaps for integrating the nD urban data often required by these applications as discussed in the following section.

3.2 Data integration approaches

Many approaches have been proposed by industry and academia to address the challenges posed by open urban data integration. Tran et al. [Tra+20] denotes two general categories of integration of data integration approaches: data **materialization** and data **mediation**. Data materialization facilitates multisource data integration by effectuating one or more pre-processing steps on data instances (i.e., data transformation or data cleaning) before storing them in interoperable locations where they can be utilized in digital applications that otherwise could not interact with the data [Tra+20].

Conversely, data mediation integration approaches leave data instances in their original heterogeneous sources and implement on-demand query rewriting or "on-the-fly" data processing (sometimes referred to as virtual data integration) [Tra+20].

Geospatial data mediation approaches have proposed technologies such as OBDA (Ontology Based Data Access) as a scalable method for this type of integration [Cha+21; Tra+20; Kyz+18; Pat+14]. In OBDA, the schemas of data sources are mapped to a virtual knowledge graph representing the integrated data model. These mappings are used to rewrite queries performed on the knowledge graph to extract, aggregate, convert, then integrate data from each relevant data source [Xia+18]. However, the expressivity of the ontology language used to define the virtual knowledge graph may be limited in OBDA by the type of data source. For example, an OBDA mapping between a relational database and an ontology requires the expressivity of the ontology to fall under a subset of description logic known as the *DL-Lite* family of languages [Xia+18; KRZ13]. The OWL 2 sublanguage, OWL 2 QL, is a restriction of OWL based on DL-Lite that can enable mappings between relational database tables and ontologies [Gra+08].

Additionally, Beck et al. [BBK20] categorizes urban data integration approaches—for data instances and data models–in the context of integration between the Building Information Modeling (BIM) and Geographic Information Systems (GIS) information domains into 4 types (figure 3.3):

Conversion This refers to the *transformation* or *translation* of a source information model into a target information model. This type of integration approach is discussed further in detail in section 3.2.1.

Extension This occurs when a model is supplemented with new elements from different models typically to limit data loss during conversion between these models.

(Inter)Linking Linking refers to the integration of models and data through the creation of explicit links or relations. Graph-based integration approaches such as those offered by Semantic Web technologies like RDF fall into this category. In the integration of ontological models, matching and alignment techniques can be applied to interlink data models and data [ES13]. This type of integration approach is discussed further in detail in section 3.2.2.

Merging This refers to when the concepts from two or more models are combined to create a new *shared* model.

The following sections will discuss how these approaches are applied in geospatial and urban data integration at levels M_1 and M_0 of the OMG metamodeling framework.

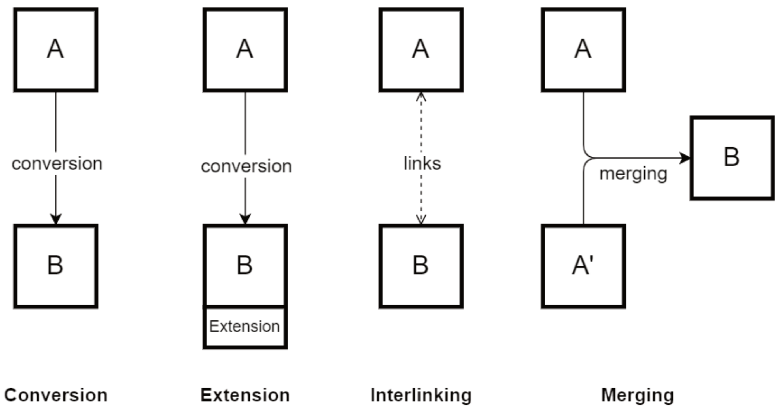

Figure 3.3.: Conversion, extension, interlinking, and merging examples of A (and A') towards model B from [BBK20].

3.2.1 Model-driven transformation approaches

One of the most widely implemented approaches to transform data models between different modeling languages is MDA (Model-Driven Architecture) proposed by the OMG. MDA and similar approaches such as Model-Driven Engineering (MDE) and Model-Driven (Software) Development (MDE, MDSE) propose using automated *model-driven transformations* to translate models between different modeling and description languages. MDA in particular was developed by the OMG to use MOF based metamodels within the OMG metamodeling framework [Kut16]. Model-driven approaches allow users to abstract software development and data modeling problems using abstract data models [SK03], and then derive interoperable unabstracted data schema from these models using model-driven transformations [Kut16]. These transformations are particularly useful in data specification in the event formal data schema are not proposed with research works [BKN18].

In MDA, three types of models are considered for transformation (figure 3.4) [Ope22; Kut16]. Highly abstracted models of systems are called *Platform Independent Models* (PIM). These models should not take into consideration technical aspects of what is being modeled (e.g., the data format, programming language, or hardware architecture where an information system is stored on or executed with is not specified by the model). Conceptual data models typically qualify as PIMs they specify domain-specific concepts and relations in a data format agnostic manner.

Data models that do take these technical aspects into consideration are called *Platform Specific Models* (PSM). For example, where a PIM may define a primitive data type as a *string*, a PSM could specify it as a `VARCHAR` in the context of an SQL

database environment [KWB03]. Although PSM models may define a system or data in an implementation ready way, they are still abstract models formalized in languages such as UML much like logical data models.

Finally, *Platform Models* (PM) are equivalent to machine-readable code or data schema [Kut16] (i.e., physical data models). In the context of data models, PIMs and PSMs can be formalized in more abstract modeling or description languages such as UML, UML profiles, or ER models. However, PMs must be formalized using machine-readable concrete syntax or data formats (such as XML Schema, JSON Schema, etc.). In MDA, an example of a PM for a relational database is a Data Definition Language (DDL) schema or script formalized in SQL that can be executed to create database tables [Kut16]. As OWL ontologies are machine-readable models based on the RDF modeling paradigm they may be considered PMs from an MDA perspective. Within the OMG's four-layer metamodeling architecture, typically PIMs, PSMs, and PM fall into the M_1 layer as "models", and their data instances the M_0 layer. A summary of this comparison of data models and MDA models at different levels of abstraction is proposed in table 3.1. This comparison uses the Entity-Association modeling language proposed by the MERISE framework—a modeling framework from the late 1970s for refining abstract data models and data processes to physical/operational models based on the ER modeling paradigm [Ser10].

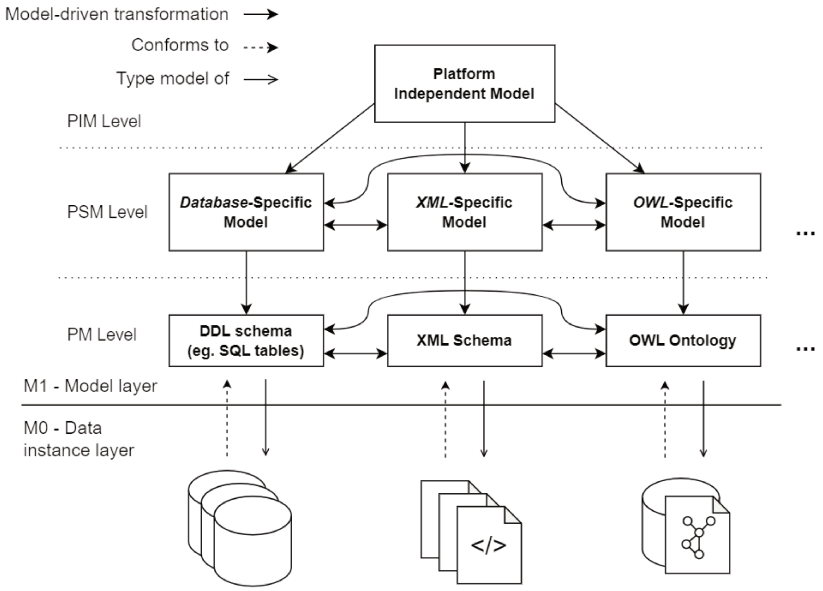

Figure 3.4.: A generalized example of model-driven transformation workflows from [Kut16] contextualized within the OMG metamodeling layers.

Table 3.1.: A comparison between MERISE/ER models and MDA models by level of abstraction.

Level of abstraction	MERISE/ER models	MDA models
High	Conceptual Data Model	Platform Independent Model
Medium	Logical Data Model	Platform Specific Model
Low	Physical Data Model	Platform Model

To effectuate these transformations, a **transformation definition** is declared between a source and a target data model or data (transfer) format, and executed by a transformation engine or tool [CH06], as shown in figure 3.5. A transformation definition is typically composed of transformation rules or transformation mappings between concepts in the source languages or patterns in the source data model towards the target language and data model [CH06]. These transformations have several qualities or features as defined by [MV06; CH06; Kut16]. For example, transformations can be effectuated *vertically* or *horizontally*. Vertical transformations are used in model refinement (e.g., a transformation workflow of PIM → PSM → PM) and in reverse engineering (e.g., a transforming workflow of PM → PSM → PIM) [Kut16; CA17]. Horizontal transformations can be used to refactor a model without changing language or to migrate a model to different languages or technical requirements. Similar to vertical and horizontal transformations, transformations to models within the same language are categorized as *endogenous* while, transformations between different languages are referred to as *exogenous*. Transformations that can execute round-trip transformations i.e., from source to target and vice-versa, are defined as *bidirectional* as opposed to *unidirectional* transformations.

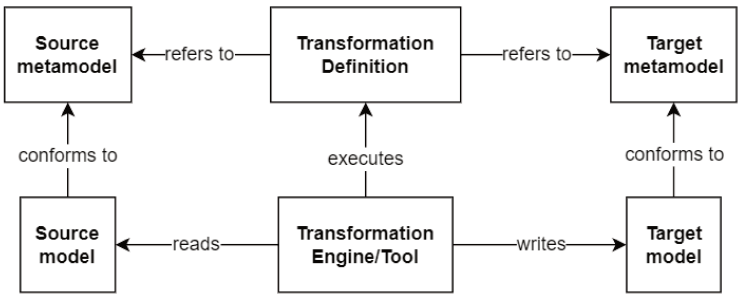

Figure 3.5.: Model-driven transformation overview from [CH06].

Concerning exogenous model-driven transformations towards ontological model targets such as OWL, it is important to consider the language with which the input data model is formalized in, as the same data model formalized in different languages can produce OWL ontologies with varying structure and vocabulary. Brink et al. [Bri+14] notes that in the case of geospatial models like GML, transformations from less abstract PM languages (such as XSD) may provide better definitions of spatial semantics as these models directly define the structure of the geospatial data they define. Alternatively, more abstract languages such as UML may have similar concepts to ontological languages like OWL and thus may have more well-defined transformation mappings like those proposed in the ISO 19150-2 standard which proposes several transformation mappings from UML to OWL [Bri+14].

UML to OWL transformations

UML to OWL transformations have seen much development in the past 20 years with even more advancements in the geospatial domain within the past 10 years. From an MDA perspective since OWL ontologies are based on the Semantic Web technology stack, they are often considered PM models. With this lens, *vertical* model-driven transformations can be executed from either OWL-specific PSMs (formalized in UML) or PIMs towards OWL ontologies.

Concerning OWL-specific PSM to OWL PM transformations, most approaches rely on either extending the UML languages with RDF(S) and OWL specific concepts through UML profiles [Gas+04] or using tags and annotations [De +17; JOH19] to define how UML concepts should be transformed to OWL. Other approaches like Chowlk [CGP22] propose UML-based graphical notations to formalize OWL ontologies in UML diagrams.

In the geospatial domain, use of the ISO 19150-2 standard for geographic information *Ontology—Part 2: Rules for developing ontologies in the Web Ontology Language* has been proposed for transforming UML models to OWL ontologies. These mappings are generic enough to transform PIMs to PMs in OWL without the use of additional annotations, UML profiles, or graphical syntaxes not native to UML. In addition, the ISO 19150-2 standard enables the transformation of concepts from the GFM metamodel (section 2.1.2) for defining non-domain specific geospatial information—an approach proposed by standardizing organizations to provide a shared geospatial and geometric vocabularies between data standards. Appendix B.1 lists a summary of proposed conversion rules between UML and OWL as noted in Jetlund et al. [JOH19].

Although, an ISO 19150-2 based approach can theoretically be applied to any existing UML model, several limitations and challenges to implementing these transformations have been identified. First, the proposed mappings create a very constrained OWL ontology that should be interpreted under the CWA assumption (section 2.2) since any source UML model it was created from is constructed under the CWA [Cox13; Bri+14]. Additionally, ISO 19150-2 is insufficiently specific regarding certain transformation mappings, for example, the `Union` classifier from the GML or `AssociationClasses` UML, both of which do not have a clear equivalent in OWL [JOH19; Ope17]. Several alternatives to these limitations have been proposed in [JOH19; Ope17; INS17], however it is still unclear when some approaches are better than others.

In addition, some UML models that have not been defined with ontological modeling concepts in mind may be transformed into "awkward" OWL ontologies [Bri+14; De +17]. For example, multiple inheritance is common practice in OWL while it is often avoided in UML [Bri+14]. [Bri+14] also notes that reusing vocabularies is a key element of improving interoperability in linked data on the web and which ISO 19150-2 does not take into consideration. Model-driven transformation tools like ShapeChange[2] support custom transformation mappings in addition to ISO 19150-2 transformations which facilitate the integration of existing linked data vocabularies. However, concerning geospatial data standards several existing options exist for mapping to vocabularies such as the ISO 19107, GeoSPARQL, and BOT (Building Topology Ontology) ontologies and extensions to these ontologies [JOH19; Ras+20]. Also, UML attributes are locally scoped to their classes, while properties in OWL are globally scoped [JOH19], i.e., in UML two attributes of the same name, but members of different classes are considered unique, whereas in OWL, they would be considered the same entity.

Jetlund et al. [JOH19] also highlights several remaining challenges in creating geospatial OWL ontologies for bidirectional information exchange between geospatial and Semantic Web applications.

Abstract classes Abstract classes are not native to OWL and ISO 19150-2 proposes denoting these classes using the `isAbstract` annotation property. As this does not prevent the instantiation of these classes, restrictions must be proposed to prevent instantiation of abstract classes in OWL.

Unions This is one of the most complex challenges to resolve as unions are used differently in the GFM and OWL [JOH19]. In ISO 19103, a union is a list of datatypes, where one and only one shall be used for an attribute value.

[2] https://shapechange.net/

ISO 19150-2 proposes transforming these unions to the set-based union class constructor in OWL which may be insufficient in certain applications [JOH19]. This approach can be supplemented with class restrictions to enforce the correct instantiation of union data types, an approach proposed by the Open Geospatial Consortium [Ope17] and utilized in the INSPIRE guidelines [INS17]. Alternatively, unions can be mapped to properties and subproperties [ZL12] or unions of disjoint classes [Cox13]. Unions may also be 'flattened' into a set of properties whose names are a pairwise concatenation of the names of attributes and associations that reference the union and each attribute of the union [Ope17]. Further study is required to understand when these approaches should be applied [Ope17].

Compositions Restrictions are needed to ensure that instances of the parts of a composition are related to the same 'parent' instance. Jetlund et al. [JOH19] suggests a combination of using a `InverseFunctional` restriction, a hierarchy of association types, and the `aggregationType` annotation from ISO 19150-2.

Code Lists OWL representations of CodeLists should be restricted to refer to valid predefined values in the data model and allow using other values. Jetlund et al. [JOH19] suggests using a combination of unions and external SKOS Concept Schemes. Otherwise, CodeLists are transformed into `owl:DataUnionOf` [ZL12].

At metamodeling level M_0, model-driven transformation based on these data models require taking into consideration relevant encoding rules or vertical transformation rules to avoid data loss [Kut16] (figure 3.6) due to the fact that PIM and PSM data models are still abstract representations of the PM models. For example, model-driven data transformations of GML and OWL data could take into consideration the encoding rules proposed by ISO 19136 (for UML to GML encoding) and ISO 19150-2. Taking into consideration these rules is imperative as the transformation definition must be aware of the structure the contents are being transformed into as a data format [Kut16].

XSD/XML to OWL transformations

An alternative to the vertical model-driven transformations previously discussed are horizontal PM to PM transformations, e.g., XSD to OWL. These transformations typically employ XSL (eXtensible Stylesheet Language) as the language of choice for declaring transformations (XSLT) between DTD or XML Schema and OWL ontologies in the RDF/XML format [HBC15]. [Bed+11] proposes a set of 40 transformation

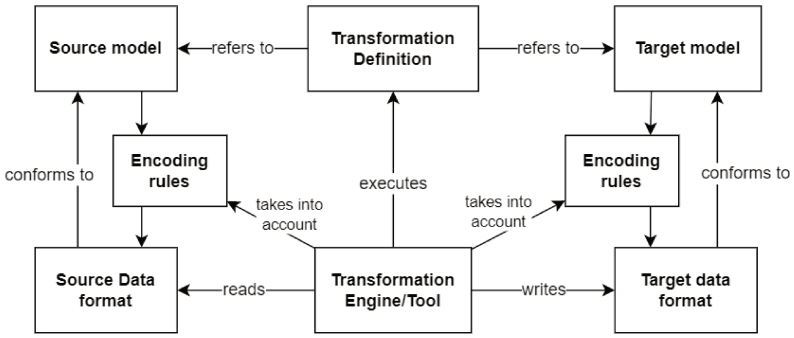

Figure 3.6.: Model-driven transformation at metamodeling level M_0 (simplified from [Kut16]) Here, an encoding rule defines how concepts and data structures in a model are represented in a specific data format.

patterns for translating XML Schema documents to OWL-DL, constituting one of the most comprehensive, generic approaches to this type of transformation [HBC15]. More recently, Hacherouf et al. [HNC19] proposes a formal, mathematical representation of XSD to improve the transformation to OWL.

While many of the proposed XSD to OWL transformation approach propose similar transformation mappings—e.g., a `xs:complexType` is generally always transformed into a `owl:Class`—several discrepancies between proposed transformation mappings exist as well. For example, [BA05; FZT04] propose a mapping transformation between the `xs:choice` construct to a class restriction in OWL that constrains a class to contain a pairwise disjoint union of properties where each property corresponds to a sub-element of the choice construct. More formally, this can be described using the description logic language provided in appendix C.2. Given a concept A and roles X, Y, and Z, where A corresponds to some complex type with a `xs:choice` construct and X, Y, and Z correspond to sub-elements of the choice.

$$A \sqsubseteq (\exists X \sqcup \exists Y \sqcup \exists Z) \sqcap \neg((\exists X \sqcup \exists Y) \sqcap (\exists Y \sqcup \exists Z) \sqcap (\exists X \sqcup \exists Z)) \tag{3.1}$$

Alternatively, Bedini et al. [Bed+11] proposes a mapping transformation of each choice sub-element to a pairwise disjoint union of classes and properties such that each class is restricted to contain one and only one of the properties (where each property corresponds to a sub-element of the original `xs:choice` construct). For example, given concepts A, AX, AY, AZ and roles X, Y, and Z, where A

corresponds to a complex type with a `xs:choice` and (AX and X), (AY and Y), and (AZ and Z) correspond pairwise to choice sub-elements of the choice.

$$\begin{aligned} AX, AY, AZ &\sqsubseteq A \\ AX &\sqsubseteq \exists X \\ AY &\sqsubseteq \exists Y \\ AZ &\sqsubseteq \exists Z \\ AX \sqcup AY \sqcup AZ &\sqsubseteq \bot \end{aligned} \tag{3.2}$$

Both of these approaches are valid interpretations of `xs:choice` with a DL, however, the former requires declaring an exponentially increasing number of axioms (as a combination of `owl:intersectionOf`, `owl:unionOf`, and `owl:complementOf` class restrictions) for each additional choice subproperty. The latter approach also introduces structural heterogeneity (**P1.b**) in the form of intermediate classes to express the same relationship between the complex type and its possible sub-elements[3].

These approaches have applied to urban data standards such as CityGML and IFC to generate OWL ontologies [UJS20]. However, the transformations proposed in [Bed+11; UJS20; HNC19] only take into consideration the transformation of XML Schema and not XML data instances. Approaches also allow for the transformation of data at metamodeling level M_0 (figure 3.7) as well as models at metamodeling level M_1 [FZT04; BA05; CN08; Kra+15]. The methodology in these approaches often takes in an XML Schema, an optional (but conformant to the schema) XML document, and a proposed transformation mapping in the form of an XSLT stylesheet. Sometimes these approaches create a new XSLT stylesheet for transforming XML data that conforms to the schema in a secondary transformation, other approaches may perform the transformation of data instances in parallel to the generation of an OWL ontology [HBC15]. It is worth noting that some of these approaches also propose transformations for generating OWL ontologies from only XML documents (i.e., without a schema) [BA05]. However, approaches that do not take into consideration the schema defining the underlying structure of data risk data loss or risk generating an incomplete ontology [BA05].

In the geospatial domain, spatially aware XML data instance transformations have been proposed—sometimes referred to as spatial ETL (Extract-Transform-Load), a term that originates from data warehouse integration [Kut16; Tra+20]. Transformation tools such as TripleGeo [Pat+14] and GeoTriples [Kyz+18] take into

[3] This is similar to the critique of the intermediate classes generated by the ISO 19150-2 transformation of GFM Unions to OWL highlighted by [Ope17].

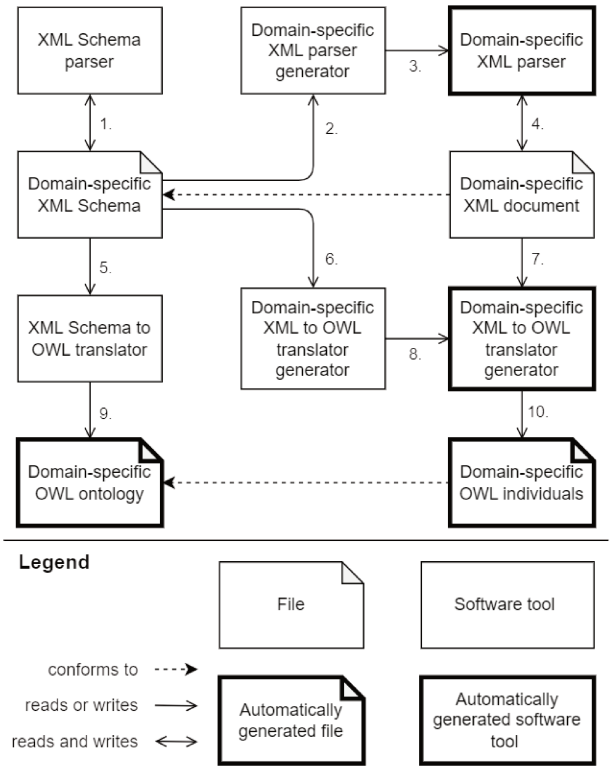

Figure 3.7.: An overview of XML to OWL software tools from Kramer et al. [Kra+15]. Each the arrow corresponds to a read or write activity from a software tool. These activities are numbered in the ordered they are executed.

consideration geospatial data formats (such as GML and GeoJSON) and propose transformations of XML schema and XML data in these formats to OWL. Spatial concepts from these standards are mapped to spatial standards in the Semantic Web such as GeoSPARQL [BK12] and stSPARQL [KKK12].

Another approach is proposed by Brink et al. [Bri+14] for transforming GML data into GeoSPARQL friendly Well-Known-Text (WKT) through XSLT. It relies on the fact that the original GML version 1.0 was developed with an RDF encoding—and despite being serialized in only XML since GML 2—it retained the RDF subject-predicate structure up through GML 3. The assumed triple structure of GML is as follows: each even depth XML element represents an RDF subject; each odd depth element is a predicate that contains either datatype values or references to other RDF subjects. To generate an ontological TBox to accompany the transformed GML data instances (ABox), [Bri+14] proposes using model-driven transformations from UML to OWL. However, as stated in the previous section, implementing both of these transformations requires each transformation to take into consideration the

encoding rules or transformation mappings of the other to ensure that the ABox and TBox of the generated ontologies are conformant and consistent.

3.2.2 Model and data linking

As previously stated, linking is a common practice for reusing and unifying heterogeneous data models and data. Regarding ontological models, *ontology alignment* (sometimes referred to as ontology matching) is a powerful method for integrating data models and data instances with Semantic Web technologies [ES13]. An *alignment* is a set of *correspondences* between two ontologies where a correspondence is a relation between a class, property, or individual from each ontology (figure 3.8). Alignments are typically declared pairwise between a set of ontologies in order to define an *ontology network*. There currently exist several automatic and manual ontology alignment approaches for nD urban data models and data [UJS21; DMF13; Vil+17].

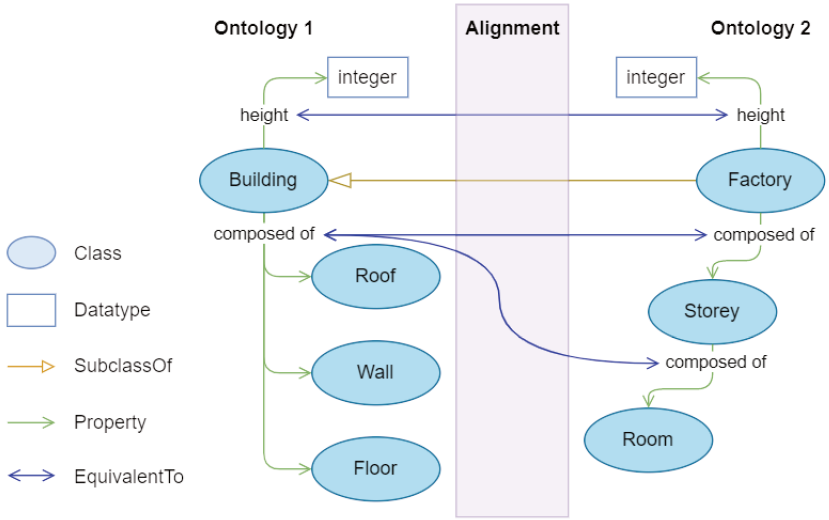

Figure 3.8.: An example of an alignment between a building ontology and a factory ontology. The alignment is composed of a correspondence between the building and factory class and the 'composed of' and 'height' properties.

At metamodeling level M_0, approaches can match different digital representations of the same real-world city object. Approaches may propose applying geospatial *feature matching* or *entity resolution techniques* that compare the spatiosemantic characteristics of city objects, such as a representation's geospatial coordinates (in 2D or 3D) or non-spatial attributes such as textual annotations, administrative

information, identifiers, etc. to determine when two digital representations are modeling the same real-world object [PMG15; SGV06; Bal+22; Ber19].

At metamodeling level M_1, aligned ontologies can be organized differently in order to improve ontology reuse and promote interoperability between different UoD. Wache et al. [Wac+01] identifies three main approaches to this (figure 3.9):

Single ontology This approach is composed of a *global* 'monolithic' ontology for modeling data from different information domains. It is straightforward to implement but does not offer heterogeneous definitions of data. Adding and removing sources based on this model may require adapting the ontology as a whole.

Multiple ontology Several *local* or domain-specific ontologies are implemented and aligned. This may be costly to implement and maintain, but does facilitate heterogeneous definitions of data. Comparisons between local ontologies is possible with this approach, but may be difficult due to the lack of a common vocabulary.

Hybrid ontology This approach proposes implementing a global ontology as a shared vocabulary. This ontology is also referred to as an *upper* ontology. It is less costly to implement than the multiple ontology approach. Adding new heterogeneous data sources only requires creating a new ontological model of the data source and aligning it to the upper ontology. Comparison of heterogeneous ontologies is facilitated by the shared vocabulary.

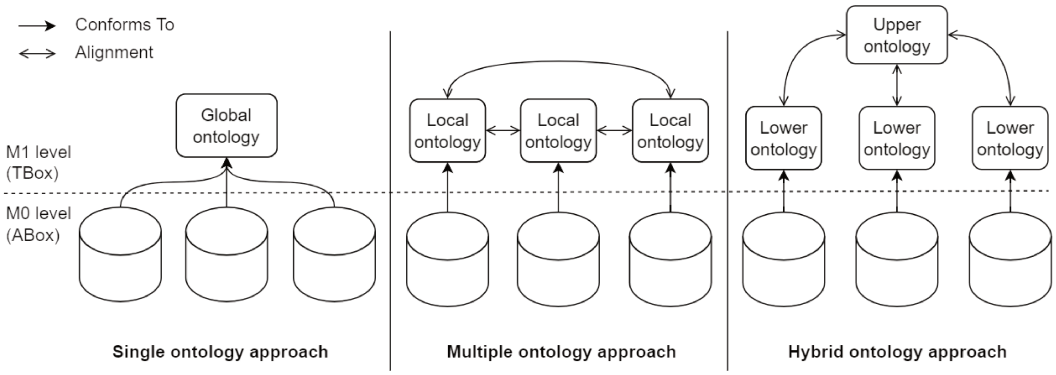

Figure 3.9.: Different approaches to reusing ontologies, adapted from Tran et al. [Tra+16].

3.3 Data validation

Once data models and their conformant data are transformed, data validation approaches are often implemented. These approaches assure that information has not been lost during transformation and may measure the quality of the integrated data models and data are sufficient for an application of integration. There are many validation approaches for structured and semi-structured nD urban data, to verify that transformed data conforms to a specified data model.

For ensuring that different representations of data models are conformant to a data standard, standardizing organizations often propose explicit rules and tests for validating conformance to a data model. For instance, ISO 19105[4] proposes defining standards with *conformance clauses* or classes, where a clause defines the requirements that must be met to be conformant to a standard. Conformance clauses are defined alongside *conformance test cases* for validating that a particular requirement or set of requirements are met. Data standards based on the ISO 191xx series of standards (like CityGML) often implement conformance clauses as a part of an *abstract test suite*.

Validating data quality is typically done by verifying that the data instances resulting after a transformation do not violate the constraints of a data model. The constraints that can be used to specify data models are dependent on the abstract syntax of the language used to define the model. For instance languages such as UML, XSD, and OWL allow for the definition of the cardinality or multiplicity restrictions to bound the number of attributes or properties instances of concepts may have. Data models can be supplemented by constraint and rule languages such as Schematron[5] for XML Schema or SHACL[6] (Shapes Constraint Language) for RDF graphs.

In logical languages, more expressive languages can define more complex constraints as *rules* at the cost of computation efficiency during reasoning (as introduced in section 2.2). Geospatial data applications using Semantic Web technologies can use data model constraints and rules formalized in languages such as OWL and Semantic Web Rule Language (SWRL) to verify that their integrated data is logically consistent (referred to as consistency checking) [Bat+17; Tra+16]. For example, Samuel et al. [SSG20] proposes formal DL rules for validating that versioned 3D city models are logically consistent. This work proposes rules that state *the existence time of a*

[4] https://www.iso.org/standard/76457.html
[5] https://schematron.com/
[6] https://www.w3.org/TR/shacl/

version is implicitly contained within the existence time of every one of the features it contains:

$$\begin{aligned} \forall v \forall vf(featureMember(v, vf) \land hasExistenceTime(v, tmv) \\ \land hasExistenceTime(vf, tmf) \\ \rightarrow in(tmv, tmf)) \end{aligned} \tag{3.3}$$

This type of verification is often applied in applications that propose to automatically and logically infer new information from existing data using reasoners [Tra+20] or spatio-temporal analysis of territorial events [Tra+16; Har15; Ber19]. Inference can be performed by using reasoners such as Pellet, HermiT, or Fact++ [Sir+07; Gli+14; TH06], through queries using the SPARQL[7] query language [Tra+16], or with tools that can parse SWRL rules using SQWRL such as the SWRLTab plugin in the Protégé ontology editor or OwlReady2 [OD10; Mus15; Lam17].

In the context of integrating open data that conforms to international standards, approaches that rely on data model transformations can implement validation steps to ensure that the integrated data is conformant to the original standard and thus internationally interoperable. Hlomani and Stacey [HS14] identifies several validation metrics from the Semantic Web, see table 3.2.

Metric	Definition
Completeness	Measures whether the domain of interest is adequately covered. All questions that the ontology should be able to answer can be answered.
Conciseness	Reflects whether the ontology defines irrelevant elements with respect to the domain to be covered or redundant representations of the semantics.
Consistency	Describes that the ontology does not contain or allow any logical inconsistencies.

Table 3.2.: Several ontological validation metrics from Hlomani and Stacey [HS14].

[7] https://www.w3.org/TR/sparql11-query/

3.4 Conclusion

The concepts and techniques presented in this chapter can be applied to nD urban data integration in various different use-cases and applications. This thesis proposes using model-driven transformations towards ontological languages and graph data formats to facilitate urban data integration. Chapters 4 and 5 will explore the use of more concrete PMs and more abstract PIMs in formalizing a nD urban data models as ontologies through model-driven transformations. The proposed approaches will confront the respective transformation limitations discussed in this chapter. To ensure a high level of data quality and reusability, these approaches implement data model and data validation steps. They will also explore how linking approaches such as alignment can be used to reuse existing geospatial and temporal ontological data standards

Part II

Contributions

4 Towards a semantic web representation from a 3D geospatial urban data model: From CityGML to OWL

As previously stated in chapter 1, the ultimate goal of the 3D semantic urban data integration process is to render data easily exploitable by digital applications to provide the user with unified views of the different representations of the evolution of the urban landscape. To accomplish this goal, model-driven transformations are proposed to facilitate the integration process, where the data models proposed as parts of data standards and research works are used by transformation tools to guide data transformation towards more *interoperable* data formats. In order to respond to the 3D urban data integration problems detailed in section 1.1, this chapter presents the aforementioned contributions from section 1.3:

- **C1.a:** Formalization of 3D urban data models from evolving standards
- **C4.a:** Development of a model-driven data transformation workflows
- **C5.a:** Integration of 3D urban datasets from real-world open data

The work presented in this chapter has been published in a technical report [Vin+20] and in the proceedings of the 16th Spatial Analysis and GEOmatics conference [Vin+21b].

4.1 Methodology

The proposed methodology begins with an identification of relevant data models, formats and standards to be used in the integration process according to the data models, formats and standards studied in chapters 2 and 3. As reusing standards is a key requirement of this integration approach, the target language must provide methods for extending and combining data models. The targeted language must be capable of defining constraints to provide methods for validating when data is conformant to the model. For this purpose, this thesis proposes using *ontological*

languages and *knowledge graph formats*; more specifically, the *OWL 2 DL* sublanguage of OWL and RDF. As discussed in section 2.1.1, OWL is a widely adopted language for defining these ontologies as a W3C recommendation sufficiently fulfills the aforementioned needs by providing a basis for exchanging data across urban information domains and providing rich data structures for storing and combining heterogeneous data sources. Also, as discussed in section 2.2 it provides one of the most expressive, formal abstract syntaxes of the ontological languages while retaining decidability. This will provide a mechanism for validating the conformance of data transformed with our proposed approach to the transformed ontological data model to be discussed in section 4.3.2.

In addition, existing geospatial data standards have been proposed with ontological models, such as the GeoSPARQL ontology, thus providing well-known vocabularies that can be reused to improve the interoperability of the data transformed by our proposed approach (section 2.2). These vocabularies can be integrated through ontology alignment approaches as required by an integration application of this methodology.

To produce the aforementioned contributions, **C1** and **C5**, model-driven XSLT transformations (**C4**) are proposed extending previous works on XSD and XML to RDF, RDFS, and OWL transformation approaches. The proposed data pipelines are described in figure 4.1, which can be broken down into 3 main transformation activities based on the MDA (platform) model concepts (section 3.2.1):

1. Transformation of a platform model (PM) to create one or more urban data models as an ontology
2. Transformation of a PM to generate domain-specific mappings of data instances (conformant to the PM) to ontological data instances (conformant to the generated ontology)
3. Transformation of domain-specific data instances to ontological instances using the transformation mappings generated in the previous transformation activity

During each transformation, several challenges need to be overcome. For instance, the structure and semantics (problems **P1.a** and **P1.b** as defined in section 1.1) of the transformed data instances must conform to the ontology transformed from the original model. In addition, languages like XML Schema Definition (XSD) often use modeling concepts that do not have a direct equivalent in OWL or RDF (**P1.c**). How can mappings be created to overcome these problems and maintain interoperability with the original model? In addition, as the ontological model and data generated by these transformations are declared in a (more) formal, logical language, how

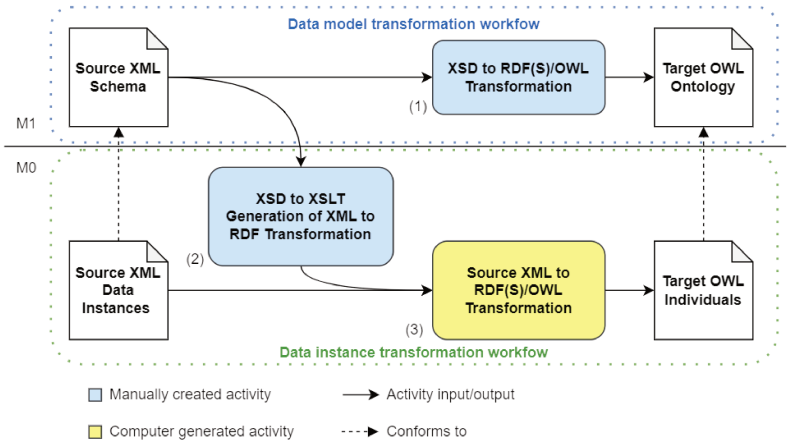

Figure 4.1.: Overview of our proposed XSLT transformation approach. The first activity is used to transform one or more XML Schema to an ontology formalized with the RDF(S)/OWL language. The second and third activities are used to transform XML documents conformant to the aforementioned XML Schema to ontological instances (or individuals) conformant to the generated ontology. The activities, data models, and datasets have been contextualized within the MOF metamodeling levels M_1 (above) and M_0 (below) respectively.

can this information be transformed so that it remains logically consistent to enable reasoning and validation based on constraints? Finally, how can data models from existing data standards in the target language, be reused to provide interoperability with existing data?

Section 4.2 presents the transformation mappings used in the first two transformation activities. Section 4.3 demonstrates and discusses how the proposed transformations are used to create a 3D urban data ontology from the CityGML data standards and real-world CityGML data. Section 4.4 discusses the advantages and disadvantages of the proposed transformations in addition to noteworthy alternatives. Section 4.5 concludes this chapter with the perspectives of the work presented thus far.

4.2 Model-driven XSD to OWL transformations

As an initial approach, we consider that structured (or semi-structured) data must conform to a data model, and that model can be expressed by a PM formalized in XML schema. To transform XML Schema documents to OWL ontologies the transformation mapping patterns provided in table 4.1 are utilized. As introduced in section 2.1.1, the modeling concepts (or abstract syntax) of XSD are composed of 24

"constructs" with `xs:element`, `xs:complexType`, `xs:simpleType`, and `attribute` being among the most commonly used to define the concepts and hierarchical structure of XML data. Notably, a modification of the existing mapping transformation of `xs:choice` XML construct is proposed (section 4.2.1) in addition to several new XML instance transformations (section 4.2.2). In table 4.1, *global* elements are `xs:elements` declared at the root of an XML Schema and *local* elements are descendant `xs:elements` of some complex type.

	Source XSD pattern description	Target OWL pattern description
1.	An `xs:complexType`	An `owl:Class`
2.	An `xs:simpleType`	An `rdfs:Datatype`
3.	A global `xs:element` with a complex type	An `owl:Class` with an `rdfs:subClassOf` relationship to the class generated from the complex type (mapping 1)
4.	A global `xs:element` with a simple type or primitive datatype	An `rdfs:Datatype` with an `owl:equivalentClass` relationship to the datatype generated from the simple type (mapping 2)
5.	A local `xs:element` with a complex type and descendant of a complex type	An `owl:ObjectProperty` with an `rdfs:domain` containing the class generated from the parent type (mapping 1) and an `rdfs:range` containing the class of its own type (mapping 1)
6.	A local `xs:element` with a simple type or primitive datatype and descendant of a complex type	An `owl:DatatypeProperty` with an `rdfs:domain` containing the class generated from the parent type (mapping 1) and an `rdfs:range` containing the datatype of its own type (mapping 2)
7.	An `xs:attribute`	An `owl:DatatypeProperty` with an `rdfs:domain` of the parent type and `rdfs:range` of its own type or an `rdfs:Datatype`
8.	An `xs:sequence` or `xs:all` compositors	An `owl:intersectionOf` class restriction of the properties (mappings 5, 6) generated by the subelements of the compositor

9.	An `xs:group` and `xs:attributeGroup` groups	An `owl:Class`
10.	A `substitutionGroup` attribute	An `rdfs:subClassOf`
11.	An `xs:extension` and `xs:restriction`	An `rdfs:subClassOf` or `owl:subPropertyOf` depending on the type referenced by the base attribute of the extension or restriction
12.	A `minOccurs` attribute	An `owl:minCardinality` class restriction
13.	A `maxOccurs` attribute	An `owl:maxCardinality` class restriction
14.	**An xs:choice**	**An owl:disjointUnionOf class restriction**

Table 4.1.: The chosen XML Schema to OWL transformation mapping patterns (at the MOF metamodeling level M_1) based on [FZT04; BA05; Bed+11; Kra+15]. The mappings proposed in this work are highlighted in bold.

4.2.1 Transforming the xs:choice construct

During the data model transformation workflow, a modification of the `xs:choice` construct is proposed. The `xs:choice` construct is used to declare when an `xs:complexType` XML element may contain one and only one of several descendant `xs:element` constructs. As discussed in section 3.2.1, there are two proposed representations of the `xs:choice` construct. Since the proposal of OWL 2, the approach in Bohring and Auer [BA05] can be improved by using class restrictions with the OWL term, `owl:disjointUnionOf`. This is a more syntactically concise equivalent to declaring a disjoint union of properties with a combination of `owl:intersectionOf`, `owl:unionOf`, and `owl:complementOf` class restrictions as the number of restrictions does not increase exponentially with each new descendant `xs:element` of an `xs:choice`.

An example of this transformation during the data model transformation workflow is shown in figure 4.2. Here, the `xs:complexType`, `"ExternalObjectReferenceType"` contains an `xs:choice` with two descendant `xs:element` declarations. The complex type is transformed into a class that subsumes a disjoint union of two class restrictions corresponding to each descendant element. Note that the name and type of the

elements become the name of the property and value datatype in each restriction. It also avoids introducing structural heterogeneity (problem **P1.b**) through the introduction of intermediate classes.

```
<xs:complexType
  name="ExternalObject
        ReferenceType">
  <xs:choice>
    <xs:element
      name="name"
      type="xs:string"/>
    <xs:element
      name="uri"
      type="xs:anyURI"/>
  </xs:choice>
</xs:complexType>
```

```
<owl:Class
  rdf:about="http://www.opengis.net/
    gml#ExternalObjectReferenceType">
  <owl:disjointUnionOf
    rdf:parseType="Collection">
    <owl:Restriction>
      <owl:onProperty
        rdf:resource="http://www.
          opengis.net/gml#name"/>
      <owl:someValuesFrom
        rdf:resource="xs:string"/>
    </owl:Restriction>
    <owl:Restriction>
      <owl:onProperty
        rdf:resource="http://www.
          opengis.net/gml#uri"/>
      <owl:someValuesFrom
        rdf:resource="xs:anyURI"/>
    </owl:Restriction>
  </owl:disjointUnionOf>
</owl:Class>
```

Figure 4.2.: An example xs:complexType with a xs:choice construct from the core module of the GML 3.1 application schema (left) and the result of transforming the aforementioned XML constructs to a disjoint union of property restrictions with our proposed approach (right).

4.2.2 Data instance transformation

As illustrated in figure 4.1, the transformation mappings for data instances from XML to OWL are generated automatically through their own XML to XSLT transformation. This initial transformation implements the general mappings proposed in table 4.2 to generate the proposed stylesheet. The proposed transformation mappings at metamodeling level M_0 must mirror the mappings proposed at metamodeling level M_1 [BA05]. Thus, these proposed transformation mapping patterns are based off of the aforementioned mappings in table 4.1 to produce an XSL stylesheet, that in turn, can transform XML data instances to RDF.

XML Schemas use the `xs:element` construct to declare what XML elements can be instantiated in a conformant XML document and what type this element is. The `xs:element` construct is declared either globally, as a child of the root of an XML Schema, or locally, as a descendant of `xs:complexType` constructs. As `xs:complexType` constructs are transformed into the `owl:Class` concept (table 4.1,

	Target XSD pattern description	Target XSL pattern description
1.	An `xs:element` based on an `xs:complexType` or `substitutionGroup` of an `xs:complexType`	An `xsl:template` for generating an `owl:NamedIndividuals` with an `rdf:type` named after the `xs:element`. The template also calls the template for its corresponding `xs:complexType` or `substitutionGroup` (mapping 2)
2.	A Local `xs:element` that is a descendant of an `xs:complexType` or `xs:group`	An `xsl:template` for generating either an `owl:ObjectProperty` or `owl:DatatypeProperty` based on if the type of the `xs:element` is complex or simple respectively
3.	An `xs:complexType`, `xs:group`, or `xs:attributeGroup`	An `xsl:template` that calls relevant templates for every possible property or text the element could have (mappings 4-6)
4.	An `xs:attribute` of an `xs:complexType` or `xs:attributeGroup`	An `xsl:template` for generating an `owl:DatatypeProperty`
5.	An `xs:simpleType` with an `xs:restriction` or `xs:extension` of a native XML datatype	An `xsl:template` that selects the inner text of an element (to be used as the value of some `rdfs:Datatype`)
6.	An `xs:simpleType` with an `xs:restriction` or `xs:extension` of another `xs:simpleType`	An `xsl:template` that calls the relevant template for the `xs:simpleType` it is based on

Table 4.2.: Proposed XSD to a domain-specific XML to RDF(S)/OWL transformation mapping patterns based off of table 4.1.

mapping 1) and classes in OWL can have instances (i.e., individuals), declared elements are transformed into `owl:NamedIndividuals` at metamodeling level M_0 (table 4.2, mapping 1). In the case an `xs:element` is declared locally, an additional mapping to either an `owl:ObjectProperty` or `owl:DatatypeProperty` is generated depending on if the type of the element can have complex content or not (mapping 2 in table 4.2).

For example, line 1 of the left listing in figure 4.3 declares a global element, `_CityObject`, as a complex type named `"AbstractCityObjectType"`. Mapping 1 transforms this global element into the `xsl:template` at line 1 of the right listing of the same figure. This template matches any XML element with the tag

`_CityObject` and transforms it into an OWL individual. The possible properties that this individual may have according to its class are handled by calling the template for its type, `"AbstractCityObjectType"` (line 7 of the right listing).

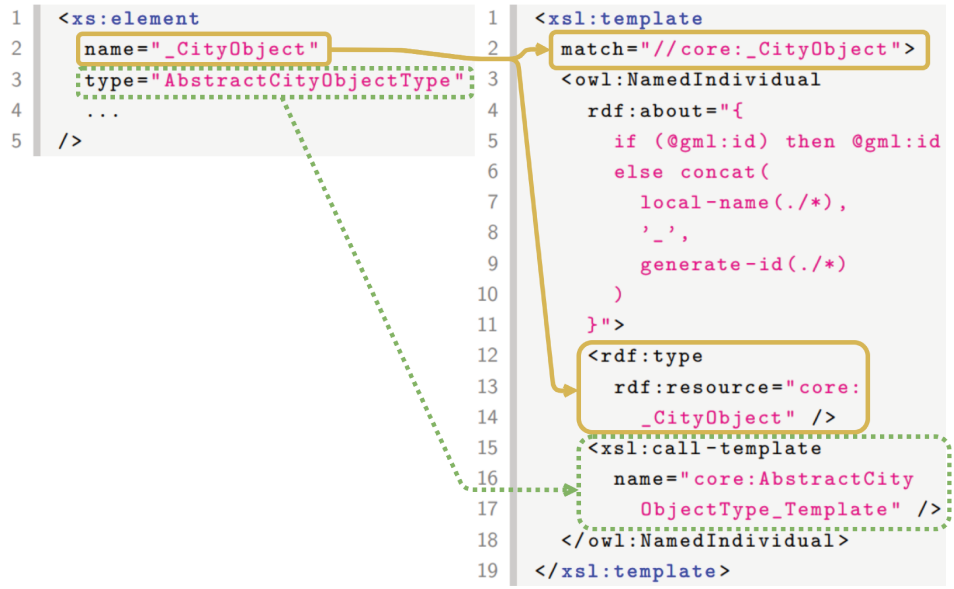

Figure 4.3.: An XML Schema extract from the CityGML standard defining the element *_CityObject* (left) and the generated transformation mappings for *_CityObject* during the data instance transformation workflow for transforming CityGML 2.0 specific XML data to RDF(S)/OWL (right).

Mappings 3–6 are used in conjunction to determine what properties complex and simple types can have and whether these properties reference other individuals (in the case of object properties) or primitive values (in the case of datatype properties). This is required since complex and simple types that extend or restrict other types can be viewed as specializations of those types in XML Schema at metamodeling level M_1. Consequently, when generating the data instance transformation mappings, an `xsl:template` must call all other relevant templates for all the potential referenced types and `substitutionGroups` the class of an individual could inherit from.

This is demonstrated by figure 4.4 where the complex type, `"AbstractCityObject Type"`, is declared at line 2 of the left listing. In this example, the possible descendant elements of the complex type are declared by the type it extends, `"gml:AbstractFeatureType"` (line 7), and its descendant elements, `creationDate` and `externalReference` (lines 10 and 14 respectively). To transform the attributes and descendant elements of an instance of type `"AbstractCityObjectType"` to OWL, the `xsl:templates` named `"AbstractCityObjectType_Template"` is generated in the right listing (line 2). This template calls each relevant template for

each of the possible attributes and child elements (lines 2–18 of the right listing in figure 4.4).

Mapping 2 also transforms the local elements (`creationDate` and `externalReference`) of this example into the `xsl:templates` for the properties these elements correspond to (lines 21 and 28 of the right listing in figure 4.4). The element, `creationDate`, has the type value of the primitive `xs:date` datatype (line 11 of the left listing). This means that `creationDate` should be treated as an `owl:DatatypeProperty` with some `xs:date` value (table 4.1, mapping 6). Thus, the generated `xsl:template` for `creationDate` transforms a matching XML element into a property of the same name and extracts the inner text of the matching element (lines 23–25 of the left listing).

The element `externalReference` has the type value `"ExternalReferenceType"` (line 15 of the left listing), which is a complex type (not shown in figure 4.4). As the element `externalReference` is a complex type and is a descendant of a complex type, it must correspond to some `owl:ObjectProperty` (table 4.1, mapping 5). Thus, the generated `xsl:template` for `externalReference` transforms a matching XML element into a property of the same name similar to the transformation for `creationDate`. However, instead of extracting the inner text, an `rdf:resource` attribute is declared pointing to the generated identifier of its child element.

Additionally, this work proposes how `xs:group` and `xs:attributeGroup` can be transformed at metamodeling level M_0. At metamodeling level M_1, both of these constructs are transformed into `owl:Class` as they help declare the descendant elements and attributes an `xs:complexType` may contain. Also like the `xs:complexType` construct, `xs:group` and `xs:attributeGroup`, are not instantiated at metamodeling level M_0. This behavior is similar to an abstract class in UML. Because of this functionality in XML, both of these constructs are transformed into templates that call the relevant templates of their descendant elements, identical to how `xs:complexType` constructs are transformed at metamodeling level M_0.

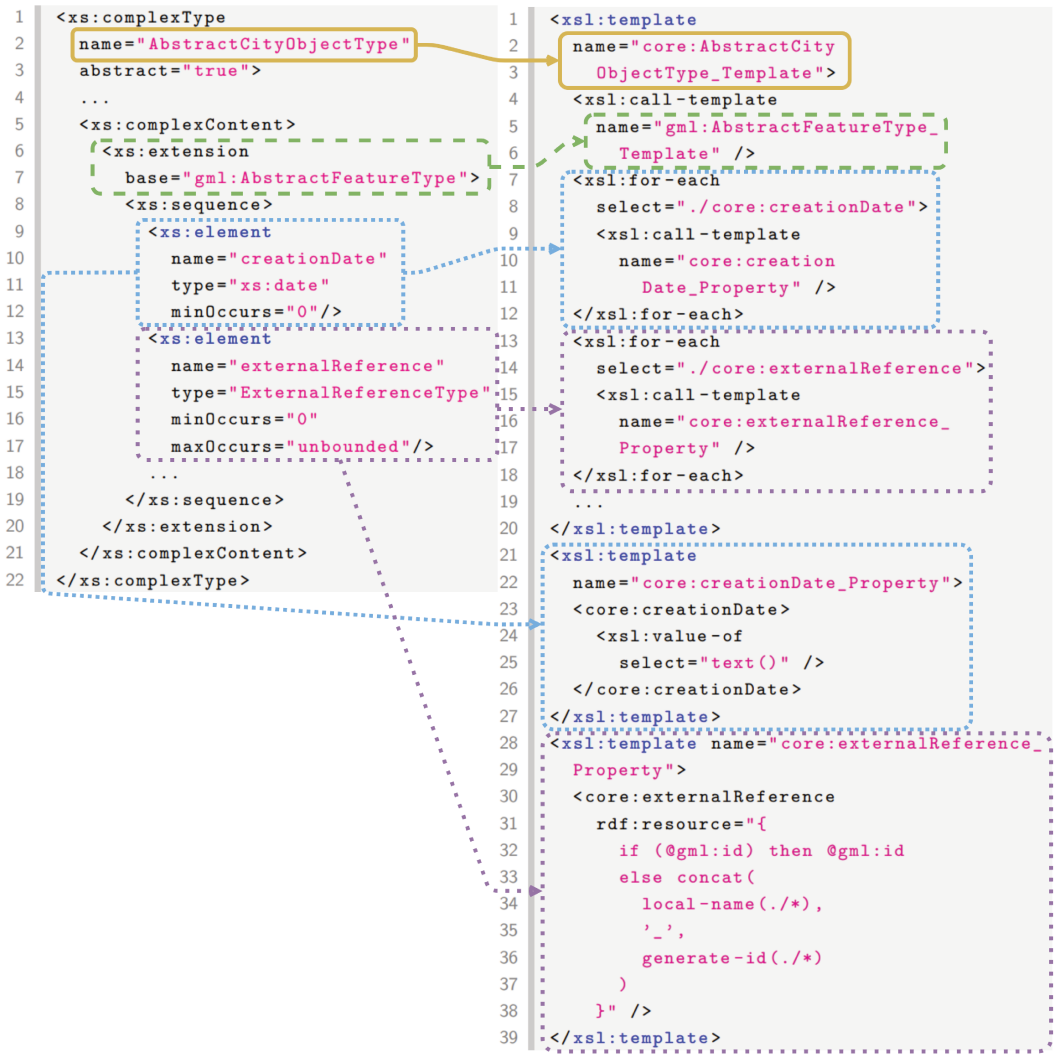

Figure 4.4.: An XML Schema extract from the CityGML standard defining the complex type, *AbstractCityObjectType* (left) and the generated transformation mappings for *AbstractCityObjectType* extracted from the XSLT stylesheet for transforming CityGML 2.0 specific XML data to RDF(S)/OWL (right).

4.2.3 Identifier and namespace transformation strategy

When transforming data models and data, how the identifiers for concepts and data instances are transformed between languages and formats must be taken into consideration. As stated in section 2.1.1, RDF uses a similar system of identifiers as XML. In XML, each element is identified by a 'tag' that is composed of a 'namespace' URL (that may be represented by a prefix) and a 'local name'. In RDF, IRIs (Interna-

tionalized Resource Identifiers) are used to identify RDF entities and are composed of a 'namespace IRI' and an optional 'fragment' (denoted by a # character).

To transform the identifiers at metamodeling level M_1, the names of transformed elements are constructed as a concatenation of the original namespace, and the name of the original, based on the approach proposed in [Bri+14], as follows:

```
rdf:about="[namespace]#[local name]"
```

The results of this transformation are demonstrated in figure 4.2 for the complex type, `ExternalObjectReferenceType` and its sub-elements. An exception to this pattern occurs when a locally declared `xs:element` references a globally declared primitive type `xs:element` of the same name. In this case, these elements would be transformed into an `owl:ObjectProperties` and either an `owl:Class` or `rdfs:Datatype`. When this occurs, the string 'has' is prefixed to the fragment of the generated property to avoid creating a conflicting identifier for the class or datatype, as proposed in [Bed+11; BA05], as follows:

```
rdf:about="[namespace]#has[local name]"
```

At metamodeling level M_0, identifiers are transformed by appending an automatically generated identifier to a predefined domain name and the local name of the element, as follows:

```
rdf:about="[domain name]#[local name]_[generated identifier]"
```

This identifier structure helps ensure the generated identifiers are unique across transformed datasets. Additionally, as GML is a commonly used XML standard for storing and sharing geospatial XML data, the proposed transformations use the `gml:id` identifier attribute to construct IRI fragment similar to what is proposed by Brink et al. [Bri+14]. In this case, the identifier is transformed as follows:

```
rdf:about="[domain name]#[gml:id]"
```

The following section presents experimentations of the proposed transformations and their application to standardized real-world urban datasets.

4.3 Integration process illustrated with CityGML

The transformations defined in the section 4.2 are applied to the CityGML 2.0 and 3.0 XML Schema[1] to produce an initial 3D urban data model from an evolving data standard. In addition, these transformations are used to integrate a real-world CityGML 2.0 open dataset from the metropole of Lyon and a CityGML 3.0 example dataset.

These two transformation workflows are illustrated in figure 4.5. The CityGML model is divided into several thematic modules, i.e., building, bridge, transportation, etc. Because these modules are interdependent and all dependent on the GML standard, the complete model is represented by sets of XML schema documents. 43 different schema for CityGML 2.0 (29 schemas for the GML 3.1 standard and 14 for the CityGML 2.0 standard itself)[2]. CityGML 3.0 adds three new modules (construction, dynamizer, and versioning) and relies on the 31 schemas from GML 3.2 for a total of 48 schemas.

To create holistic transformations, the schema of each version of the CityGML and GML standard are combined into two 'composite' XML schema (one for CityGML 2.0 and another for 3.0) as shown in figure 4.5. This prerequisite step also normalizes the namespaces and prefix declarations across both versions of the standard to ensure consistent, fully qualified IRI namespaces are generated by the transformations. After the composite schemas are created, they are passed to the data model and data instance level transformation activities. Each transformation activity effectuates the transformation mappings declared in a respective XSLT stylesheet.

The more complex data instance level transformation activity is further detailed in figure 4.6. Here, the composite CityGML XML Schemas are passed to an XSLT transformation activity that generates a CityGML-specific XML document to XML/RDF document XSLT stylesheet. This stylesheet is used to transform the XML documents conformant to the CityGML GML encoding standard. After the data instance transformation, a postprocessing activity is run on the transformed data instances to enable geospatial queries on transformed GML geometry by integrating the GeoSPARQL standard. This postprocessing activity is detailed in the following section.

[1] Note that at the time of implementation, the CityGML 3.0 XML Schema were in an unofficial draft state.

[2] The xAL address standard is not considered in this experimentation

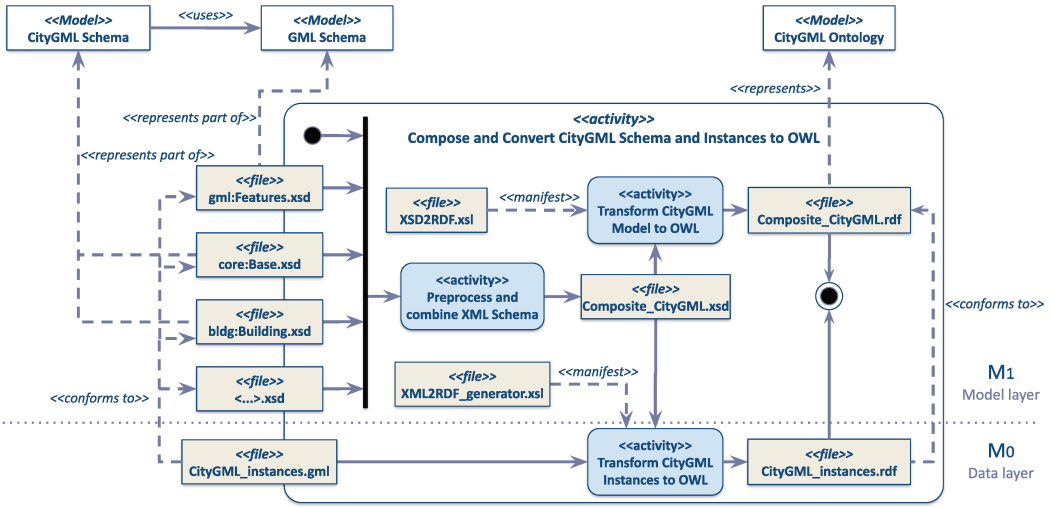

Figure 4.5.: A UML activity diagram of the proposed transformation workflow for transforming XML Schema and XML instances applied to the CityGML standard. The models and file artifacts have been contextualized within the MOF metamodeling levels M_1 (above) and M_0 (below) respectively.

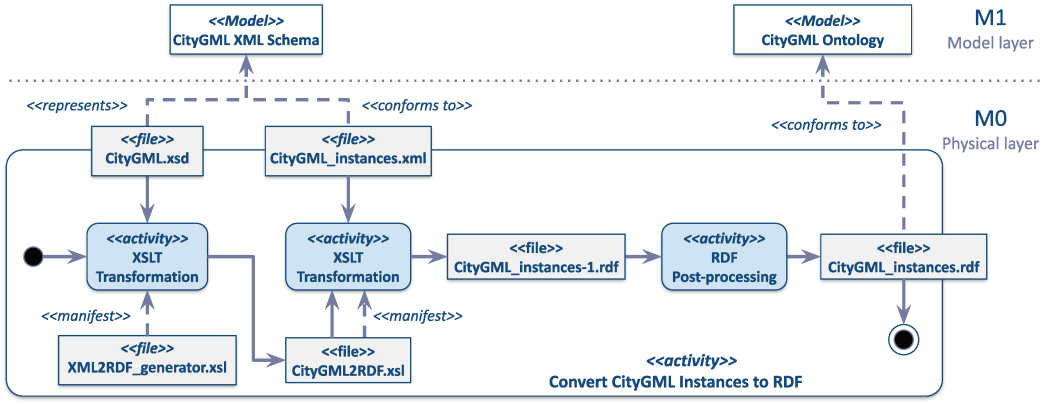

Figure 4.6.: A more detailed UML activity diagram of the proposed transformation workflow applied to transform CityGML XML instances. The models and file artifacts have been contextualized within the MOF metamodeling levels M_1 and M_0 respectively.

4.3.1 Integrating the GeoSPARQL standard

To improve interoperability with existing geospatial linked data, the GeoSPARQL standard is used to facilitate the use of geospatial queries and to provide a standardized vocabulary for geospatial information. To do this, the generated CityGML ontologies must be aligned with the GeoSPARQL ontology. First, a correspondence relation

from the gml:_Feature class of the transformed GML XML schema is declared as an rdfs:subClassOf of geo:Feature from the GeoSPARQL ontology. Since all the CityGML thematic classes inherit (directly and indirectly) from the gml:_Feature superclass, and rdfs:subClassOf declarations are transitive, this implies the thematic classes of the CityGML ontologies are subclasses of geo:Feature. The geometry classes transformed from the GML XML schema can be similarly integrated by declaring the superclass gml:AbstractGeometryType as an rdfs:subClassOf of geo:Geometry.

These two correspondences form the alignment between the generated ontologies and the GeoSPARQL ontology. It also places the thematic classes from CityGML and the geometric classes from GML within the domain and range of the geo:hasGeometry object property, respectively. This permits instances of the CityGML thematic classes to assert when they have a geometric representation as GML geometry.

Additionally, the data instance transformation proposed in section 4.2.2 can be modified to transform the XML trees (that correspond to GML geometry) into a string value of the geo:gmlLiteral datatype. An XML tree represents a valid geo:gmlLiteral datatype if all of its "elements are directly or indirectly in the substitution group of the element {http://www.opengis.net/gml}_Geometry" [PH12] for GML 3.1 and 2.1. For GML 3.2 the substitution group is {http://www.opengis.net/gml/3.2}AbstractGeometry. This transformation is effectuated by modifying the generated XSLT template for the complex type gml:AbstractGeometryType to transform XML elements of this type into geo:gmlLiteral datatypes.

Once, these alignments and transformations are effectuated, the 2D footprints of instances of the thematic classes from CityGML can be geospatially queried. An example query is noted in listing 4.1 and illustrated in figure 4.7.

```
1  PREFIX geo: <http://www.opengis.net/ont/geosparql#>
2  PREFIX geof: <http://www.opengis.net/def/function/geosparql/>
3
4  SELECT ?geometry
5  WHERE {
6    ?geometry geo:asGML ?gml .
7
8    BIND (
9     "<gml:LinearRing ... srsName='EPSG:3946'>
10       <gml:posList>1842640 5175720 1842640 5175750
11         1842660 5175750 1842660 5175720 1842640 5175720
12       </gml:posList>
13     </gml:LinearRing>"^^geo:gmlLiteral
14   as ?boundingBox)
15
```

```
16    FILTER (geof:sfIntersects(?gml, ?boundingBox))
17  }
```

Listing 4.1: An example GeoSPARQL query to select all geometry that intersects a given bounding box.

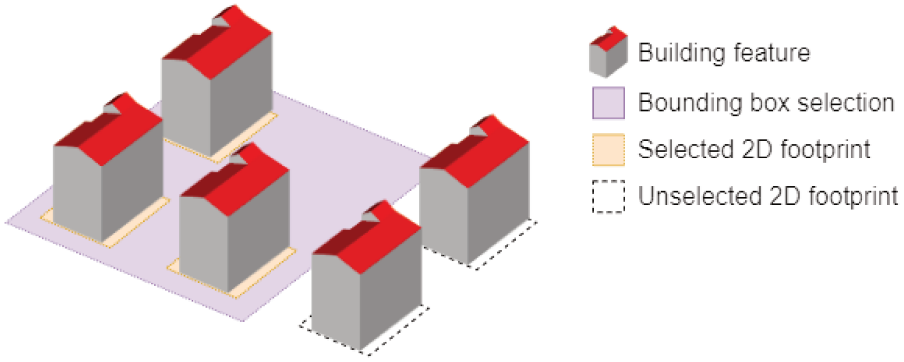

Figure 4.7.: An example of selecting 3D building features through their 2D building footprint with an intersecting bounding box.

4.3.2 Data Model and data instance validation

To verify that transformed data instances (as individuals) are conformant to the generated ontologies, reasoners are used since OWL-DL is a formal (logical) language. As stated in section 2.2, a reasoner can infer when an individual and their asserted properties are not logically consistent with the constraints asserted on their classes.

For example, figure 4.2 shows the transformation result of an `owl:disjointUnionOf` transformed from an `xs:choice` construct as proposed in section 4.2.1. Here, the intended behavior of the `owl:disjointUnionOf` is to restrict the class 'ExternalObjectReferenceType' to have either the property `http://www.opengis.net/gml#name` or `http://www.opengis.net/gml#uri` but not both.

To test this, an individual 'someReference' was instantiated with both of these properties. A reasoner declared the ontology inconsistent, as intended (figure 4.8). Any inconsistencies found in each generated ontology were first manually repaired before using this approach to validate data instance conformance. The results of this activity are presented in the following section.

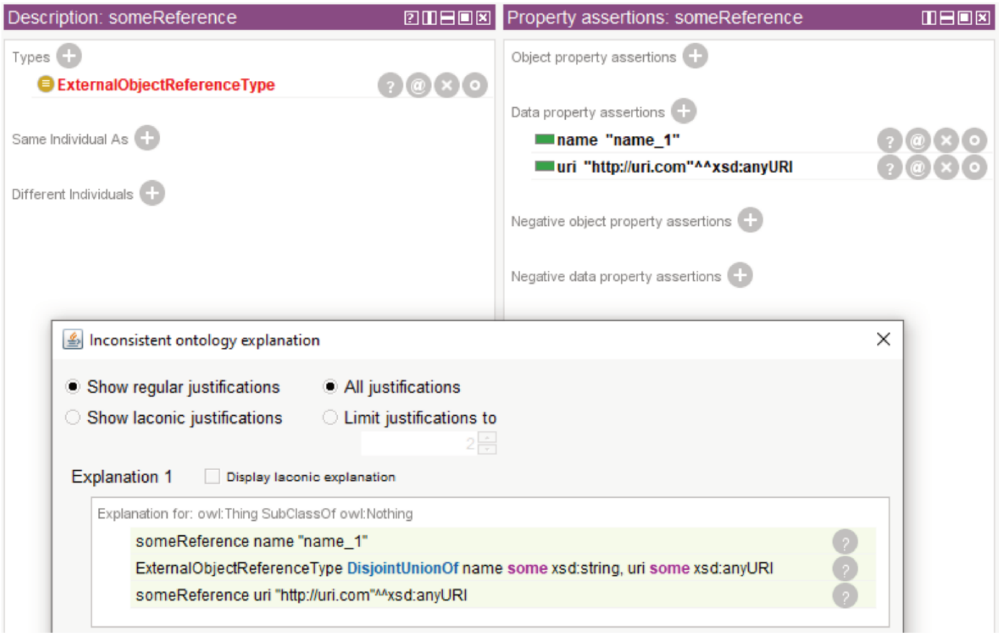

Figure 4.8.: An example implementation of the HermiT reasoner to detect inconsistent assertions of owl:DisjointUnionOf.

4.3.3 Results: 3D urban data model and integrated datasets

Information regarding the final generated CityGML 2.0 and 3.0 ontologies is denoted in table 4.3 and is compared to existing CityGML 2.0 OWL ontologies from Métral and Falquet [MF18] and Chadzynski et al. [Cha+21]. The CityGML ontology from the University of Geneva [MF18] is also created using XSL transformation between XML Schema and OWL. To provide a more concise CityGML ontology [MF18] proposes merging classes from complex type definitions whose names end in `Type` with classes of the same name without the trailing `Type` in their name. Assuming the transformation mappings for `xs:element` and `xs:complexType` to OWL are similar between this work and Métral and Falquet [MF18], this simplification removes roughly half of the generated classes. In addition, the ontology proposed in [MF18] does not include a transformation of the GML schema. Axioms transformed from the GML schema account for over half of the generated ontology.

The ontology proposed in [Cha+21] contains several modifications of the University of Geneva's ontology to make it logically consistent according to OWL DL. For example, multiple tests were effectuated on the ontology such as examinations of the ontology's *consistency* and *conciseness* (section 3.3) [Cha+21]. Errors and inconsistencies found during these tests were corrected. In addition, [Cha+21]

adds axioms to the ontology based on use-case specific needs determined by a *completeness* test.

Ontology	# of Axioms	# of Classes	# of object properties	# of datatype properties
CityGML 2.0 with GML	7517	1246	745	247
CityGML 2.0	3061	389	329	101
[MF18]	1254	185	281	92
[Cha+21] (without imports)	1760	343	263	23
CityGML 3.0 with GML	8443	1647	901	230

Table 4.3.: A comparison of different CityGML ontologies generated from XML Schema. Ontologies generated with the approach proposed in this chapter are highlighted in bold.

To test the proposed data instance transformation workflow a real-world open CityGML 2.0 dataset[3] was transformed from the 1st arrondissement of Lyon, France. The selected datasets represent the buildings of this urban zone in 2015 using a LOD2 GML geometry. The texture data and information from the generic module of CityGML have been removed from the datasets to focus on the transformation of GML data and instances of the CityGML core and building modules. Additionally, an example CityGML 3.0 dataset[4] from the OGC is transformed. This dataset features integrated time series data through CityGML 3.0's Dynamizer module. To support geospatial queries with GeoSPARQL, the 3D geometry of buildings was flattened into 2D using the FME spatial ETL tool[5]. Alternatively, this step can be omitted for applications that only require storing 3D geometry or that can support decoding `geo:gmlLiteral` geometry as proposed in [Bon+19]. Table 4.4 lists information regarding the resulting datasets. A visualization of a building (individuals and their classes) is shown in figure 4.9. The generated ontological CityGML data models are reused in chapter 6 in an nD urban data application demonstration.

[3] https://download.data.grandlyon.com/files/grandlyon/imagerie/2015/maquette/LYON_1ER_2015.zip
[4] https://github.com/opengeospatial/CityGML3.0-GML-Encoding/blob/main/resources/examples/Dynamizer/Building_CityGML3.0_with_Dynamizer_and_SensorConnection_V2.gml
[5] https://fme.safe.com/

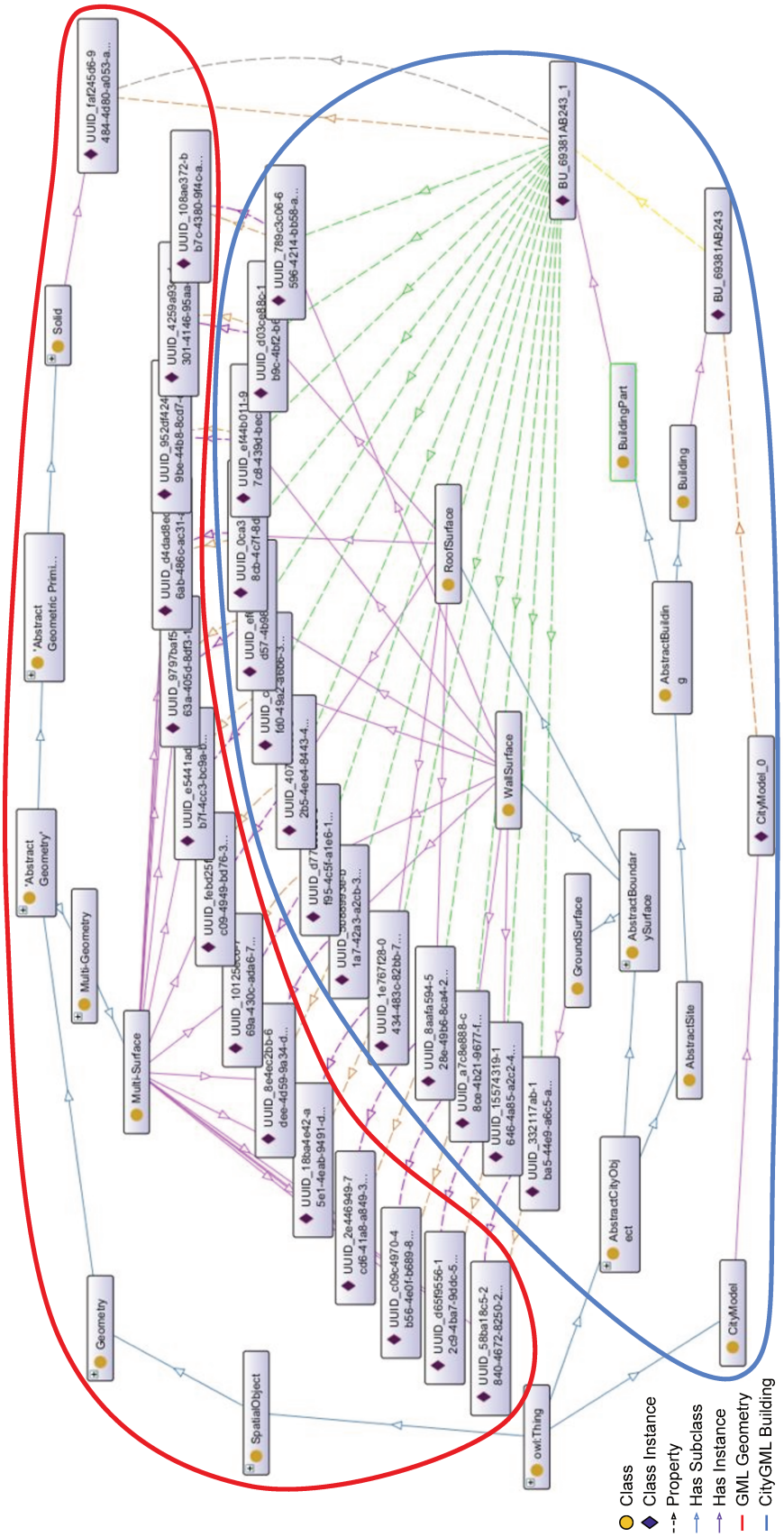

Figure 4.9.: The resulting CityGML building data from the metropole of Lyon as an ontology from [Vin+21b], visualized with OntoGraph. Geometric classes and individuals are circled in red. Thematic classes and individuals are circled in blue.

Dataset name	# of input XML elements	# of Axioms	# of Individuals
LYON_1ER_BATI_2015	524,990	1,844,941	424,563
Building_CityGML3.0_with_Dynamizer_and_SensorConnection_V2	150	453	113

Table 4.4.: A comparison of the transformation results of 1st district of Lyon, France from 2015 and an example CityGML 3.0 dataset with the data instance transformation workflow.

4.4 Discussion

The proposed transformations follow the "garbage in, garbage out" principle that poorly formed data input into a program, will produce nonsensical results, and thus assume that the GML and CityGML instance documents provided are well-structured and conform to their application schema. However, a limitation of this approach is its reliance on the semantic limitations of XML Schema. This is most evident with the transformation of the GML schemas that CityGML is based on. Brink et al. [Bri+14] notes that OWL ontologies generated from the GML XML Schemas may have "well-defined" spatial semantics compared to ones generated from UML models of the GML standard. While this section does not quantify what is meant by "well-defined" in this context, it is clear that the generated GML ontology is incredibly detailed and intricate compared to the official GML ontology from the GeoSPARQL standard. This level of detail may not be necessary for every geospatial application or use case. Certain applications may consider the simpler semantics of the existing GML ontology as sufficient for representing the spatial semantics of a model or data as opposed to the proposed GML ontology generated from XML Schema. Also, transformations from a language with expressivity and more overlapping modeling concepts with OWL may produce a more concise ontology. This may be desirable to avoid post-transformation modifications of the generated ontologies, as was implemented in [MF18].

Alternatively, Brink et al. [Bri+14] proposes using the ISO 19150-2 standard for UML to OWL transformation of geographic information standards instead of a direct XML Schema to OWL transformation. Initial explorations indicate that the ISO 19150-2 ontology mappings may be favorable since there is less language heterogeneity

between UML and OWL compared to XSD and OWL [Bri+14; Cox13]. However, ISO 19150-2 does not propose a transformation of data instances into OWL, and thus may only improve the generated 3D urban data model without supplemental transformations. This alternative approach is explored in chapter 5 and compared with our approach proposed in this chapter.

In addition, the proposed transformations occasionally produce conflicting axioms in the generated ontologies depending on if duplicate names are used to define properties. For instance, the `gml:exterior` element is declared twice in GML 3.1: once as a child element of a convex 3D 'solid' and again as a child element of a 2D or 3D 'ring'. This is possible in XML Schema due to how locally declared elements have a local scope to their parent types. Semantically, both of the `gml:exterior` element declarations refer to the same concept, but the identical naming of these elements results in two distinct properties being mapped to the same property in OWL. This may cause inconsistencies when declarations of different elements of the same name are declared that map to different OWL concepts, e.g., if an `owl:Class` and an `rdfs:Datatype` of the same name are declared, an inconsistency would arise as in OWL-DL, an entity cannot be class and a datatype at the same time. [Cha+21] notes that similar results were observed in the CityGML 2.0 ontology provided by the University of Geneva which must be manually corrected.

There are also two aspects of the proposed approach where data loss is unavoidable. One is concerning the transformation of `xs:sequence` which is used to declare a set of valid *ordered* child elements of complex types. In a graph formalism such as RDF, the ordering of properties is not permitted without introducing some amount of structural or semantic heterogeneity (although approaches for accomplishing this exist such as reification or the LISP-like `rdf:List` vocabulary). Additionally, the GeoSPARQL standard has not currently implemented support of 3D geospatial queries (although this is in development, cf. section 7.1), meaning GeoSPARQL can only be used to store 3D GML data but not query it. Approaches like BimSPARQL [ZBV18] have been proposed for facilitating 3D spatial queries, however, these approaches are specific to BIM data standards such as IFC. Currently, generic 3D geospatial query frameworks do not exist for linked data and graph formats like RDF.

4.5 Conclusion

This chapter highlights several major contributions of this thesis. Notably, improvements were proposed to the existing model-driven transformation approaches of semi-structured data (such as XML) towards formal knowledge graph languages and formats (contribution **C4**) such as the transformation mapping of the `xs:choice` construct. While the proposed transformation approach is largely generalized, several specializations for common geospatial data standards in the data formats they consider such as GML and GeoSPARQL are also proposed. The proposed transformation workflows are also made available through a proof of concept transformation tool UD-Graph[6] that will be discussed in chapter 6. Accordingly, these proposed transformations are used firstly, to integrate an evolving 3D urban data standard, CityGML (**C1**), as a formal 3D urban data ontology. Secondly, these transformations are used to integrate real-world 3D city model data into conformant instances of the generated ontology (**C5**).

During the experimentation of the work proposed in this chapter, the advantages and limits of our approach were tested, and several perspectives are identified for future works. Chapter 5 will explore the impact of using abstract data models such as conceptual UML models to guide model-driven transformations. How these transformation approaches can be applied to data model extensions—a common practice for providing interoperability across information domains—is taken into consideration. Finally, the use of formal rules will be further explored to improve the validation of 4D urban data transformed using these methods through inference-based approaches such as reasoning.

[6] https://github.com/VCityTeam/UD-Graph

5 Leveraging Standards in nD Urban Data Ontology Generation

This chapter explores how using more abstract and conceptual data models can affect the model-driven data integration process. In addition, this chapter improves upon the proposed methodology by including model-driven transformations of data model extensions in conjunction with the models they extend. It also improves upon the proposed constraint-based conformance validation approach of data instances by implementing formal 'horn-like' rules to facilitate more complex data validation In particular, these improvements are applied to better integrate the temporal dimension of evolving urban data.

To respond to the nD urban data integration problems detailed in section 1.1, this chapter presents the aforementioned contributions from section 1.3:

- **C1.b:** Formalization of additional nD urban data models from evolving standards
- **C3.a:** Formalization of rules for validating temporal relations
- **C4.b:** Further development of a model-driven data transformation workflows
- **C5.a:** Integration of additional evolving 3D urban datasets from real-world open data as linked data

This chapter presents work published in the proceedings of the 19th European Semantic Web Conference [Vin22] and in the ISPRS of the Photogrammetry, Remote Sensing and Spatial Information Sciences Annals[Vin+21a]. The latter publication was presented at the 16th international 3D GeoInfo Conference where the work received the **"Outstanding Paper"** award in the category of *Data Integration and Information Fusion*[1].

[1] https://3dgeoinfo2021.github.io/

5.1 Methodology

To improve our approach proposed in chapter 4 and provide the aforementioned contributions, a more detailed methodology for integrating nD data and data models is proposed (figure 5.1). This methodology is divided into 6 steps (**S1–S6**).

Step **S1** starts with the identification of *data standards*, *models*, and relevant *encodings* and *data formats*. In this chapter, a more abstract language should be selected as the source language to respond to research question **RQ3** (section 1.1) and the discussion raised in section 4.4: *How are model-driven transformation approaches affected by models at different levels of abstraction?*. For this, *UML* is identified as the source language. As discussed in section 2.1.1, although alternative languages exist for creating highly abstracted data models (e.g., EA/ER languages for database management or EXPRESS for defining Building Information Modeling standards like IFC), UML is more widely adopted by academics in computer science fields and data standardization organizations from different urban data silos regarding today's urban and geospatial data models. It also has the most mature model-driven transformation tools and frameworks such as MOF and MDA (section 3.2.1). To compare the effects of utilizing more abstract data models to effectuate model-driven transformations with our approach proposed in chapter 4, the same target language is chosen (*OWL 2 DL*).

As suggested by Noy, McGuinness, et al. [NM+01], ontology development is an iterative process. For this reason, each step of the methodology proposed in this section is concluded by a validation activity (highlighted in green in figure 5.1). If the result of a transformation is determined to be invalid, the proposed transformation definitions and source models from previous steps may need to be adjusted (if not, the produced ontology or data must be corrected). Since RDF(S) and OWL-DL are respectively, a concrete syntax and a formal (logical) languages, the validation activities of steps **S2–S5** can be automated using inferencing approaches such as reasoners like was done in section 4.3.2. **S5** goes beyond using constraints for data validation and implements horn rules (in SWRL) for enriching the logical validation process of the transformed data (discussed in section 5.2.4). These rules primarily focus on the validation of temporal data and the temporal relations these data may have.

Additionally, this chapter will examine model-driven transformations that take into consideration UML models that use the General Feature Model UML profile proposed by the OGC TC 211 working group as an ISO standard. As mentioned in section 2.1.2, the GFM metamodel is used in many geospatial data standards (including GML)

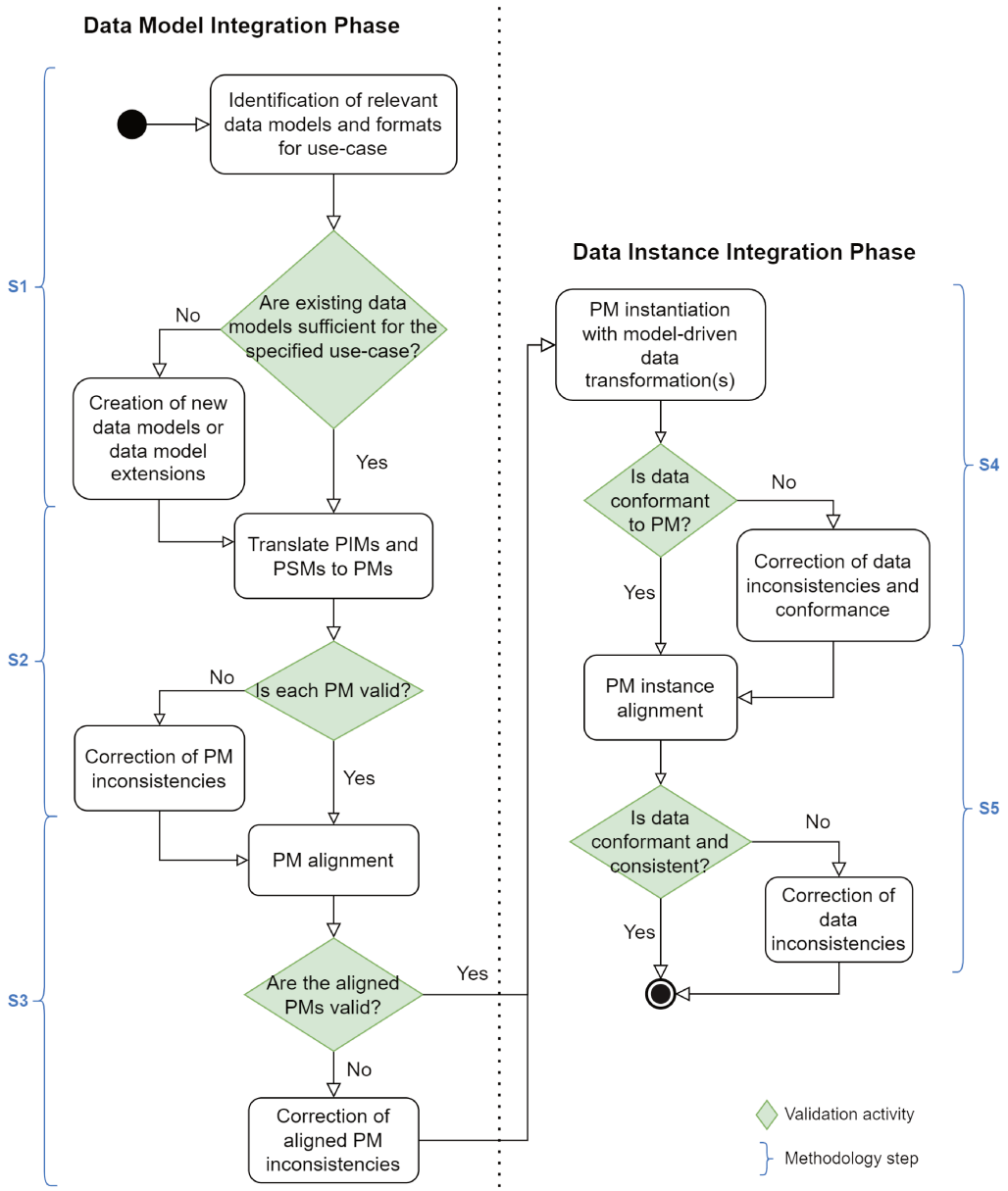

Figure 5.1.: Our proposed methodology for standards-based model-driven nD data integration. Phase 1 (left) details the process for integrating data models. Phase 2 (right) details the process for integrating data instances utilizing the models created in Phase 1. The methodology is divided into 5 steps (**S1–S5**) highlighted in blue. Model and data validation activities are highlighted in green. Also, models at different levels of abstraction are referred to using MDA concepts (PIM, PSM, PM).

to provide an interoperable vocabulary for geospatial information. The rest of this chapter will illustrate how steps 2 through 6 of the methodology are implemented as follows. Section 5.2 discusses in detail steps **S2–5** of the proposed methodology.

Section 5.3 illustrates how the methodology was applied to create an nD urban data ontology from the CityGML data model, extensions to the CityGML data model, and transform real-world CityGML data. Finally, section 5.4 discusses the advantages and disadvantages of the methodology and the experimentations performed with it and concludes this chapter with the perspectives of the work presented thus far.

5.2 Abstract model-driven data integration approach

Our proposed integration methodology in this chapter has several differences from the methodology presented in chapter 4. As discussed in section 3.2.1, model-driven transformations between data formats (at MOF metamodeling level M_0) based on more conceptual models may need to take into consideration encoding rules of the source and target formats. This is especially relevant for PIM data models. By definition, PIMs define the structure of data in a way that features at least model heterogeneity (problem **P1.c**) between their representations of data and how data is instantiated according to the syntax of some data transfer format. This is not an issue for unabstracted PM data models which directly define the structure data with the same modeling paradigm as the data. Thus, the proposed transformations take into consideration relevant encoding rules as shown in figure 5.2.

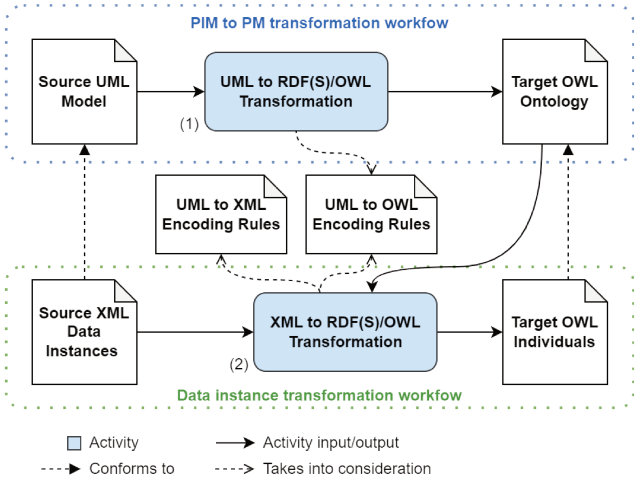

Figure 5.2.: An overview of the proposed model-driven transformations based on abstract data models. The first activity is a UML to RDF(S)/OWL transformation of the CityGML model. The second activity is a model-driven transformation using the generated ontology to guide the transformation of data conformant to the CityGML model.

In addition, this approach proposes using the generated target ontology to guide the data instance integration workflow instead of generating an intermediate transformation from the source model (figure 5.2). This permits the data instance transformation to take into consideration alignments (or extensions) to the transformed models created after the data model transformation. The following section discusses how heterogeneity issues are resolved between UML and OWL during the proposed data model integration phase.

5.2.1 Step S2: UML PIM to OWL PM Data model transformation

Several approaches for transforming UML to OWL were discussed in section 3.2.1. Of these approaches, this work proposes using the transformation mappings proposed by ISO 19150-2 for two reasons. First, these mappings are generic enough to transform UML PIMs to PMs in OWL, meaning a ISO 19150-2 based approach can theoretically be applied to any existing UML model. While PSMs like OWL-specific UML models may use UML profiles or graphical syntaxes not native to UML to define a more precise transformation of OWL-specific UML models to OWL ontologies, these data models are less common and cannot be transformed to PM targets other than OWL ontologies. Secondly, this approach is preferred as it enables the transformation of concepts from the GFM metamodel for defining non-domain specific geospatial information. As discussed in section 2.1.2, the GFM is used by standardizing organizations like the OGC and the ISO 211 TC to provide shared geospatial and geometric vocabularies between data standards, including the CityGML 2.0 and 3.0 conceptual UML models. However, implementing the ISO 19150-2 transformations requires taking into consideration the known limitations, ambiguities, and best-practices of the standard. Building off of the works discussed in section 3.2.1, this thesis proposes three approaches regarding the transformation of `<<Union>>`, `<<Enumeration>>` and `<<CodeList>>` from the GFM metamodel to OWL.

Transforming «Union» from the GFM metamodel to OWL

The existing transformation approaches for `Union` are introduced in section 3.2.1. In order to identify which `Union` transformation approach to utilize, the following strategy is proposed in table 5.1.

Mappings 1 and 2 are inspired from [ZL12] which proposes transformations of unions to classes and union attributes to properties and sub-properties (figure 5.3).

	UML pattern	Proposed transformation mapping
1.	Union with attribute members that reference **only classes**	Transform the Union into a `owl:ObjectProperty`; transform attribute members to `rdfs:subPropertyOf` the ObjectProperty (based on [ZL12])
2.	Union with attribute members that reference **only data types**	Transform the Union into a `owl:DatatypeProperty`; transform attribute members to `rdfs:subPropertyOf` the DatatypeProperty (based on [ZL12])
3.	Union with attribute members that reference **both classes and data types**	"Flatten" the Union into `owl:ObjectProperties` and `owl:DatatypeProperties` as proposed in [Ope17]

Table 5.1.: The proposed transformation mappings for Unions to OWL.

However, this thesis does not propose transforming the `Union` itself to a class as proposed in [ZL12]. [Ope17] notes that when instantiating the ontology, unions transformed into classes are instantiated as "intermediate" individuals between the instances of the class referencing the union and the instantiated value of the union[2]. During the data instance transformation, these "intermediate" individuals may introduce structural heterogeneity (**P1.b**) between data stored in RDF and other formats. Instead, mappings 1 and 2 propose transforming the union into only a superproperty and union attributes into sub-properties of the superproperty.

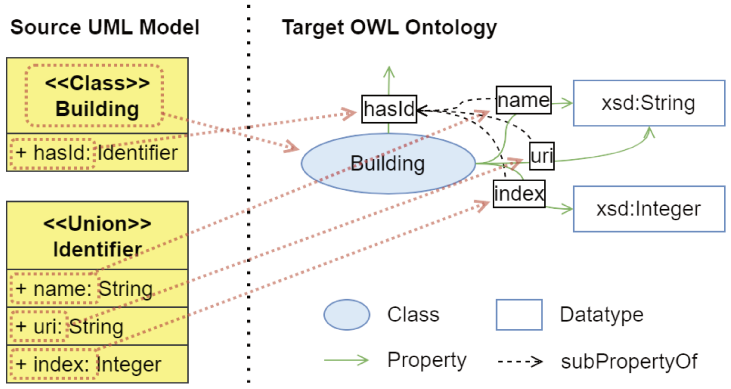

Figure 5.3.: An example of a Union transformation according to mapping 2. The union, *Identifier*, contains 3 attributes of primitive datatypes that are transformed into datatype properties. An attribute, *hasId*, references the union and is also transformed into a datatype property. The datatype properties created from the union are declared as sub-properties of the *hasId* property.

[2]This behavior is similar to how *xs:group* and *xs:attributeGroup* function as discussed in section 4.2.2

Mapping 3 is proposed for when the attributes of the `Union` would be transformed into a combination of ObjectProperties and DatatypeProperties which cannot share the same superproperty in OWL (figure 5.4). This approach is less desirable to mappings 1 and 2, as it implies the semantic loss (**P1.a**) of the `Union` itself during transformation, however, it generates less structural heterogeneity (**P1.b**) by producing a more concise representation of a `Union`.

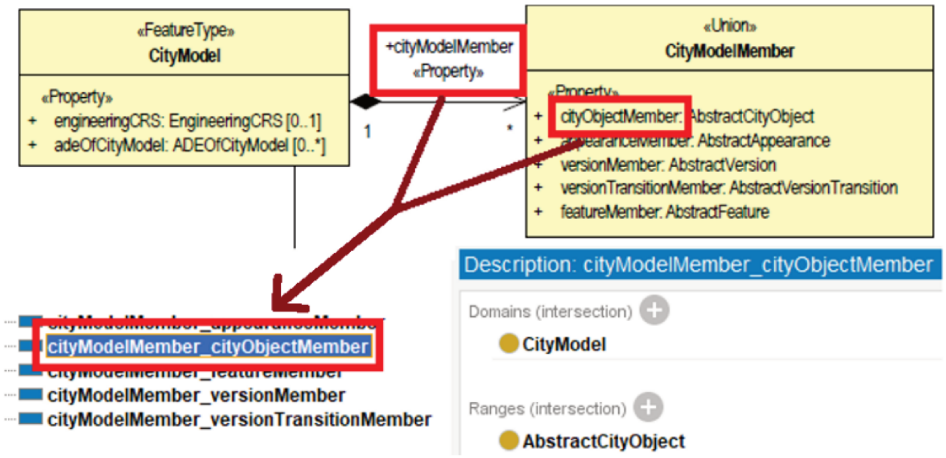

Figure 5.4.: An example of Union flattening (mapping 3) of *core:cityModelMember* and *core:cityObjectMember* (top) and their mappings to OWL (bottom) from the CityGML 3.0 UML model.

Transforming «CodeList» and «Enumeration» from the GFM metamodel to OWL

As discussed in section 2.1.2 and appendix B.1, a `CodeList` in OWL can be transformed into a SKOS concept scheme (either an internally or externally defined scheme). Also, `Enumeration` can be transformed to either `owl:DataUnionOf` or to SKOS concept schemes as there is little distinction between a `CodeList` and a `Enumeration` (besides the requirement that a `CodeList` must be extensible). Because of this similarity, this chapter proposes treating `Enumeration` as `CodeList` and transforming `CodeList` to externally defined SKOS concept schemes based on the approach proposed in [Ope17]. This approach allows `Enumeration` representations in OWL to be reused across different versions of the same model and provides a mechanism for easily extending their vocabularies. To ensure the predefined values of the source `Enumeration` are preserved, the external vocabulary must contain a concept scheme with the values as concepts of the scheme. An exemplification of how this method of transforming these concepts is useful for supporting cross-

domain vocabularies is shown in section 5.3 regarding heterogeneous vocabularies for defining temporal urban evolution.

5.2.2 Step S3: Data model alignment

When applying a model-driven transformation approach based on ISO 19150-2, the ontologies generated by these transformations can be aligned before or after the transformation. Modular ontology networks can be automatically produced from a single UML model by dividing the model using a `<<Package>>` per desired ontology. This can also be applied to extensions of existing UML models by declaring new classes, attributes, associations, and data types in a unique package. In this approach, any generalizations and `AssociationRoles` declared between classes of different packages are transformed into `rdfs:subClassOf` and corresponding `owl:ObjectProperty`, which represent the correspondences of these alignments.

Additionally, alignments can be similarly declared towards external, existing ontologies. This can be done by declaring these types of relationships in a UML model towards a class whose package namespace and class name correspond to the IRI namespace and URI fragment of some existing OWL class. Alternatively, custom transformation mappings towards well-known classes from existing ontologies can be utilized as proposed in [Ope17]. Certain transformation tools such as ShapeChange[3] enable this approach by permitting declaring custom transformation mappings in this manner. An example of this approach is proposed in section 5.3.2.

5.2.3 Step S4: Data instance transformation

This section proposes a model-driven syntactic transformation between XML and RDF to effectuate the instantiation of networks of ontologies from existing data. To do this, the class and properties axioms (or the TBox) of the ontology network are queried to determine what `owl:Class`, `owl:ObjectProperty`, `owl:DatatypeProperty`, or `rdfs:Datatype` a datum should be asserted as an instance of (as a part of the ABox of the ontology). These queries must answer the following questions of an atomic datum within a dataset:

- Q1: Which Class, Property, or Datatype in the network defines the datum?
- Q2: Does the datum represent geometry?

[3] https://shapechange.net/

- Q3: Does the datum represent a temporal property?

To answer Q1, the different types of axioms used to constrain classes and properties of formal ontologies must be taken into consideration. For example, universal and/or existential quantifications may be used in the definition of classes to refer to any properties they may have. Additionally, the `rdfs:domain` and `rdfs:range` of properties may be declared, creating references to the classes that may instantiate them. These types of axioms provide information on what properties an individual may have at a distance of 1 in a knowledge graph. In addition, semantic or structural heterogeneity introduced by transformation mappings must be considered. For example, the "flattening" and local naming convention approaches proposed by [Ope17] in response to the limitations of ISO 19150-2 must be taken into consideration if they are used during step **S2** of the method.

When transforming structured tree or hierarchical data structures, such as XML, data is considered as a set of nodes, where each node may have parents and/or children. Determining if a node should be transformed into an individual asserted as a type of a class or a property, is done by querying if each node has classifying information and the classifying information of its parent and children. In XML, a datum is an `element` which is classified by its *tag*. Searching if a class or property axiom exists in the ontology network with the same identifier as the `element`'s XML tag can determine what said XML element should be transformed into in OWL.

For instance, query listing 5.1 shows how this can be done for verifying if a node corresponds to an `owl:ObjectProperty` with a SPARQL query. This query uses a node's classifier "*nodeType*" and its parent's classifier, "*parentType*", to return if an object property exists that contains a class with the identifier *parentType* in their domain (lines 3–4) or if a universal quantification class restriction exists of a property with the identifier *nodeType* (line 7–12)[4]. Lines 5 and 13 are used to include all declared subclasses and equivalent classes that correspond the parent node's classifier in the query.

```
1 ASK WHERE {
2   {
3     :nodeType a owl:ObjectProperty ;
4       rdfs:domain ?domain .
5     :parentType (owl:equivalentClass|rdfs:subClassOf)* ?domain .
6   } UNION {
7     ?someClass a owl:Class ;
8       rdfs:subClassOf [
9         a owl:Restriction ;
```

[4]Note that if ontologies in the network are generated using local naming conventions, the identifier *:nodeType* should be prefixed with the name of the as *:parentType[separator]nodeType*.

```
10          (owl:allValuesFrom|owl:someValuesFrom) ?someOtherClass ;
11          owl:onProperty :nodeType
12        ] .
13      :parentType (owl:equivalentClass|rdfs:subClassOf)* ?someClass .
14    }
15  }
```

Listing 5.1: A query for finding an ObjectProperty assertion

Regarding Q2 and Q3, determining which data are considered geometric or temporal information is specific to the data standards being integrated and any relevant encoding rules being considered. For example, a transformation of GML data could reuse our approach proposed in section 4.3.1, where any XML element in the `substitutionGroup` of (i.e., inherits from) `{http://www.opengis.net/gml}_Geometry` and `{http://www.opengis.net/gml3.2}AbstractGeometry` are considered geometry.

Once these types of queries are defined, they can be used to transform the data of a tree structure to individuals in the ABox of the ontology network as shown in algorithm 1[5]. To do this, the algorithm iterates over all input nodes of the tree (line 2). In line 3, the function $name(x)$ is used to return the 'classifying identifier' of a node (e.g., an element tag in XML). The result of $name(x)$ is compared to the concept names (or class identifiers) and role names (or property identifiers) in the ontology network. A triple will be created if the identifiers of the children of the node correspond to either an `owl:Class` (line 7), an `owl:ObjectProperty` (line 14), an `owl:DatatypeProperty` (line 21), or an `rdfs:Datatype` (line 28). All of these cases will construct and add the corresponding triple to a set representing the graph to be returned.

This algorithm also uses three utility functions: $push(x)$, $serializeGeometry(x)$, and $generateID(x)$. $push(x)$ simply adds a triple to G, the set of triples to be output by the algorithm. $serializeGeometry(x)$ is used to generate primitive string values from an input tree node and its descendant nodes. These values can be used to instantiate geometric data as `rdfs:Datatypes` such as GeoSPARQL's `geo:gmlLiteral` (lines 10–12) as proposed in section 4.3.1. The nodes processed by this function could either be removed from the input tree to speed up the algorithm or left in the input tree to capture the structure and semantics of geometry in the generated graph. The $generateID(x)$ function creates a unique identifier that corresponds to the input node. In tree data structures where the number of input nodes is known, such as XML documents, this identifier could correspond to the index of the node in the tree.

[5] In this algorithm XML attributes are treated as child elements of the element they belong to.

Algorithm 1 An algorithm for transforming an XML tree structure to individuals of an ontology network. Comments are denoted by '▷'.

Input: An ontology $O = \{O_C, O_{OP}, O_{DP}, O_D\}$, where O_C is a set of classes, O_{OP} is a set of object properties, O_{DP} is a set of datatype properties, and O_D is a set of datatypes

Input: A set of tree nodes, $N = \{n_1, n_2, \ldots n_m\}$, where each node, n_{1-m}, has a set of child nodes, $n.children$ and a (possibly empty) primitive value, $n.value$

Output: a set of triples $G = \{\langle s_1, p_1, v_1\rangle, \langle s_2, p_2, v_2\rangle, \ldots \langle s_n, p_n, v_n\rangle\}$

$G \leftarrow \{\}$
for n in N **do**
  if $\exists name(n) \in O_C$ **then** ▷ $name(x)$ returns a concept or role name from x
    $nId \leftarrow generateID(n)$
    $G.push(\langle nId, \texttt{'is a'}, name(n)\rangle)$
    for c in $n.children$ **do**
      if $\exists name(c) \in O_C \wedge \exists p \in O_{OP} \wedge (p.name(c) \sqsubseteq name(n))$ **then**
        $cId \leftarrow generateID(c)$
        $G.push(\langle nId, p, cId\rangle)$
        if $c \sqsubseteq Geometry$ **then** ▷ $Geometry \equiv$ `geo:Geometry`
          $geometry \leftarrow serializeGeometry(c)$
          $G.push(\langle cId, hasSerialization, geometry\rangle)$
        end if ▷ $hasSerialization \equiv$ `geo:hasSerialization`
      else if $\exists name(c) \in O_{OP}$ **then**
        for gc in $c.children$ **do**
          if $\exists name(gc) \in O_C$ **then**
            $gcId \leftarrow generateID(gc)$
            $G.push(\langle nId, name(c), gcId\rangle)$
          end if
        end for
      else if $\exists name(c) \in O_{DP}$ **then**
        if $c.value$ **then**
          $G.push(\langle nId, name(c), c.value\rangle)$
        end if
        for gc in $c.children$ **do**
          $G.push(\langle nId, name(c), gc.value\rangle)$
        end for
      else if $\exists name(c) \in O_D \wedge \exists p \in O_{DP} \wedge (p.name(c) \sqsubseteq name(n))$ **then**
        $G.push(\langle nId, p, c.value\rangle)$
      end if
    end for
  end if
end for

5.2.4 Step S5: Temporal Data instance integration and validation

After the target ontological data model has been instantiated, these data instances can be aligned to complete the integration process for providing unified views of data. In this step, data is aligned according to the needs of the application of data integration. For example, as discussed in section 2.3, integration processes are often performed in this context to combine snapshots, event-based, and versioned models of the urban landscape into continuous 4D (3D+time) models of city evolution. A real-world integration use case explores this process in section 5.3.

Like our approach in chapter 4, reasoning approaches on ontological constraints are used to determine if the transformed data instances (ABox) are conformant to the network (TBox). In the context of validating the evolution of temporal scenarios of urban evolution using existing data standards, this section proposes an extension to OWL-Time to facilitate logical temporal data validation between temporal instants and intervals. This is required for snapshot-based and versioned city models where specific snapshots or versions of urban areas are only considered valid at the instant of time when they are published. OWL-Time is used as opposed to other existing temporal models (discussed in section 2.3) or proposing a new temporal model as it is a candidate W3C recommendation that is already widely adopted which aligns with the standards-based approach of this thesis.

The proposed extension contains instant-instant and instant-interval relations for reasoning-based validation of temporal evolution. Figure 5.5 illustrates the existing temporal relations from OWL-Time and the proposed relations highlighted in orange.

The following proposed relations are defined as `owl:ObjectProperties` as follows:

- `time_ext:in`: If an instant T_1 is '*in*' a proper interval T_2, then the beginning of T_1 is after the beginning of T_2 or is coincident with the beginning of T_2, and the end of T_1 is before the end of T_2 or is coincident with the end of T_2, except that end of T_1 may not be coincident with the end of T_2 if the beginning of T_1 is coincident with the beginning of T_2. This relation complements the existing `time:intervalIn` for comparing proper intervals.
- `time_ext:during`: If an instant T_1 is '*during*' a proper interval T_2, then the beginning of T_1 is after the beginning of T_2, and the end of T_1 is before with the end of T_2. This relation complements the existing `time:intervalDuring` for

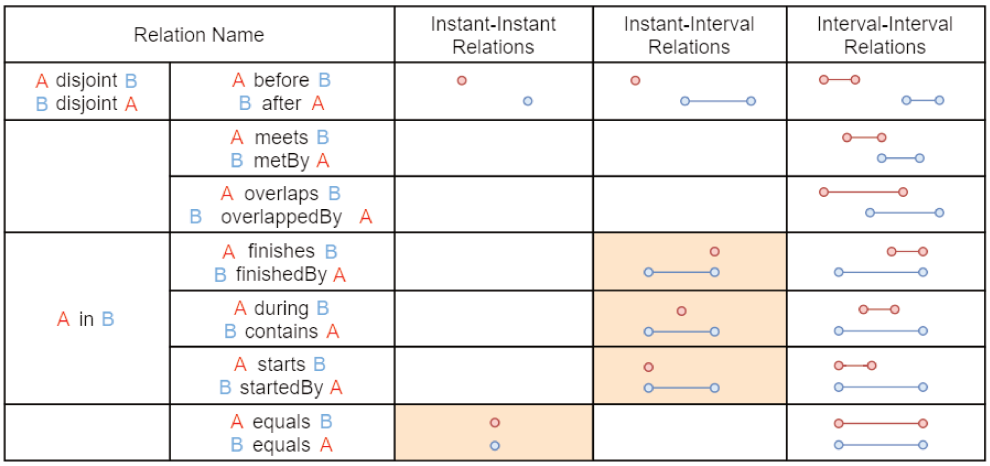

Figure 5.5.: A synthesis of the existing temporal relations in OWL-Time and the proposed instant-instant and instant-interval relations (highlighted in orange).

comparing proper intervals. Additionally, the property `time_ext:contains` is proposed as the inverse property of `time_ext:during`.

- `time_ext:finishes`: If an instant T_1 '*finishes*' a proper interval T_2, then the beginning of T_1 is after the beginning of T_2, and the end of T_1 is coincident with the end of T_2. This relation complements the existing `time:intervalFinishes` for comparing proper intervals. Additionally, the property `time_ext:finishedBy` is proposed as the inverse property of `time_ext:finishes`.
- `time_ext:starts`: If an instant T_1 '*starts*' a proper interval T_2, then the beginning and end of T_1 is coincident with the beginning of T_2, and the end of T_1 is before the end of T_2. This relation complements the existing `time:intervalStarts` for comparing proper intervals. Additionally, the property `time_ext:startedBy` is proposed as the inverse property of `time_ext:starts`.
- `time_ext:equals`: If an instant T_1 '*equals*' another instant T_2, then the beginning of T_1 is coincident with the beginning of T_2, and the end of T_1 is coincident with the end of T_2. This relation complements the existing `time:equals` for comparing proper intervals.

To facilitate the temporal validation of these relations 'horn-like' rules are proposed in appendix C.2 (formalized in SWRL). These rules are proposed with an exclusive interpretation of the bi-temporal timestamps of intervals. For example, to infer that a temporal instant, i_1 occurs before i_2, the following rule is proposed using the Description Logic language ($\mathcal{AL}$) introduced in section 2.2 and defined in appendix C.2 with horn rules:

$$\begin{aligned}\forall a \forall b(hasEnd(a, i_1) \wedge Instant(i_1) \wedge inXSDDateTimeStamp(i_1, t_1) \\ \wedge hasBeginning(b, i_2) \wedge Instant(i_2) \wedge inXSDDateTimeStamp(i_2, t_2) \\ \wedge lessThan(t_1, t_2) \rightarrow before(a, b))\end{aligned} \tag{5.1}$$

In addition to inferring new relations, rules are proposed to infer inconsistencies. For example, the bi-temporal timestamps of a `time:TemporalEntity` are declared with the properties `time:hasBeginning` and `time:hasEnd`. These timestamps are temporal `time:Instants` that use the datatype property `time:inXSDDateTimeStamp` to define when they exist through an `xsd:DateTime` datatype value. A temporal entity that has an ending timestamp that occurs *before* its beginning timestamp is logically inconsistent. This can be inferred with the following rule for a temporal entity, T:

$$\begin{aligned}\forall T(TemporalEntity(T) \wedge hasBeginning(T, i_1) \wedge Instant(i_1) \\ \wedge inXSDDateTimeStamp(i_1, t_1) \wedge hasEnd(T, i_2) \wedge Instant(i_2) \\ \wedge inXSDDateTimeStamp(i_2, t_2) \wedge greaterThan(t_1, t_2) \\ \rightarrow Nothing(T))\end{aligned} \tag{5.2}$$

Finally, two more object properties are proposed as part of the extension, `time_ext:hasExistenceTime` and `time_ext:hasTransactionTime`. These properties represent the association of an entity to a temporal entity (instant or interval) that represents the time when that thing exists in the real world or when the entity is added or modified in a dataset or database. These properties are declared as `rdfs:subPropertyOf` the `time:hasTime` property.

5.3 Experimentations

Using the established transformation workflows and temporal validation rules, our proposed methodology is applied to integrate static real-world 3D semantic city model snapshots to provide a unified view of scenarios of urban evolution. To apply this approach to an evolving standard—and compare the results of this approach with the previously proposed method (chapter 4)—the CityGML 2.0 and 3.0 will be integrated are chosen for this integration application.

To effectuate this integration application, the following data integration pipeline is implemented (figure 5.6), based on our proposed methodology. First, two conceptual data models are prepared for the UML to OWL transformation, one for CityGML 2.0 and another for CityGML 3.0. A model for CityGML 2.0 is created from the data standard's documentation[6]. As the Versioning module from CityGML 3.0 does not exist for CityGML 2.0 but is important for this integration use case, the Versioning ADE proposed in [Cha+17] is updated and added to the CityGML 2.0 UML model. Finally, GML and `<<Feature>>` classes in the CityGML 2.0 UML model are replaced with their equivalent classes from ISO 19107 and the `<<FeatureType>>` metaclass from the GFM metamodel. This allows geometry mappings applied to version 3.0 of the model to be reused for version 2.0. Extracts of these UML models are provided in appendix A.

[6] https://portal.ogc.org/files/?artifact_id=47842

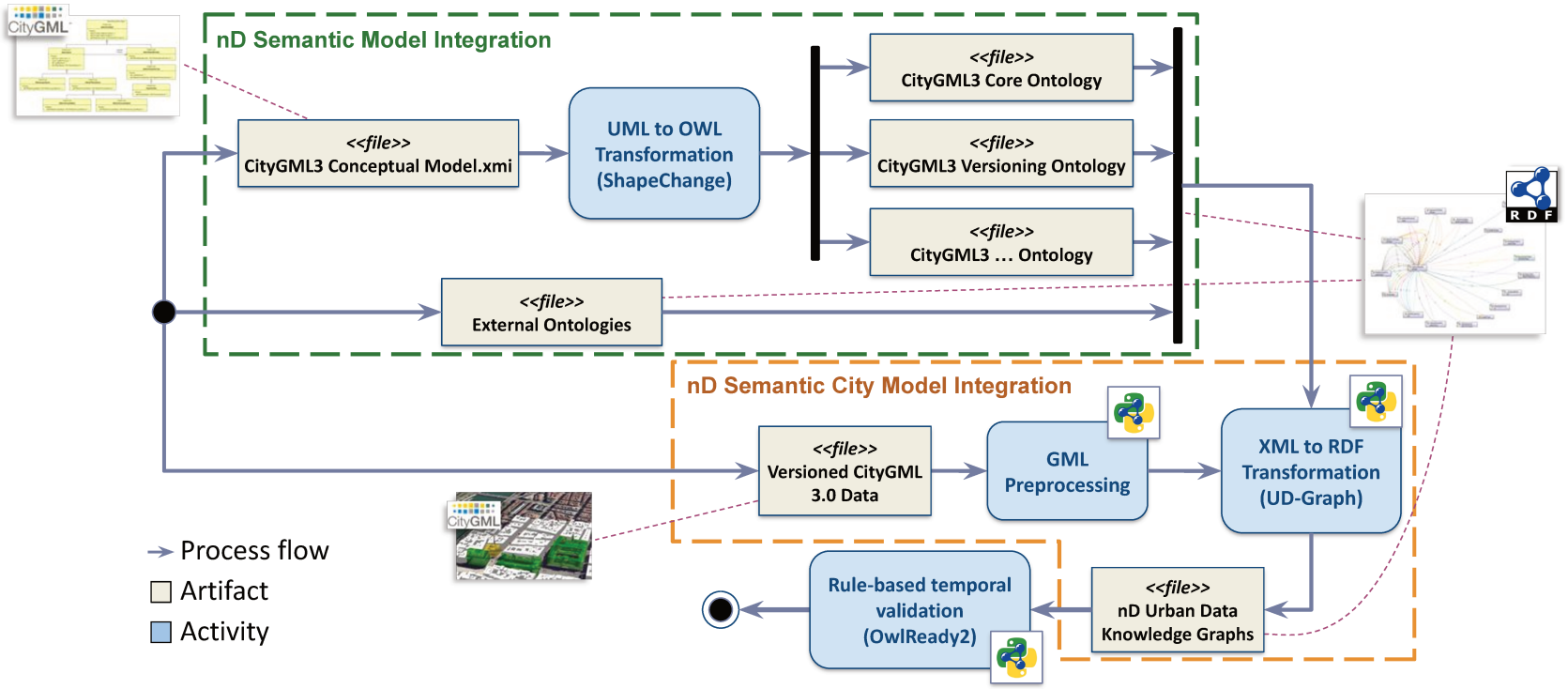

Figure 5.6.: UML activity diagram of implemented model-driven nD urban data integration pipeline exemplified with the CityGML 3.0 standard. The data model transformation workflow (**S2-S3**) is highlighted in green and the data instance transformation workflow (**S4-S5**) is highlighted in orange.

5.3.1 Transforming the CityGML Model to an Ontology Network

Once the source models have been identified and prepared, they can be transformed into the targeted data format and language. Transformation tools such as ShapeChange[7] can be used to read UML models and execute model-driven transformation towards various PM language targets such as XML Schema, JSON Schema, and RDF(S)/OWL (using transformations based on ISO 19150-2). Figure 5.7 illustrates the effectuated transformation of the CityGML UML models including generated alignments (as dependencies at the PM level) to existing ontologies.

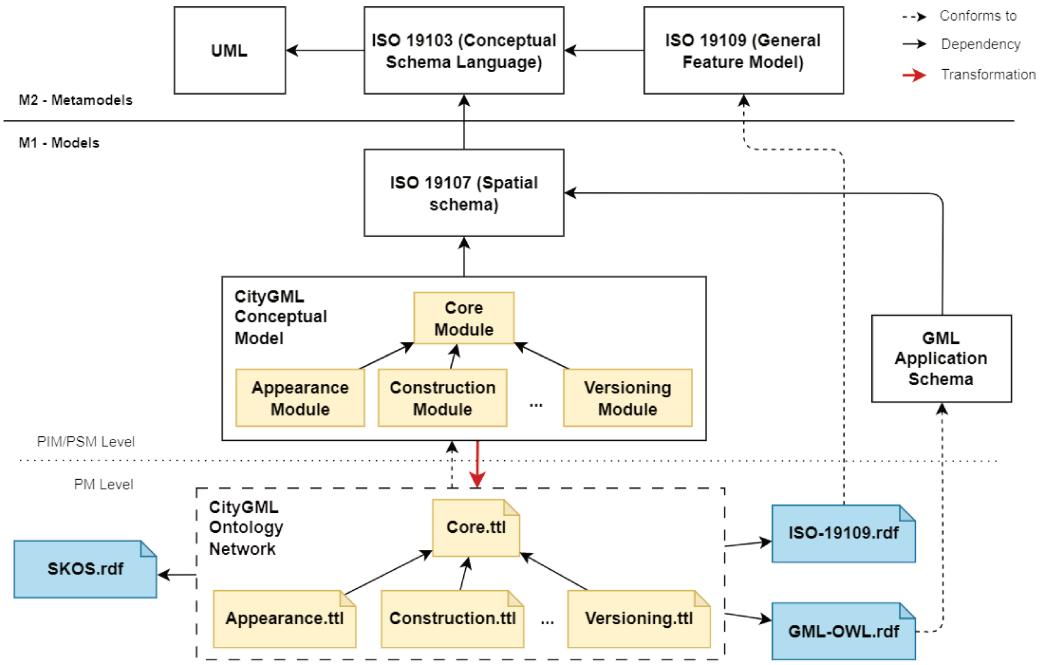

Figure 5.7.: Dependencies and conformance between the CityGML, GML, and GFM models before and after transformation. At the PM level, pre-existing models that are being reused are highlighted in blue.

Our proposed approaches in section 5.2.1 for transforming `Union`, `Enumerations`, and `CodeList` are applied. All other implemented UML to OWL transformation configurations are listed in appendix B.1. The reader should note that alternate configurations are possible depending on the integration use case and are discussed in section 4.4.

[7] https://shapechange.net/

5.3.2 Data model alignment and extension

To complete the nD urban data network of ontologies, the generated CityGML ontologies are aligned with the GeoSPARQL[8], GML[9], OGC/ISO TC 211, SKOS[10], OWL-Time[11], and the proposed OWL-Time extension ontologies. As shown in figure 5.8, these ontologies serve as top-level ontologies that provide interoperable vocabularies that can be reused by domain-specific model such as CityGML and ontologies from other domains. These ontologies are grouped into 3 categories in figure 5.8. First, the *geospatial upper ontologies* like the GML, ISO 19107[12], and ISO 19136[13] ontologies are used to provide vocabularies for geometric concepts. The GeoSPARQL and ISO 19109[14] ontologies provide vocabularies for geographic concepts such as geospatial features.

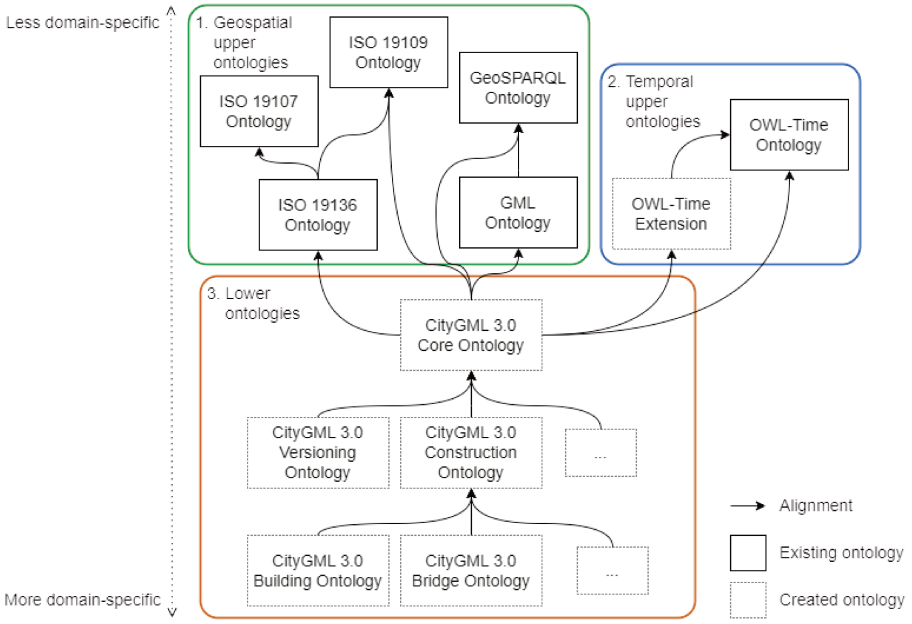

Figure 5.8.: Overview of alignments between ontologies of the proposed ontology network. Upper ontologies serve as common vocabularies that can be shared across information domains. Lower ontologies are (more) domain-specific.

The *temporal upper ontology* group contains the OWL-Time ontology and its proposed extension provide vocabularies for temporal concepts. The CityGML ontologies are

[8] http://www.opengis.net/ont/geosparql
[9] http://www.opengis.net/ont/gml
[10] https://www.w3.org/2009/08/skos-reference/skos.rdf
[11] http://www.w3.org/2006/time
[12] https://def.isotc211.org/ontologies/iso19107/
[13] https://def.isotc211.org/ontologies/iso19136/
[14] https://def.isotc211.org/ontologies/iso19109/

categorized in a more domain-specific *lower ontology* group. In this ontology network, the CityGML Core ontology serves as a sort of "middle" ontology as it functions as a domain-specific upper ontology for the other ontologies generated from the CityGML model.

Most classes are aligned primarily through subsumption axioms with the property *rdfs:subClassOf* if not other properties. Correspondences can be formalized in UML before transformation and remain formalized in OWL after transformation. For example, the superclass of all urban objects in the CityGML Conceptual Model, `core:AbstractFeature`, is declared as a subclass of the superclass `iso19107:AnyFeature` from the GFM model in the UML model and remains declared after transformation with the property *rdfs:subClassOf*. Other mappings with the GML and OWL-Time ontologies are declared during the transformation are made such as those proposed by ShapeChange[15]. The remaining alignments are created manually after transformation.

Figures 5.9 and 5.10 illustrate a subset of the ontology network, highlighting the alignments between the geospatial, temporal, and SKOS ontologies respectively. Additionally, figure 6.10 demonstrates an example of an Enumeration transformed into a SKOS concept scheme for declaring different types of transactions. As discussed in section 2.3, different snapshot-based and event-based approaches for modeling the evolution of geospatial entities may require different vocabularies. Properties in these figures are labeled with globally scoped naming conventions for readability; a local scope is used in the actual network. As proposed in [Cox13; Ope17], properties named using a local scope are identified with a URI fragment prefixed with the name of the class it was created from and a separator string:

```
#[parent class][separator][property name]
```

For example, the `vers:to` property in the Versioning module ontology of figure 5.10 would be identified with the following URI fragment using local scope naming conventions and a '.' as a separator:

```
#VersionTransition.to
```

[15] https://shapechange.net/resources/config/StandardMapEntries_iso19107-owl-gml.xml

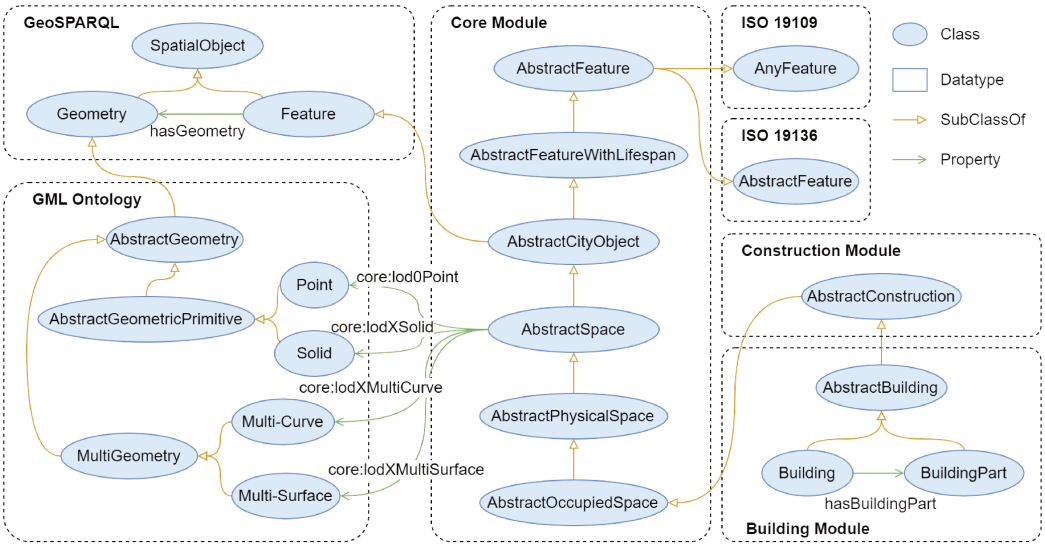

Figure 5.9.: The geospatial classes and properties of the integrated ontology network from CityGML, GML, and GeoSPARQL.

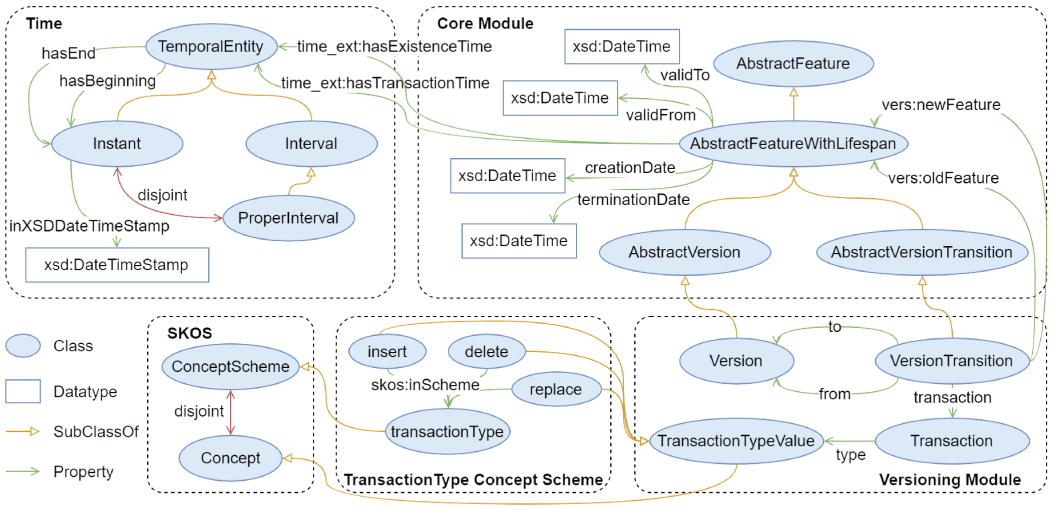

Figure 5.10.: The classes and properties of the integrated ontology network from CityGML, SKOS, and OWL-Time for modeling city evolution through versioned city snapshots.

After alignment, the ontologies represent the TBox of the ontological network. The next section shows how to integrate the data instances conformant to the original PSMs as data instances of the PM (the ontology network) and thus complete the ABox of the ontology network.

5.3.3 Data instance transformation

To illustrate how our proposed approach can be used to instantiate the ontological network, a fictional historical succession of 3D semantic city models from the 1st district of the city of Lyon, France[16] in CityGML 2.0 are used. The building and geospatial data from these datasets are converted to CityGML 3.0 with an open-source conversion tool[17] and then updated with temporal versioning data. This dataset was stored on the CityGML 3.0 Encoding GitHub[18]. The UD-Graph transformation tool introduced in chapter 4 was extended to effectuate the proposed model-driven data instance transformation from section 5.2.3.

Figure 5.11 illustrates the dataset which is composed of a historical succession of 2 versions of a city model containing 4 buildings. Version 1 of the city model, intitled `LYON_1ER-2000_01_01`, contains two buildings, `BU_69381AL50_2000-01-01` (or *'b1'*) and `BU_69381AL49_2000-01-01` (or *'b2-1'*). Version 2 of the model, intitled `LYON_1ER-2015_01_01`, also contains two buildings, `BU_69381AL49_2015-01-01` (or *'b2-1'*) and `BU_69381AL47_2015-01-01` (or *'b3'*). A version transition between these two versions is composed of 3 transactions (or changes): a deletion (`Transaction_1`) of building *b1* from version 1, a replacement (`Transaction_2`) of building *b2-1* with *b2-2*, and an insertion (`Transaction_3`) of building *b3* into version 2. The results of this transformation are visualized in figure 5.12.

Using SPARQL queries we can enrich the bi-temporal timestamps of the city objects in the dataset with the proposed OWL-Time temporal entities as shown in listing 5.2. Once enriched, the dataset can be temporally validated using tools for reading and inferring over OWL+SWRL rules such as the SWRLTab in Protégé ontology editor, OwlReady2, or SQWRL [OD10; Mus15; Lam17]. Once the ABox of the ontology network is validated as conformant to the TBox and temporally consistent, the integration process is considered complete.

[16] https://data.grandlyon.com/
[17] https://github.com/tum-gis/citygml2-to-citygml3
[18] https://github.com/opengeospatial/CityGML3.0Encodings/tree/master/CityGML/Examples

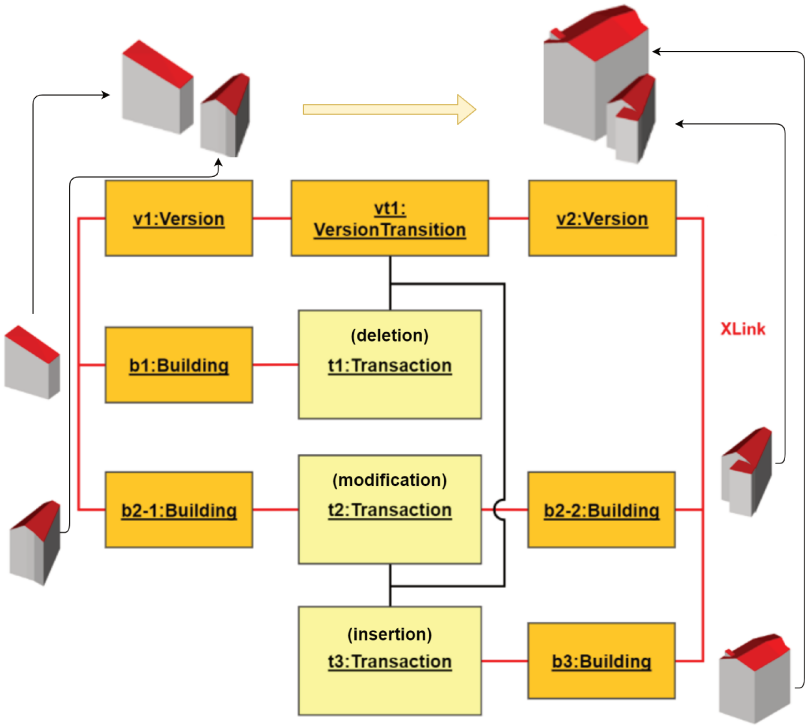

Figure 5.11.: A simplified diagram of the example historical succession of 3 buildings based on the CityGML 3.0 Versioning module.

```
INSERT {
  ?te a time:TemporalEntity ;
    time:hasBeginning ?i1 ;
    time:hasEnd ?i2 .
  ?i1 a time:Instant ;
    time:inXSDDateTimeStamp ?t1 .
  ?i2 a time:Instant ;
    time:inXSDDateTimeStamp ?t2 .
  ?f time:hasTime ?te .
} WHERE {
  ?f a ?featureClass ;
    core:AbstractFeatureWithLifespan.validFrom ?t1 ;
    core:AbstractFeatureWithLifespan.validTo ?t2 .
  ?featureClass rdfs:subClassOf* core:AbstractFeatureWithLifespan .
}
```

Listing 5.2: SPARQL query for enriching CityGML features with OWL-Time temporal entities.

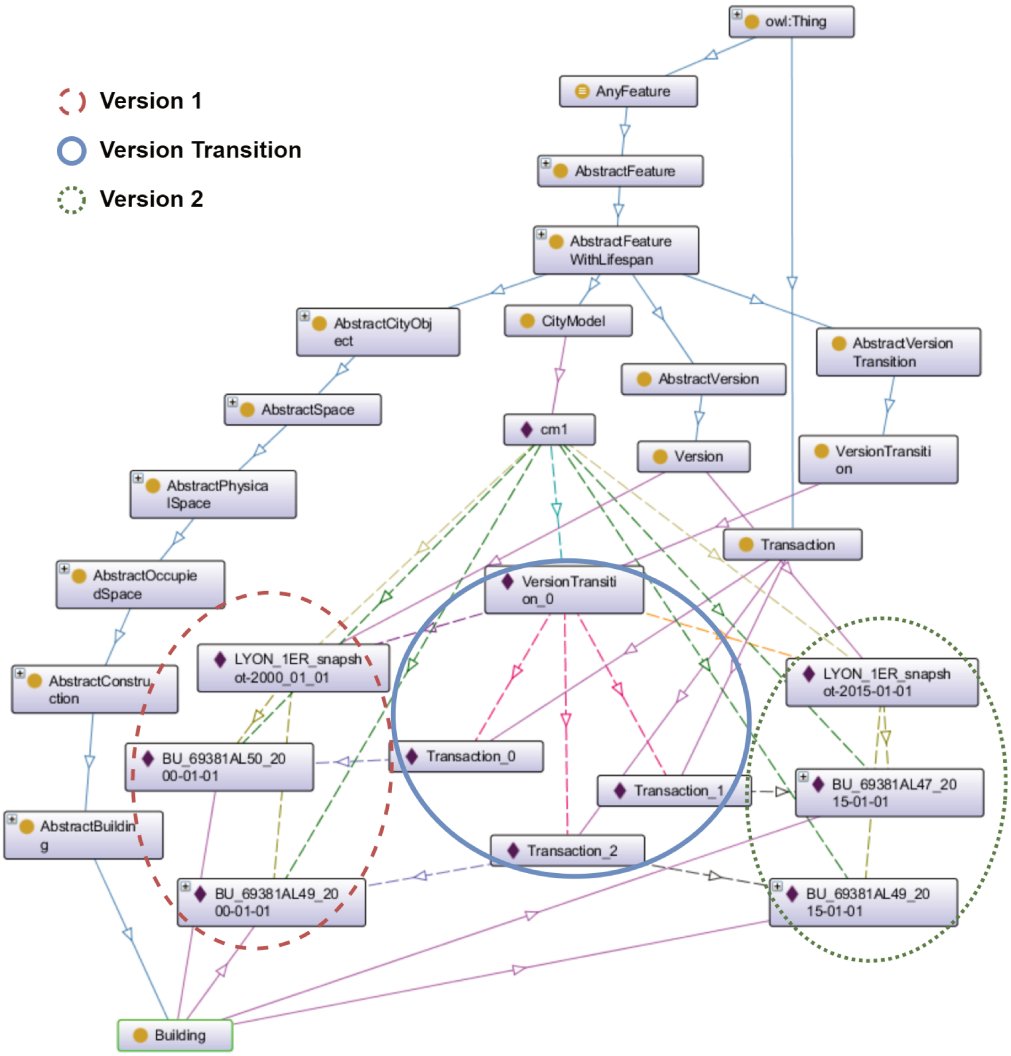

Figure 5.12.: A visualization of the transformed RDF graph of the historical succession.

5.3.4 Transformation workflow results

This section presents the final results of the effectuated transformations. The results of the UML to OWL transformations are listed in table 5.2. From these results, it appears that the UML-based approach is more concise than the previous XSD-based approach in terms of the number of `owl:Class`, `owl:ObjectProperty`, and `owl:DatatypeProperty` declarations. However, the number of axioms declared in the TBox of these ontologies is much greater in the UML-based approach. This is likely due to the transformation of constraints and annotation properties.

Ontology (network) name	# of Axioms	# of Classes	# of Object properties	# of Datatype properties
UML-based CityGML 2.0	3577	138	349	39
XSD-based CityGML 2.0	7517	1246	745	247
XSD-based CityGML 2.0 (without GML)	3061	389	329	101
CityGML 2.0 from [MF18]	1254	185	281	92
CityGML 2.0 from [Cha+21]	1760	343	263	23
UML-based CityGML 3.0	6748	374	500	141
UML-based CityGML 3.0 (without imports)	4579	285	424	108
XSD-based CityGML 3.0	8443	1647	901	230

Table 5.2.: A comparison of different ontologies generated from the CityGML UML models and XML schema. The ontologies listed do not take into consideration axioms from imported or aligned ontologies from outside the CityGML standard, e.g., GML, GeoSPARQL. Ontologies created through our approach proposed in this chapter are highlighted in bold as "**UML-based**". Ontologies created through our approach proposed in chapter 4 are "XSD-based".

Table 5.3 presents the results of the XML to OWL data instance transformations. The results of these transformations show a clear reduction in the number of individuals produced between our XSD-based approach and our approach proposed in this chapter. This appears to be, in part, due to the difference in conciseness of the ontologies as the latter approach relies on ontologies with fewer classes to instantiate and the former relying on ontologies that may contain repetitive classes or extraneous classes, however, further analysis is required.

Dataset name	# of input XML elements	# of Axioms	# of Individuals
UML-based Historical Succession	572	704	126
UML-based LYON_1ER_BATI_2015	524,990	566,245	102,140
XSD-based LYON_1ER_BATI_2015	524,990	1,844,941	424,563

Table 5.3.: A comparison of transformation results of the 3D model of the 1st district of Lyon, France from 2015 and the sample historical succession from the OGC. These metrics do not take into consideration axioms from imported ontologies. Datasets created through our approach proposed in this chapter are highlighted as "**UML**-based". Datasets created through our approach proposed in chapter 4 are "**XSD**-based".

5.4 Discussion

By implementing automated conversion tools, ISO standard generation of data models as ontology networks is possible and the benefits of ontology-based integration approaches can be taken advantage of. The CityGML standard is especially synergistic with this approach, as the XML Schema data model for XML/GML encoded data is also directly transformed from the UML model. Thus, class and property names from the UML model closely resemble their XML/XSD and RDF(S)/OWL representations with little semantic heterogeneity. The proposed transformation mappings from CityGML 3.0 to RDF knowledge graphs are also designed to limit creating structural heterogeneity during transformation while remaining interoperable with their original conceptual model. In addition, semantic graph formats like RDF align well with CityGML 3.0's graph-like representations of city models in the Versioning module. This can have many potential applications like change detection of city objects between versions of a city model which may rely on graph formats [NK20] and smart city applications based on Semantic Web technologies [Gau+15; Bis+14].

However, there are three modeling limitations of this approach to be addressed in future works. First and foremost, while geometry in the CityGML 3.0 standard is based off of the General Feature Model (GFM), there are several instances of semantic heterogeneity (problem **P1.a**, section 1.1) between the GFM and the XML encoding of the GML model. Similarly, several discrepancies in naming conventions between the GeoSPARQL ontology, ISO 19107 models, and GML 3.2 data were identified. For example, the application schema uses `gml:id` as a unique local identifier for geometric entities, while this does not exist in the GeoSPARQL ontology

and ISO 19107 uses the `featureID` attribute as its identifier. To solve this, custom transformation mappings are implemented during the data instance transformation workflow to avoid data loss during transformation.

Secondly, while UML models transformed by ISO 19150-2 may be 'platform-independent', these models may still need to be adjusted to facilitate a complete transformation. For example, the transformation of *association*, *aggregations*, and *compositions* roles requires these entities to be named to ensure they have an appropriate identifier in OWL. This is not an obligated or ubiquitous practice in UML modeling. [JOH19] notes several other modeling practices that may need to be considered when transforming UML models with this approach to ensure the creation of high quality ontologies.

Finally, as the generated CityGML 3.0 ontologies are constructed from UML's frame-based, *closed-world* assumption of the conceptual model, the mapping transformations to OWL's more *open-world* assumption of the conceptual model are subjective and require some interpretation [Bri+14; Cox13]. These ontologies are defined according to a more restrictive interpretation to guard as many constraints as possible. Because of this, it is possible that while this ontology network works well to generate standardized nD urban data knowledge graphs, other applications of UML to OWL mappings may require a more *open-world* interpretation depending on the purpose of the ontologies. It is not yet clear if a universally applicable UML to OWL transformation exists.

To improve the limitations of our proposed approach, future courses of action include exploring improving the existing semantic model and validation rules by integrating more urban data models like the Workspace CityGML extension from [SSG20] for modeling concurrent scenarios of urban evolution. Additionally, nD urban data applications will be implemented to demonstrate how the data generated by these transformation approaches can be exploited for providing users with unified views of city evolution.

6 Applications, standardization, and reproducibility contributions

This chapter presents several use cases that apply the methodology proposed in chapter 5 to integrate nD urban data. In particular, two nD urban data web applications are implemented to visualize integrated ontological data models and data and facilitating the spatial, temporal, and semantic navigation of these nD data as presented in sections 6.2 and 6.3. In addition, section 6.4 presents the technical contributions of this thesis and how the tools and transformations effectuated in this and previous chapters can be reproduced. Finally, section 6.5 presents several contributions to the CityGML 3.0 standard proposed during the course of this thesis.

To respond to the nD urban data integration problems detailed in section 1.1, this chapter presents the aforementioned contributions in section 1.3:

- **C2:** Formalization of data model extensions for navigating concurrent scenarios of urban evolution
- **C3.b:** Formalization of rules for validating scenarios of evolution of the urban landscape
- **C5.b**, **C5.c:** Production of nD urban datasets from real-world open data
- **C6:** Demonstration of nD urban integration applications for navigating scenarios of evolution
- **C7:** Reproducibility
- **C8:** Contributions to the CityGML 3.0 GML Encoding

This chapter presents work presented at the GeoDataDays 2022 Geodata challenges event[1] and subsequently published in the journals, Mappemonde [Col+23] as well as Transactions in GIS [Sam+23].

[1] https://www.geodatadays.fr/page/GeoDataDays-2022-Les-Challenges-Geodata/113

6.1 Software and technical architecture overview

As mentioned in section 1.2, this thesis takes place within the VCity research project of the LIRIS laboratory. The project explores scientific bottlenecks of visualizing and analyzing the evolving urban landscape and develops open-source tools within the UD-SV (Urban Data Services and Visualization) framework (conceived and built in VCity) [Sam+23] to provide 3D urban data analysis tools for researchers and industry. Several contributions to the UD-SV (Urban Data Services and Visualization) framework are made to exploit the data models and data created by the model-driven approaches proposed and presented in chapters 4 and 5 and provide users with integrated views of city evolution.

The UD-SV framework is composed of a set of modular software components for urban data management, computation, and analysis. These components are separated into a four-tier architecture as shown in figure 6.1. The *data server-tier* contains storage solutions and geospatial nD urban data sources interoperable with UD-SV components. The urban data in this tier is stored either as files on file servers or within 3D geospatial databases (such as 3DCityDB [Yao+18]). To provide a storage solution for nD urban knowledge graphs produced during this thesis, geospatial triple-stores (such as Strabon [KKK12] and Parliament[2]) are used in this tier.

The *processing server-tier* contains tools for manipulating, transforming, and cleaning 2D and 3D geospatial data. Notably, the Py3DTiles and Py3DTilers [Mar+22; JSG20; Col+22] are used to convert these data into data formats (like 3DTiles) for visualization on the web. The model-driven **UD-Graph** and ShapeChange transformation tools used to effectuate the transformations in chapters 4 and 5 are also located in this level. The *web server-tier* is not utilized in this chapter, but contains components for providing real-time geospatial data storage and data processing web services. The *client-tier* provides end-user services and applications for visualizing and navigating geospatial data such as UD-Viz—a JavaScript (JS) framework based on the geospatial iTowns library[3] [GDG22; JSG20; Jai+21; Sam+23]. A proposed extension to UD-Viz to enable SPARQL queries is discussed in the following section.

The components of the different tiers are organized into use-case driven data integration pipelines. An example of how these components can be orchestrated as such is presented in section 6.2 in figure 6.5.

[2] https://github.com/SemWebCentral/parliament
[3] https://www.itowns-project.org/

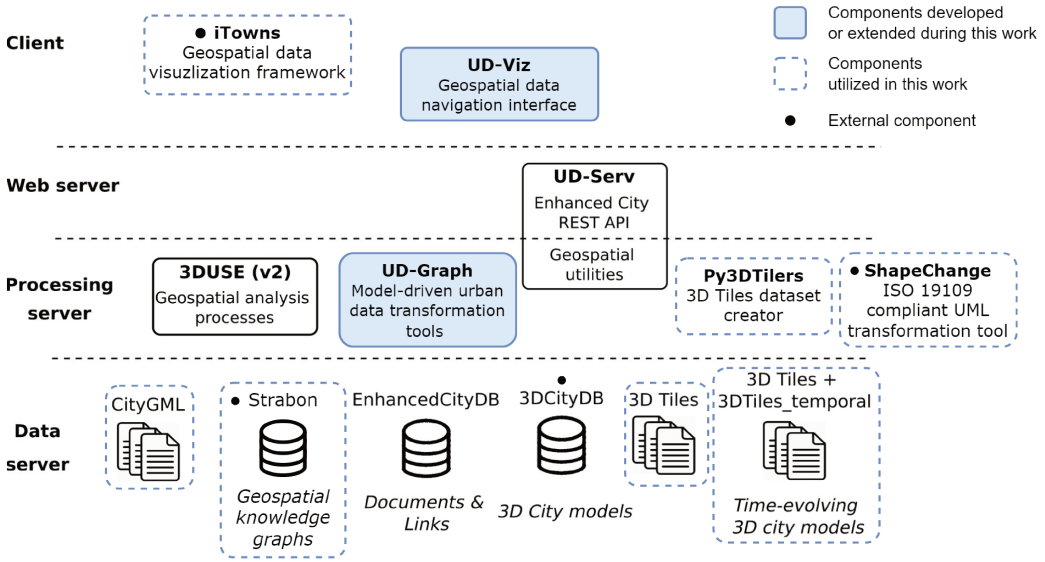

Figure 6.1.: Current software and 4-tier technical architecture of the UD-SV platform extended from [Sam+23]. Components developed or extended during this these are highlighted in blue. Other components or software used during this thesis are encircled in blue.

6.1.1 Knowledge graph visualization approach

As discussed in sections 1.2 and 3.1, 3D urban data has seen an increase in utility for creating applications to visualize and analyze the evolving urban landscape. Typically, 3D city models are visualized for this purpose through a *3D scene*. To provide integrated views of this data (and its underlying data models), we propose a two-part interface. This interface contains complementary views of the knowledge graph data (in RDF) stored in triple-stores alongside a view of a 3D scene. The goal of this interface, is to allow users to **query** the knowledge graph data and use the results to provide **semantic, temporal, and topological context** for urban features in a 3D scene (figure 6.2). In return, the 3D scene provides a **spatial context** for the data visualized in the view of the knowledge graph. These views also act as interfaces to **navigate** between, and **interact** with different representations of urban phenomena over time as will be demonstrated in sections 6.2 and 6.3. This interface is provided by extending the UD-Viz JS framework.

The interface for querying and viewing the knowledge graph data provides several methods for visualizing the results of a SPARQL query (figure 6.3 above). A **Table** visualization mode is provided with search and filter support. Next, a **JSON** mode provides a textual representation of the query result using the JSON data format. This mode is primarily intended for debugging purposes.

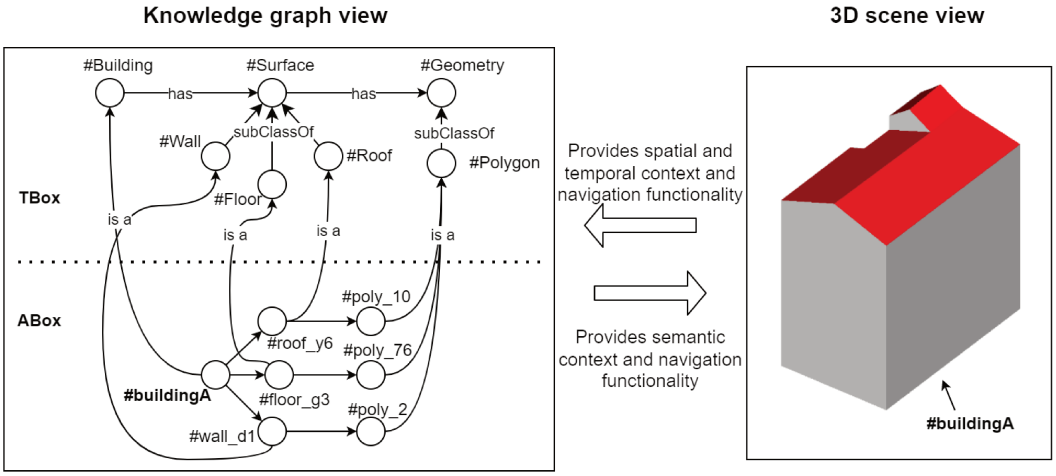

Figure 6.2.: An illustration of our proposed interface for integrated views of nD urban data. The left view provides information of some geospatial feature, 'buildingA', and its relation to other entities in the knowledge graph. The right view provides information of the locality and geometry of 'buildingA'.

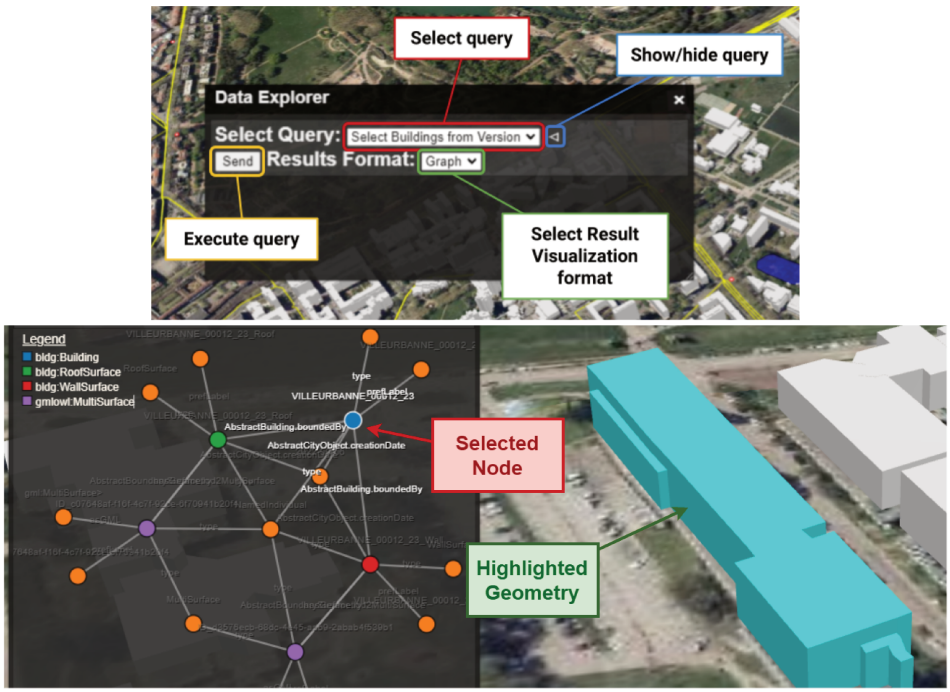

Figure 6.3.: The different controls of our proposed UD-Viz SPARQL extension user interface (above) and an example of a highlighting geometry in the 3D scene by hovering a mouse cursor over a corresponding feature in the graph visualization mode (below).

Finally, a **Graph** mode is provided to visualize interactive knowledge graphs using the D3.js[4] JavaScript library. Additional functionality was proposed to provide a spatial and semantic integration between the visualized knowledge graph and the 3D scene. For example, dragging a mouse cursor over a graph node that corresponds to a geospatial feature will highlight its corresponding geometry in the 3D scene (figure 6.3 below). Additionally, clicking on a node corresponding to a geospatial feature focuses the camera on its corresponding geometry in the 3D scene.

To detect when a node in the graph corresponds to a geometric feature, the identifier of the node and geometry must match. Both UD-Graph and Py3DTilers use the `gml:id` attribute of GML data as the identifier of geospatial features in their respective data transformations (as discussed in section 4.2.3). This allows entities in the graph to be linked to 3D geometries and locations in the 3D scene and vice-versa. In this application, the identifier of the graph node is represented by the URI fragment of the node. The identifier of the geospatial feature in a 3DTiles dataset is represented by the value of a key–value pair stored in the dataset, with the key `id`.

To visualize data in the graph mode, a 'CONSTRUCT-like' SPARQL query must be sent to the triple-store. As introduced in section 2.2, CONSTRUCT queries are used to return data as a graph instead of a tabular format. An example of a CONSTRUCT-*like* type of query is shown in listing 6.1. Here, a query must be structured such that the first 3 variables (`?subject`, `?predicate`, and `?object` in the example) correspond to RDF triples in the graph to be returned. UD-Viz uses these variables to reconstruct the graph to be displayed to the user. The optional variables, `?subjectType` and `?objectType`, (see lines 6, 12, 18) are used to determine the color of each node. Using the `UNION` operator, different graph patterns results can are stored in the same `?subject`, `?predicate`, and `?object` variables (see lines 8, 13) The implementation of these 3 visualization modes was supported by a class of masters students, in particular the Table and JSON visualization modes.

```
1  SELECT ?subject ?predicate ?object ?subjectType ?objectType
2  WHERE {
3    {
4      ?subject ?predicate ?object ;
5        a ?subjectType .
6      OPTIONAL { ?object a ?objectType }
7      FILTER (?subject == :building_A30)
8    } UNION {
9      :building_A30 bldg:AbstractBuilding.boundedBy ?subject .
10     ?subject ?predicate ?object ;
```

[4] https://d3js.org/

```
      a ?subjectType .
    OPTIONAL { ?object a ?objectType }
  } UNION {
    :building_A30
      bldg:AbstractBuilding.boundedBy/geo:hasGeometry ?subject .
    ?subject ?predicate ?object ;
      a ?subjectType .
    OPTIONAL { ?object a ?objectType }
  }
}
```

Listing 6.1: SPARQL query for constructing a CityGML Building matching an ID.

Section 6.2 demonstrates how this interface provides users with a means to navigate the **semantic** information integrated through this method. Section 6.3 demonstrates how this interface provides users with a means to navigate the **temporal** information integrated through this method.

6.2 Methodology use case 1: Integrating heterogeneous representations of La Chaufferie, La Doua Campus (2009-2018)

The first application of our proposed methodology is to document the evolution of 'la Chaufferie'—the former thermal plant of La Doua university campus of Villeurbanne, France—and the surrounding area between 2009 and 2018 on the web. The retired campus boiler house—an industrial heritage building—is given a new lease of life thanks to this web documentation project that integrates heterogeneous, multisource 3D models of la Chaufferie and the campus from different points in time alongside geolocalized multimedia images (figure 6.4).

The integration approach is shown in figure 6.5 which is divided into four parts: (1) input nD heterogeneous urban data, (2) data integration process, (3) integrated data visualization and navigation on the web (4) with various data representations. **Part one** highlights the 3 types of heterogeneous urban data to be integrated: 2D cartographic datasets, 3D building and city models based on Building Information Model (BIM) and City Information Model (CIM) data standards, and multimedia documents such as images—however this section focuses on the integration of the 3D models. An IFC model of the factory is used to represent the building at a high level of detail from the University of Lyon's cultural heritage archival service. This BIM

Figure 6.4.: The implemented nD urban data web application. The 3D scene contains a 3D model of the Chaufferie with a geospatial 'pinned' multimedia image of the factory overlaid using the technique proposed in [GDG22].

(IFC) model is integrated alongside four 3D city snapshots of the entire university campus (from the years 2009[5], 2012[6], 2015[7], and 2018[8]) showing how the factory and the surrounding buildings have changed between the available years. These 3D city model snapshots or vintages are periodically published (often triennially) by the Metropole of Grand Lyon[9] for each commune within the greater metropolitan territory as CityGML data.

Part two of the integration approach contains the data processing (data cleaning and transformation), data storage, and data access methods that are effectuated on these urban data to facilitate their integration (figure 6.5). In order to facilitate the **spatial** (geospatial and geometric) and **semantic** integration of these 3D data, these data are transformed to data formats designed for storing and sharing 3D geospatial and semantic data on the web, such as 3DTiles[10] and OWL. Spatial transformations from these data towards the 3DTiles format are effectuated using Py3DTilers as proposed in the previous section. Likewise, semantic transformations are effectuated using

[5] https://download.data.grandlyon.com/files/grandlyon/imagerie/2018/maquette/VILLEURBANNE_2009.zip
[6] https://download.data.grandlyon.com/files/grandlyon/imagerie/2018/maquette/VILLEURBANNE_2012.zip
[7] https://download.data.grandlyon.com/files/grandlyon/imagerie/2018/maquette/VILLEURBANNE_2015.zip
[8] https://download.data.grandlyon.com/files/grandlyon/imagerie/2018/maquette/VILLEURBANNE_2018.zip
[9] https://data.grandlyon.com/
[10] https://www.ogc.org/standard/3DTiles/

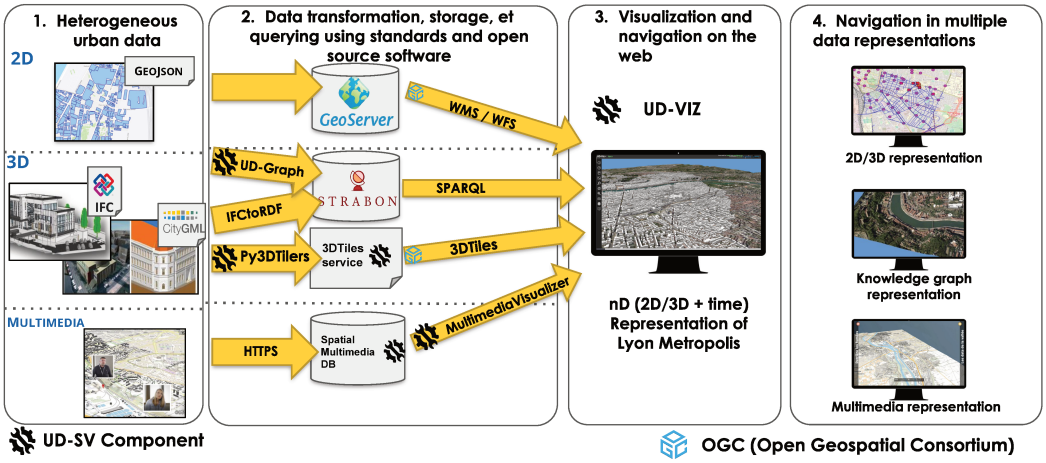

Figure 6.5.: The heterogeneous nD urban data integration approach translated and updated from [Col+23].

UD-Graph (and the integration methodology proposed in chapter 5) and in the case of the IFC data, standard-specific transformation tools such as IFCtoRDF[11] [PV12]. These transformations must be effectuated such that the original identifiers of the input data are preserved so that these data can be referenced in either their spatial or semantic output formats [Col+22].

Part three provides users a means to visualize and interact with the integrated data through an nD urban data web application using the extension proposed in the previous section. **Part four** illustrates the 3 types of urban data representations provided by UD-Viz for this purpose: 2D/3D digital models, *knowledge graph*, and multimedia urban data representations. **In this implementation, navigation of the semantic data is provided by the knowledge graph data representation.** Temporal navigation is provided by the temporal 3DTiles extension proposed in [JSG20].

Figures 6.6 and 6.7 show the knowledge graph representation of la Chaufferie (left) within the 3D scene containing geolocated 3D model of the same building (right). These figures visualize the 3D models and knowledge graphs of la Chaufferie at different levels of detail. Figure 6.6 showcases the factory at a CIM level of detail, visualizing a less detailed 3D model and a simpler graph containing all the available semantic and geometric data on the building from the CityGML dataset in 2012.

Figure 6.7 showcases a more detailed representation of the factory at a BIM level of detail, visualizing a more detailed 3D model from the IFC dataset and a graph of all instances of the `ifc:IfcSlab` class—i.e., horizontal construction components

[11] https://github.com/pipauwel/IFCtoRDF

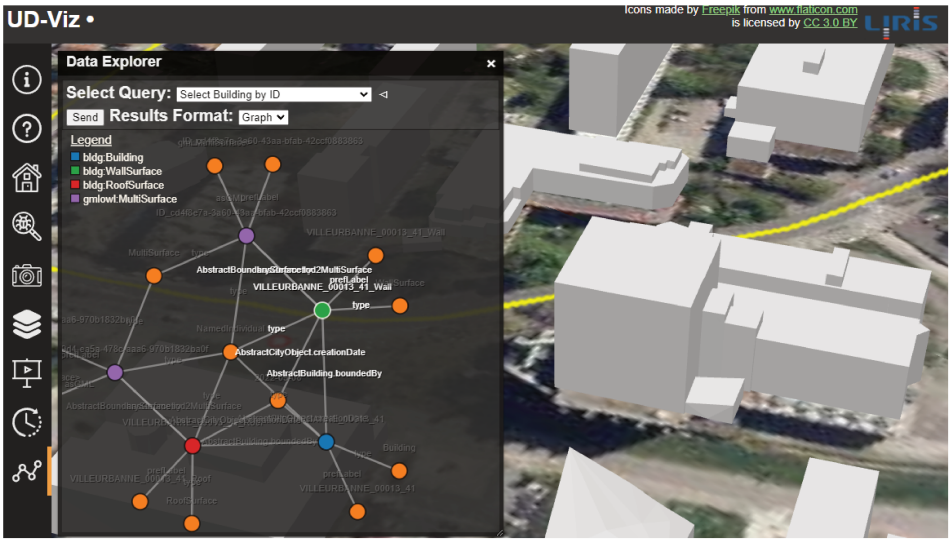

Figure 6.6.: The result of a graph query of all thematic surfaces (i.e., wall, roof, and ground surfaces) of the CityGML model of la Chaufferie from 2012.

such as floors from the IFC standard. The following section discusses how these components are utilized to improve the contextualization of time in nD urban data applications, and notably, improve the extension to the UD-Viz component for visualizing concurrent points of view of urban evolution in 3D virtual scenes.

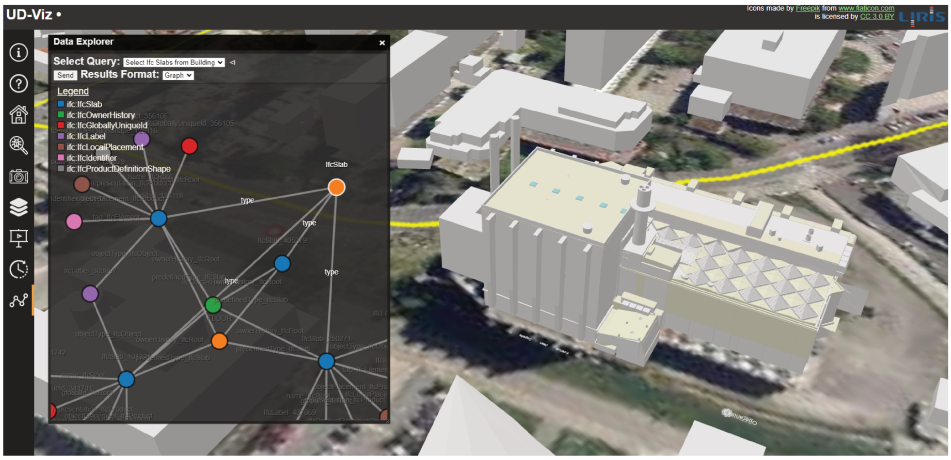

Figure 6.7.: The result of a graph query of all instances of the class ifc:IfcSlab (i.e., horizontal construction components from the IFC standard) and their relationships of la Chaufferie (below).

6.3 Methodology use case 2: Integrating concurrent scenarios of evolution of the Gratte-Ciel neighborhood of Villeurbanne (2009-2018)

This section presents the results produced by applying this model-driven methodology to a real-world urban planning data integration use-case: Modeling the evolution of the Gratte-Ciel neighborhood of Villeurbanne, France from 2009 to 2018 from two concurrent perspectives. For context, the triennially published vintages of the Metropole of Lyon may contain certain "remarkable" or noteworthy buildings (such as town halls or cultural heritage sites). These remarkable features are occasionally remodeled in detail and stored in a separate 3D model, accompanying the main vintage. At the heart of the Villeurbanne commune, are several skyscrapers (gratte-ciels) constructed between 1927 and 1934 as a part of an urban housing project [Gal05]. These buildings, along with the commune's town hall, have been digitally remodeled in this way. The two vintages from 2009 and 2012 include two 3D models each, one for the main vintage excluding the remarkable buildings and another including only the remarkable buildings. The vintages from 2015 and 2018 instead include a complete vintage including the remarkable buildings and a separate 3D model for the remarkable buildings at a higher level of detail. Depending on whether the "remarkable" buildings are included in a 3D model of the city during certain years, different versions of these vintages can be constructed for the same instant of time.

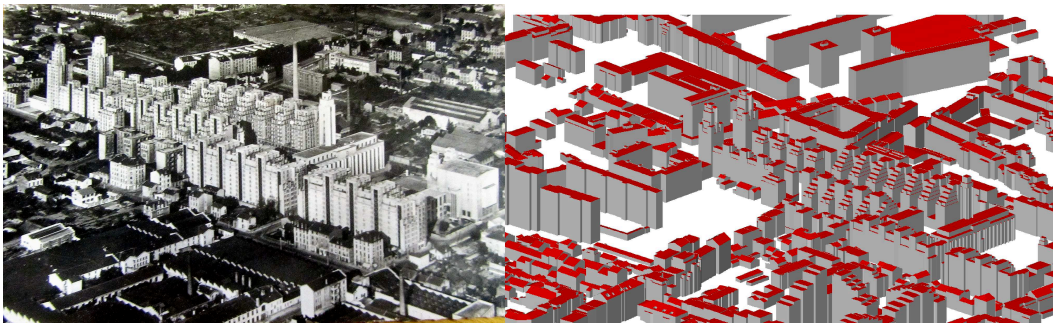

Figure 6.8.: An aerial photo of the Gratte-Ciel neighborhood from 1936 [Com20] (left) and from 2018 as a 3D city model (right)

The resulting integrated data model and rules used to integrate and validate these vintages is presented below. In addition, an urban data web application is proposed to facilitate the navigation and comparison of the integrated vintages.

6.3.1 Integrating workspaces and vocabularies for modeling urban evolution

To provide the data structures and vocabularies necessary to represent concurrent viewpoints of urban evolution, the Workspace CityGML ADE proposed by Samuel et al. [SSG20] (section 2.3) is integrated using our proposed model-driven methodology from section 5.1. This is done by recreating the original Workspace UML model as an extension of the CityGML 2.0 model and updating the Workspace UML model to conform to CityGML 3.0. Excerpts of these UML models are provided in appendices A.1 and A.2[12]. Both of these UML models are transformed to OWL ontologies that align with the generated CityGML 2.0 and 3.0 OWL ontology networks. For example, figure 6.9 illustrates the correspondences between the generated Workspace ontology and the CityGML 3.0 Versioning ontology.

In addition to integrating the Workspace model, an extension to the CityGML 3.0 vocabulary for transaction types is defined to enable representing transaction types according to the vocabulary proposed in [Jai+21]. Figure 6.10 illustrates this addition as an SKOS concept scheme. Here, two new transaction types are added: *union* and *division*. These types are analogous to the *merging* and *splitting* proposed in [Ren00] which denote one-to-many and many-to-one transactions between geospatial features, as discussed in section 2.3.

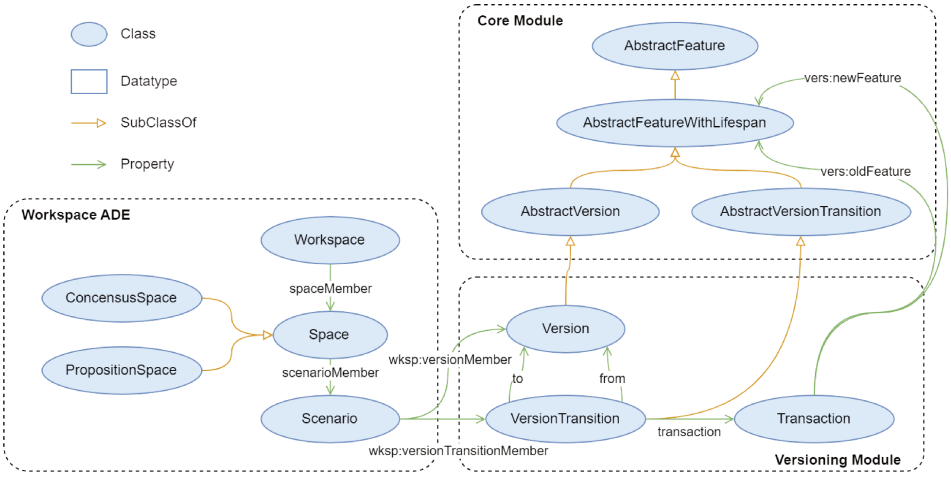

Figure 6.9.: The classes and properties of the integrated Workspace ontology aligned with the CityGML 3.0 Versioning ontology. Properties are labeled with globally scoped naming conventions for readability (a local scope is used in the actual network).

[12]This appendix is of a proposed CityGML ADE for *documenting* urban evolution, which was transformed but not utilized in this application use case.

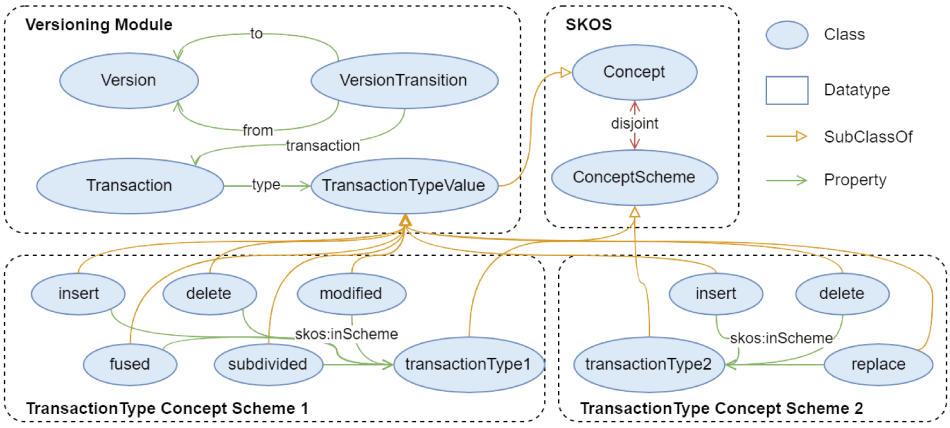

Figure 6.10.: The classes and properties of the integrated ontology network from CityGML and the proposed SKOS concept schema for modeling Transaction types. Concept schema 1 defines the vocabulary proposed in Jaillot et al. [Jai+21] and concept scheme 2 defines the vocabulary from CityGML 3.0.

6.3.2 Integrated data

The Gratte-Ciel vintages were extracted from 4 CityGML datasets of the Villeurbanne commune from the years 2009-2018. Using the CityGML relational database 3DCityDB [Yao+18] and the Py3DTilers transformation tool [Mar+22], a predefined region of interest was extracted from each vintage and transformed into the geospatial 3D web data format 3DTiles[13]. The approach in [PMG15; JSG20] was used to produce graphs of changes between the city objects of each vintage. This approach compares the 2D footprints of geospatial features between city vintages to determine if the identity of the feature has changed, and if so, what type of change occurred. Each change is classified by the transaction types introduced in section 6.3.1 are used (creation, demolition, modification, union, and division).

To enable modeling the evolution of the region using CityGML 3.0, the vintages were transformed from CityGML 2.0 using the citygml2-to-citygml3 transformation tool[14] developed by the Technical university of Munich and then transformed from XML to RDF using UD-Graph [Vin+21b]. A new workspace was instantiated in Protégé ontology editor[15] by adding each vintage to a unique version, the changes between each vintage to corresponding version transitions. Additionally, two chains of versions and version transitions were declared as scenarios of evolution of the Gratte-Ciel neighborhood as illustrated in figure 6.11; a scenario composed of vintages with a lower level of detail in the consensus space and another scenario

[13] https://www.ogc.org/standard/3dtiles/
[14] https://github.com/tum-gis/citygml2-to-citygml3
[15] https://protege.stanford.edu/

representing the evolution of the neighborhood in a higher level of detail within the proposition space of the workspace.

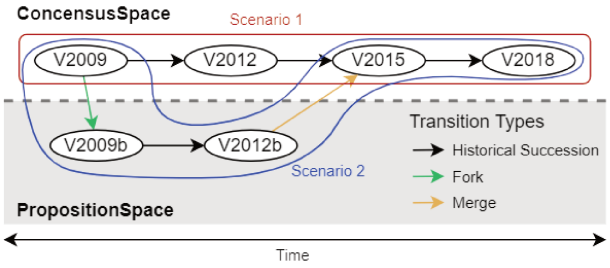

Figure 6.11.: Abstract diagram of Gratte-Ciel workspace. The workspace is composed of 2 scenarios of evolution, 6 versions, and 6 version transitions from 2009 to 2018.

As each vintage is published without temporal information, CityGML 3.0 timestamps were added to each city object, corresponding version, and version transition through a script according to the year the vintage was published. These timestamps were enriched with OWL-Time temporal entities through the approach described in section 5.3.3.

6.3.3 Rule-based validation of concurrent scenarios of evolution

To validate whether a proposed evolution scenario (and the Workspace as a whole) is temporally logical we use the rules proposed in Samuel et al. [SSG20]. These rules, originally formalized in a description logic (DL) language, have been formalized in SWRL along with the proposed generic rules for inferring and validating temporal entities from OWL-Time from section 5.2.4.

For example, the Workspace DL rule introduced in section 3.3 stating that *the existence time of a version is implicitly contained within the existence time of every one of its member features* can be rewritten as:

```
vers:featureMember(?v, ?f),
  time_ext:hasExistenceTime(?v, ?tmv),
  time_ext:hasExistenceTime(?f, ?tmf)
-> time:in(?tmv, ?tmf)
```

In conjunction with the generic rule stating that *any two temporal entities that are temporally in one another and disjoint are inconsistent*:

```
time:in(?t1, ?t2), time:disjoint(?t1, ?t2),
-> owl:Nothing(?t1), owl:Nothing(?t2)
```

Any version that exists temporally outside its feature members would be inferred as inconsistent. A complete list of the proposed rules is referenced in appendix C.3. A proof of concept test suite was developed within the UD-Graph component for validating integrated datasets according to these rules in Python using RDFLib and OwlReady2. This suite reads in a given set of ontologies and a set of SWRL rules, constructs an ontology network, and through reasoning can infer if the ontology network is consistent and new information based on the rules and constraints of the network.

The integrated data instances are validated as logically and temporally consistent with the TBox of the ontology network using the proposed rules and the test suite. This validation step is important since—as discussed in section 3.3—this helps assure that information has not been lost during transformation and helps maintain a higher quality of data produced for nD urban data applications. The following section discusses how this data can be queried to retrieve concurrent scenarios of urban evolution.

6.3.4 Workspace queries and data views

4D and nD urban data applications for facilitating the decision-making process or understanding city evolution require methods for answering questions about changes to the urban landscape, hypothetical or otherwise. Using queries over the integrated ontology network, these questions can be answered. For example, natural language questions such as "what types of features exist within a specific version?" or "how many features were 'unofficially' modified between 2009 and 2012?"

Figure 6.12 provides a use case diagram of these queries and illustrates which versions in the workspace each query would access. The first question in the figure can be effectuated through the SPARQL query proposed in listing 6.2[16] and would return the features contained within version 2018 of the workspace. This query would also return the types of these features according to their class (i.e., building, bridge, vegetation, etc.). The second question is more complex as it contains temporal and semantic components, i.e., 2009–2012 and the meaning of the word "unofficially". The query proposed in listing 6.3 could answer this question. The temporal component of the question can be resolved using a filter on

[16]Local naming conventions are used in query listings 6.2 and 6.3.

which transitions need to be considered (line 13). The semantic component can be resolved by considering only scenarios of evolution that occur in proposition space, i.e., hypothetical or unproven scenarios of evolution (line 3).

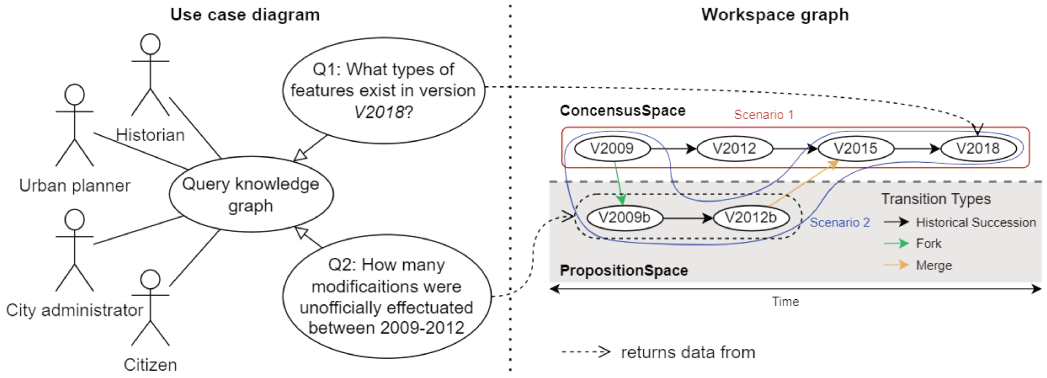

Figure 6.12.: A UML use case diagram (left) illustrating the possible users and potential queries they may effectuate on the integrated workspace data and which versions in the workspace are impacted by each query (right).

```
1  SELECT ?feature ?featureType
2  WHERE {
3    :version_2018 a vers:Version ;
4      vers:Version.versionMember ?feature .
5    ?feature a ?featureType .
6  }
```

Listing 6.2: A SPARQL query for returning all CityGML features (with types) within a version.

```
1  SELECT (COUNT(?transaction) AS ?NumOfModifications)
2  WHERE {
3    ?concensusSpace a wksp:PropositionSpace ;
4      wksp:Space.scenarioMember ?scenario .
5    ?scenario wksp:Scenario.versionTransitionMember
6        ?versionTransition .
7    ?versionTransition vers:VersionTransition.transaction
8        ?transaction ;
9      core:AbstractFeatureWithLifespan.validFrom ?from ;
10     core:AbstractFeatureWithLifespan.validTo   ?to .
11   ?transaction vers:Transaction.type type:modified .
12   FILTER( "2009-01-01T00:00:00"^^xsd:dateTime < ?from )
13   FILTER( "2012-01-01T00:00:00"^^xsd:dateTime > ?to )
14 }
```

Listing 6.3: A SPARQL query counting the number of modified transactions in proposition space occurring during an interval of time.

Based on the web application architecture presented in section 6.1, queries could also be used to create data views that can contextualize the state of a city or district amongst past, future, and concurrent states using the proposed UD-Viz extension. Listing 6.4 defines a query that collects the *Versions* and *VersionTransitions* of a Scenario using the aforementioned CONSTRUCT-like graph patterns. Figure 6.13 illustrates the resulting knowledge graph visualized by this query when executed over the integrated Gratte-ciel data. In this figure, the graph view provides context for the scenario of evolution and Version or VersionTransition is being shown in the 3D scene. Here, the transition between the two versions is visualized in the 3D scene using the approach proposed by Jaillot et al. [JSG20]. In this approach, the creation, demolition, modification, union, and division (section 6.3.1) transactions of a VersionTransition are highlighted in red, yellow, green, blue, and orange respectively. For this application, functionality was added to the graph visualization mode to permit temporal navigation of the workspace based on [JSG20]. Selecting a node that corresponds to Version or VersionTransition will visualize the corresponding snapshot or transition between snapshots in the 3D scene.

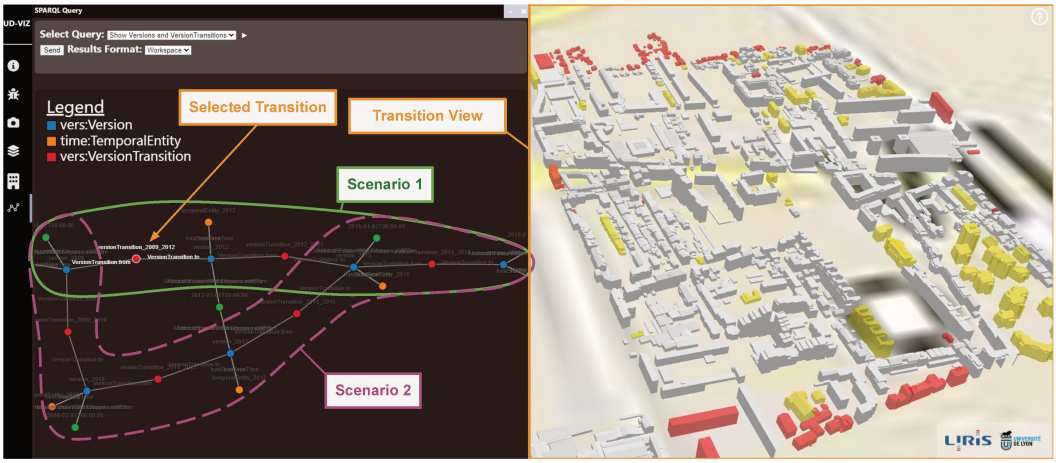

Figure 6.13.: A visualization of the transition between the years 2009 and 2012 of the Gratte-ciel neighborhood. Our proposed knowledge graph view (left) shows the Workspace including both proposed scenarios of evolution. These selected node corresponds to the transition which is being visualized in the 3D scene (right).

```
SELECT DISTINCT ?subject ?predicate ?object ?subjectType ?objectType
WHERE {
  {
    ?subject a wksp:Scenario ;
      a ?subjectType ;
      ?predicate ?object .
    OPTIONAL {
      ?object a ?objectType .
```

```
    }
  } UNION {
    ?subject a vers:VersionTransition ;
      a ?subjectType ;
      ?predicate ?object .
    OPTIONAL {
      ?object a ?objectType .
    }
  } UNION {
    ?subject a vers:Version ;
      a ?subjectType ;
      ?predicate ?object .
    OPTIONAL {
      ?object a ?objectType .
    }
  }
}
```

Listing 6.4: SPARQL query for constructing the graph of scenarios of evolution using the Workspace model.

Additionally, the generated graph is *force-directed*, i.e., nodes are repelled from each other and nodes linked together are attracted using a physics simulation. This allows for automated clustering of nodes with many similar neighbors in the graph which is useful for distinguishing different transaction types within a VersionTransition. Figure 6.14 shows our proposed generated graph of the VersionTransition between 2009 and 2012 of the Gratte-ciel neighborhood. The large clusters of orange nodes corresponds to the buildings of each Version and the blue nodes in the center correspond to the transactions of the VersionTransition.

6.4 Reproducibility approach

A large effort was made during the course of this thesis to deliver the aforementioned contributions in reproducible, interoperable, and accessible manners. Shell, and Python scripts are implemented to provide reproducible transformation workflows and nD urban data web applications. These workflows are stored in a repository intitled **UD-Reproducibility** (table 6.1, contribution 4). Additionally, contributions are supplemented with technical documentation, user guides, installation instructions, and documentation of technical issues whenever possible. Software archiving services such as Software Heritage[17] are used to provide links towards these con-

[17] https://www.softwareheritage.org/

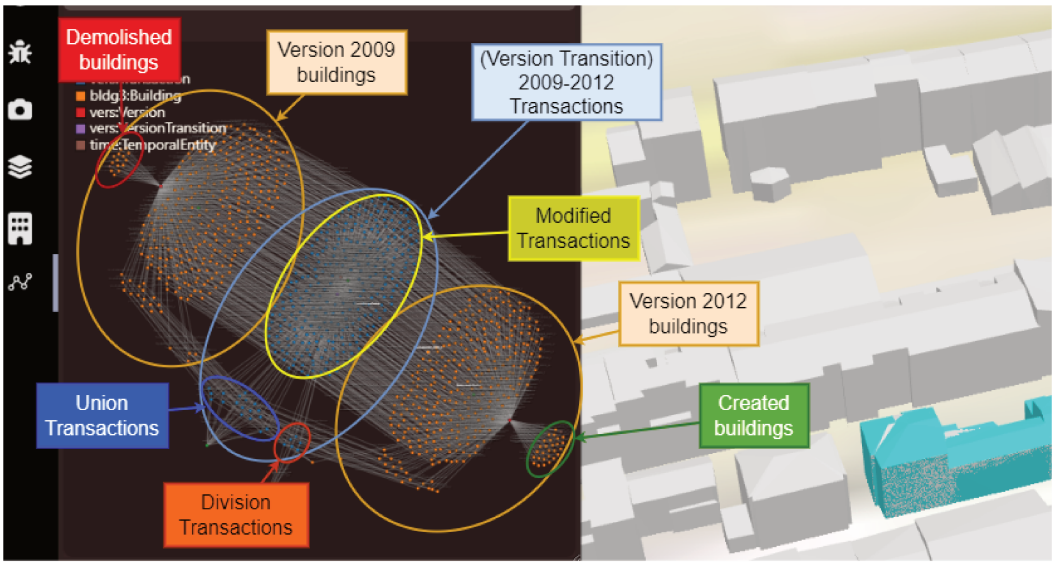

Figure 6.14.: The generated graph of the VersionTransition between 2009 and 2012 of the Gratte-ciel neighborhood in detail. The identified force-directed clusters are highlighted and labeled.

tributions (table 6.1). In addition, a *Free and Open Software* (FOSS) approach was taken to produce all materialized and technical contributions (code, documentation, data models, and data). FOSS has been promoted since the early 2000s as an approach to providing readily available and easily extensible software to international communities [Whe07]. Thus, these contributions are made available under an open LGPL-2 license[18].

Notably—thanks to a longstanding collaboration between the LIRIS laboratory and the Metropole of Lyon—the transformed datasets for the Gratte-ciel neighborhood (table 6.1, contribution 3) produced in section 6.3 are hosted on `https://data.grandlyon.com`, Metropole of Lyon's open data website as shown in figure 6.15.

	Technical contribution	Software Heritage ID or URL
1.	nD urban data models	`swh:1:dir:8f07f77f3ed973f04919cc4af3b2762d20375b95`
2.	nD urban datasets	`swh:1:dir:1e0ccbd69bf757000bd1f461ac1ddb50bf199832`
3.	2009-2018 Gratte-Ciel Data	`https://beta.recette.data.grandlyon.com/portail/fr/jeux-de-donnees/maquette-oriente-graphe-du-quartier-gratte-ciel-a-villeurbanne`

[18] `https://www.gnu.org/licenses/lgpl-3.0.html`

4.	UD-Reproducibility	https://github.com/VCityTeam/UD-Reproducibility/tree/master/Computations/RDF
5.	UML Models	swh:1:dir:5bc538c2fabb300cf3cf6c4750aecb1e1471d9e2
6.	ShapeChange configuration files	swh:1:dir:28316a8fcb9f4d47ca4b7e4217af86d0793dcbd8
7.	UD-Graph (transformation components)	swh:1:dir:2fce25376177276e02d853250fba33db83f2c4a0
8.	UD-Graph (SWRL Rule Test Suite)	swh:1:dir:0f251971359e155a07d6a67ab964d6c7f4494d3d
9.	Proposed SKOS Concept Schema	swh:1:cnt:62deff5243eb92b8807099b81ac84cf27ed9a0fb;path=/
10.	Proposed SWRL Ruleset	swh:1:cnt:86a3d1a823655c0f960ffad4331111482b075cbe;path=/
11.	SWRL Rule Test Suite Configuration	swh:1:cnt:769953472c5d23e534f45f88a520ec0924bf5308;path=/
12.	nD urban data application (UD-Viz+Strabon)	swh:1:dir:bb2858ac2c38b5a9797851ca78ce02b1c6248e81

Table 6.1.: Software Heritage identifiers and URLs of transformation tools, configuration files, data models, and datasets discussed in this dissertation. Software Heritage IDs begin with 'swh' and can be used at https://archive.softwareheritage.org/ to view the archived resources.

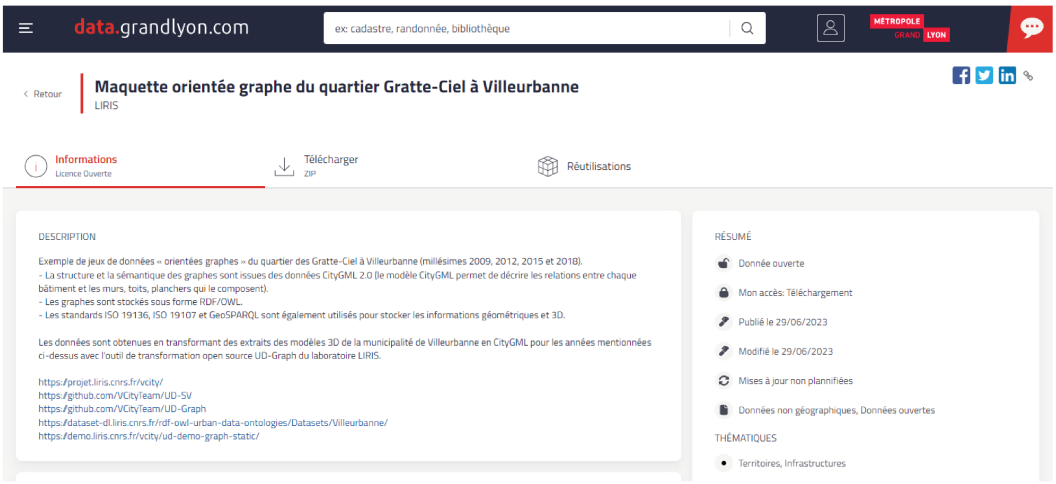

Figure 6.15.: A screenshot of the data repository for the Gratte-ciel 2009-2018 dataset, available on the metropole of Lyon's open data website.

To supplement aforementioned reproducibility approach, virtualization technologies (such as Docker[19]) were utilized to permit the developed programs and applications to be installed in portable 'containers' that can be run on a variety of operating systems and architectures without requiring changes to dependencies or libraries. Components of transformation workflows (i.e., *Shapechange*, *UD-Graph*, and *Py3DTilers*) were placed into these containers to be executed in succession on the same input and output datasets. Components for data storage and data visualization (i.e., the *Strabon* triple store, *UD-Viz*, and the *3DTiles server*) were also dockerized to facilitate web deployment. For example, the architecture illustrated in figure 6.1 was implemented during the second use case (section 6.3) to support storing, accessing, and visualizing the materialized (transformed) nD urban data. A 3DTiles server and Strabon are used to store the spatial and semantic data and form the 'backend' services of the application. The client facing 'frontend' service is provided by UD-Viz for providing users a view of the integrated data and means to navigate between representations of urban phenomena in space and time.

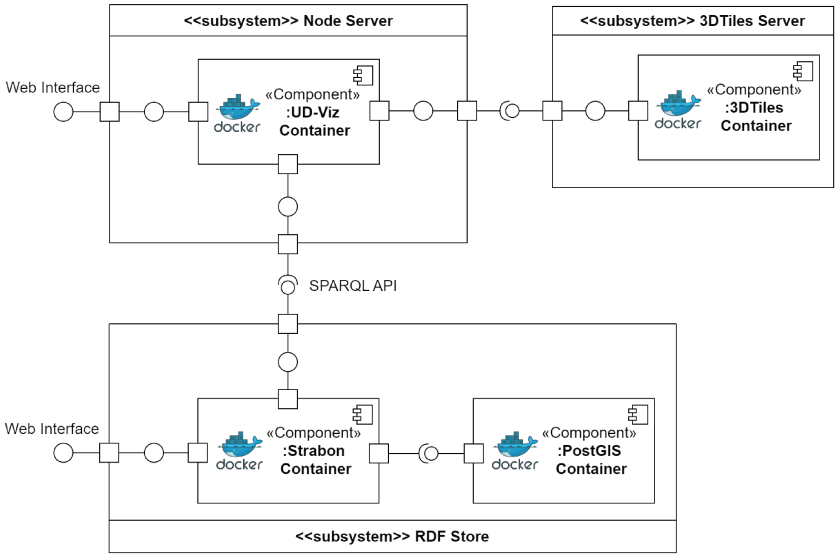

Figure 6.16.: A UML component diagram of the proposed nD urban data web application architecture implemented in the previous section.

6.5 Contributions to the CityGML 3.0 GML Encoding

LIRIS has been a member of OGC for over 10 years. Researchers working on the VCity project have participated in the implementation of standards such as CityGML

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

2.0 and CityGML 3.0. The team's earlier work, in collaboration with TUM [Cha+17], also led to the temporal (versioning) part of CityGML 3.0. During my thesis, I took part in the GML encoding of CityGML 3.0 on the behalf of LIRIS.

In addition to the aforementioned contributions, several contributions to the OGC's CityGML SWG (Standards Working Group) are made as a part of this thesis. The purpose of the CityGML SWG is to propose revisions and improvements to the CityGML data standard. Since version 3.0 of the CityGML standard was released in 2020 [KCK20], creating a GML encoding of the CityGML 3.0 Conceptual Model was one of the primary tasks of the SWG.

To realize this task, meetings were held with the other members of the SWG to discuss how different implementations of a CityGML GML encoding may affect future users of the standard. Determining these impacts requires taking into consideration the potential general and technical applications of the standard. For example, how can the GML standard be used to represent UML and GFM concepts (section 2.1.2) like `CodeLists`[20]. Or for instance, what restrictions should be implemented on how CityGML GML data can be structured to simplify the task of developers for creating CityGML GML parsers, while still allowing users to create CityGML data in creative ways that fits their needs[21].

In addition to participating in the meetings of the CityGML SWG, editorial contributions were made to the *CityGML 3.0 GML Encoding Specification*[22] [Kut+23]. This document defines the GML encoding of CityGML 3.0 including *requirements* and *abstract tests* for conformance to the standard and *mappings* of CityGML concepts from the conceptual model (UML classes) to types in the defined CityGML 3.0 XML Schema (XSD complex types).

Furthermore, **several example CityGML 3.0 datasets were created to supplement the encoding specification** while participating in the CityGML SWG. These examples help guide users in creating their own CityGML documents while meeting the conformance requirements of the standard. As a newly proposed data standard, little existing CityGML 3.0 data is available and urban data producers may rely on these examples to when creating new datasets or developing tools for parsing, visualizing, and analyzing CityGML 3.0 data.

Firstly, an example of how versioned CityGML 3.0 data can be structured using Versions and VersionTransitions was proposed during this thesis that was included in

[20] https://github.com/opengeospatial/CityGML-3.0Encodings/issues/56
[21] https://github.com/opengeospatial/CityGML-3.0Encodings/issues/6
[22] As of the submission of this dissertation, the encoding specification is still in an unofficial draft stage, awaiting final approval as an OGC standard.

the encoding specification (section **B.3.2** of the encoding document) (figure 6.17). This section was supplemented with example versioned CityGML 3.0 data of a fictive historical succession[23] based on CityGML 2.0 data also from the metropole of Lyon. This example is introduced in section 5.3.3 and transformed to a knowledge graph using the approach proposed in section 5.2.3.

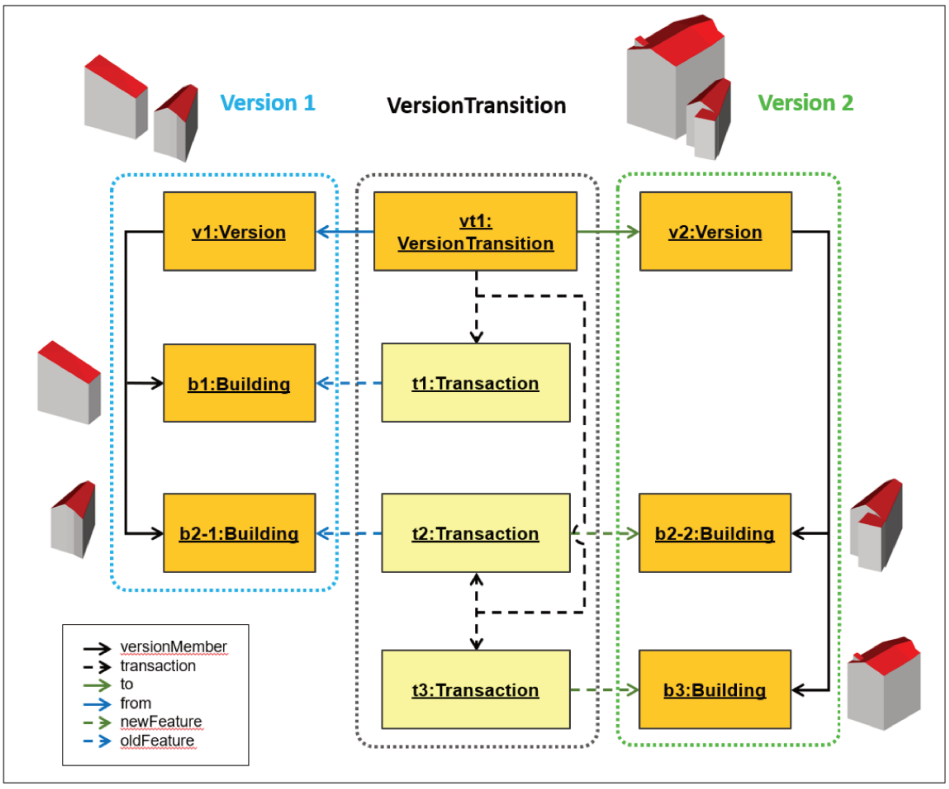

Figure 6.17.: A *"historical Succession"* illustrated using Versions and VersionTransitions from the CityGML 3.0 Versioning module from [Kut+23].

Secondly, several examples were produced for illustrating how the xAL addressing standard can be used to create structured and semi-structured address information for 3D building models[24,25,26]. Here, three address examples were provided using a highly detailed 3D model of the National Opera House of Lyon, France from 2018[27]

[23] https://github.com/opengeospatial/CityGML3.0-GML-Encoding/blob/69d6d05ea2dee19082df570fea79551ae6267eb7/resources/examples/Versioning/Basic%20examples/historicalSuccession_CityGML_3.0_LOD2_Versioning.gml

[24] https://github.com/opengeospatial/CityGML3.0-GML-Encoding/blob/main/resources/examples/Building/Opera-Lyon_CityGML3.0_LOD2_with_xAL3_FreeText.gml

[25] https://github.com/opengeospatial/CityGML3.0-GML-Encoding/blob/main/resources/examples/Building/Opera-Lyon_CityGML3.0_LOD2_with_xAL3_FullyStructured.gml

[26] https://github.com/opengeospatial/CityGML3.0-GML-Encoding/blob/main/resources/examples/Building/JeffersonBuilding_CityGML3.0_LOD1_with_xAL3_CommonTypes.gml

[27] https://data.grandlyon.com/jeux-de-donnees/maquettes-3d-texturees-2018-communes-metropole-lyon/donnees

and a low level of detail 3D model of the Thomas Jefferson building in Washington D.C., USA[28].

To create the CityGML 3.0 datasets, existing CityGML 2.0 datasets were transformed using the CityGML2-to-CityGML3 transformation tool[29] proposed by the University of Munich.

6.6 Discussion

The use case driven applications of the methodology delivered in this chapter demonstrate the effectiveness of our reproducible, standards-based, model-driven methodology proposed in chapter 5. In addition, several contributions were produced in collaboration with stakeholder organizations such as Open Geospatial Consortium and the Metropole of Lyon.

During the implementation of the proposed applications (sections 6.2 and 6.2), several geospatial triple stores were tested. Initially Parliament was used since it supports geospatial queries through GeoSPARQL. However, Strabon was eventually chosen as it supports similar geospatial and temporal queries through stSPARQL and has been shown to perform well compared to Parliament [GKK13]. However, as mentioned in section 4.4, these spatial query extensions to SPARQL only support 2D spatial queries. To avoid flattening the 3D geometry of the integrated CityGML to 2D, the querying of 3D data was effectuated with UD-Viz and iTowns on the 3D Tiles representation of the integrated. Additionally, other triple stores such as Blazegraph have been shown to be even more performant in use cases where complex geometry queries are not required [Li+22]. In future works, a more mature 3D geospatial knowledge graph query framework could be developed similar to the approach proposed in [ZBV18].

Chapter 7 further discusses potential future work, other identified future works, and synthesizes the proposed contributions of this thesis, concluding this dissertation.

[28] https://github.com/opencitymodel/opencitymodel
[29] https://github.com/tum-gis/citygml2-to-citygml3

Part III

Conclusion

7 Contributions and perspectives

There is a need for researchers and city planners to be able to combine urban data sources to study the evolution of the urban landscape. This analysis often requires the integration of data of various dimensions (nD) and from multiple standards facilitating representations of the evolving urban lifecycle. To this end, the work effectuated over the course of this thesis has the goal of improving the urban data integration process for providing users with views of city evolution.

7.1 Summary of contributions

To meet this goal, the research was guided by three principal research questions:

First, *how can nD urban data integration approaches ensure that urban data standards can be easily reused, even as these standards evolve (**RQ1**)?* In answering this question, it is observed that as data models are notably proposed to define the structure of data, these models provide a basis for interoperability. In the context of data standards, these models are updated as standards evolve. Taking advantage of this propensity, a **standards-based model-driven methodology for integrating nD urban data** is proposed where the data models underlying heterogeneous urban data sources are used to facilitate the integration process and **preserve interoperability during transformation**. To effectuate model-driven transformation workflows, an open source transformation tool, **UD-Graph** [Vin+20], is proposed (cf. contribution **C4**, stipulated in section 1.3) in conjunction with existing tools such as ShapeChange. These workflows propose transformation from data models and data instances to *ontological languages* and *knowledge graph formats*. These languages and formats are targeted to **improve reusability** by integrating the transformed data models and data as machine-readable linked data and share these results on the web using Semantic Web technologies. The proposed transformation workflows are effectuated on the evolving CityGML standard, extensions to CityGML, and real-world urban data **to formalize evolving nD urban data ontological networks** [Vin+21b; Vin+21a] (cf. contributions **C1** and **C2**) and **to produce nD urban datasets of evolving**

city areas (cf. contribution **C5**) [Col+23]. The proposed integration approach also provides means to reuse existing spatial, temporal, and semantic ontological standards to improve the interoperability of the resulting transformations.

In addition, *how can data loss be limited when transforming data between heterogeneous nD data formats (**RQ2**)?* To answer this question, **iterative validation steps are put forth during the proposed methodology**. These validation steps utilize the *constraints* and *rules* proposed as part of data models. During the integration of data models as computational ontologies, *consistency checking* is applied with reasoners to the products of the proposed transformations to ensure they are logically consistent. Through reasoning, the products of data instance transformation can be verified as conformant to the ontological models as well. Additionally, formal rules are formalized in a machine-readable format (SWRL) to verify the consistency of integrated nD urban data for providing concurrent views of urban evolution (cf. contribution **C3**). Also, **a proof of concept test suite is implemented** as a part of UD-Graph to facilitate reproducible testing based on the formalized data model constraints and rules.

Also, *how are model-driven transformation approaches affected by data models at different levels of abstraction (**RQ3**)?* Model-driven integration approaches in the geospatial domain have posed this question as it is observed that transforming data models from different languages, formats, and levels of abstraction may produce varying results. To answer this question, the proposed model-driven approach is effectuated, on more abstract **platform-independent data models** (PIM) [Vin+21a] and unabstracted **platform models** (PM) [Vin+21b].

Finally, several technical contributions are delivered as a part of this research work. To exploit the results of the proposed methodology and experimentations, several proof-of-concept urban data web applications are developed to provide users with integrated views of urban evolution (cf. contribution **C6**). These applications serve two use cases. The first use case provides users with methods of visualizing and navigating integrated **semantic** knowledge graph representations of urban data alongside **3D geospatial** representations of the data. The second use case provides users with methods to visualize and navigate **versioned snapshots** of city evolution and **concurrent viewpoints** this evolution. In addition, an effort is made to render the proposed model-driven transformation workflows and web applications **portable and reproducible** through containerization technologies such as Docker (cf. contribution **C7**).

Complementing these technical contributions, editorial contributions were made to the geospatial 3D urban data standard CityGML 3.0 and its recently proposed

GML encoding (cf. contribution **C8**). These contributions include the production of example datasets demonstrating how to structure versioned urban data and city addresses that supplement the encoding specification. These datasets are created from open 3D city models of Lyon and Washington D.C.

A synthesis of the proposed contributions is presented in table 7.1. This table estimates to what degree the proposed contributions are complete in terms of the aforementioned research questions, their level of maturity, or data quality. This thesis achieves at least 50% completion in all contributions.

	Contribution	Degree of completion
C1.a	XSD-based nD urban data model	**Mostly-completed**—Both the CityGML 2.0 and 3.0 models are aligned with a domain-agnostic geospatial data standard, are transformed to OWL, and are validated using a reasoner. Manual simplification of the ontology is still required to remove extraneous and duplicate classes and properties, and improve conciseness.
C1.b	UML-based nD urban data model	**Fully-completed**—Both the CityGML 2.0 and 3.0 models are aligned with domain-agnostic geospatial and temporal data standards. These models are transformed to OWL and are validated as logically consistent using a reasoner. Additionally, the ontologies are available online as linked open data, as their URI identifiers utilized in the dataset resolve to real-world URLs.
C2	Data model extension (Workspace data model)	**Fully-completed**—All of the Workspace ADE is updated to incorporate the latest CityGML 3.0 version. It is also transformed into an OWL ontology, and is validated as logically consistent using a reasoner. Additionally, the ontology is available online as linked open data, as its URI identifiers resolve to real-world URLs.
C3.a	Basic-temporal relation rules	**Fully-completed**—All the basic SWRL rules are formalized for inferring temporal relations from OWL-Time and the proposed OWL-Time extension.

C3.b	Workspace relation rules	**Partially-completed**—While many of the logical rules from the Workspace data model were translated to SWRL, several rules still require translation.
C4.a	XSD-based transformation workflows	**Fully-completed**—The XSD-based transformation workflow is complete.
C4.b	UML-based transformation workflows	**Fully-completed**—The UML-based transformation workflow is complete. Future works can improve upon the transformations by optimizing the code.
C4.c	Rule test suite	**Fully-completed**—As a proof-of-concept, the implemented test suite is complete. Similar to contribution *C4.b*, future works can improve upon the transformations by optimizing the code.
C5.a	1st district of Lyon 2015	**Mostly-completed**—The produced dataset is made available through Software Heritage, however the entire dataset has not been validated for logical consistency due to memory limitations of the computers effectuating the reasoning. A subset of the dataset has been validated for logical consistency using a reasoner in conjunction with the rule test suite and the proposed workspace rules. URIs should be updated to qualify this data as linked open data.
C5.b	La Doua campus 2009-2018	**Partially-completed**—The produced dataset is made available through Software Heritage, however the dataset was created as a proof-of-concept that has not been validated for logical consistency. URIs should be updated to qualify this data as linked open data.
C5.c	Gratte-ciel neighborhood 2009-2018	**Fully-completed**—The produced dataset validated according to the proof-of-concept test suite and is available on the Metropole of Lyon's open data repository. This dataset is also made available as open linked data, as its URI identifiers resolve to real-world URLs.
C6	UD-Viz extension and nD urban data web applications	**Mostly-completed**—The web applications are made available through online services, however, the proposed UD-Viz extension is still in a development stage to be released in a future version of UD-Viz.

C7	Reproducibility	**Mostly-completed**—All code, data, data models is made available with documentation wherever applicable. Concerning containerization for reproducibility, portable containers are provided for the technical contributions. However, some containers are out of date concerning more recent developments in UD-Graph and must be updated.
C8	CityGML 3.0 GML Encoding contributions	**Fully-completed**—All envisioned example datasets and editorial contributions are delivered

Table 7.1.: A comparison of the contributions proposed during this thesis in terms of the estimated degree to which they are completed. The degree of completion is scored on an *approximate* 5-point scale where a '**Fully-completed**' contribution is 100% completed, '**Mostly-completed**' corresponds to 75% completion, '**Partially-completed**' corresponds to 50% completion. This thesis achieves at least 50% completion in all contributions.

7.2 Discussion and future works

There are several perspectives for future works raised by the observed limitations of our works.

Firstly, while the proposed integration methodology can be used to migrate to ontological data representations, more work must be done to improve the semantic integration between different data standards. For example, data conformant to CityGML 2.0 and 3.0 can be integrated using this methodology, but proposing an alignment between the CityGML 2.0 and 3.0 ontologies may facilitate forwards compatibility between versions of the standard. Resolving forwards compatibility is an important requirement for developing tools such as the CityGML2-to-CityGML3 transformation tool[1] which is currently under development. Also, while alignments are proposed towards less domain-agnostic geospatial and temporal upper-ontologies (sections 4.3.1 and 5.3.2), no alignments are proposed towards domain-specific lower ontologies from other urban data silos like those discussed in section 3.1. For example, urban and built cultural heritage integration approaches can be enabled with the alignment of BIM, CIM, and GIS data models and documentation and cultural heritage standards such as CIDOC-CRM, CRMgeo, and DublinCore.

[1] https://github.com/tum-gis/citygml2-to-citygml3

Similarly, to facilitate BIM-GIS and BIM-CIM integration approaches for representing urban information at *multiple scales* while taking into account challenges to integrate both semantic and geometric data, geospatial city information models such as CityGML could be aligned with building information models such as IFC, IndoorGML or the Building Topology Ontology (BOT). Initial work on this subject has begun during the development of the work presented in section 6.2 in collaboration with Clément Colin—a doctoral student of the VCity project studying methods for "Multi-scale management, representation and visualization of assets in buildings and territories" [Sam+23; Col+23].

In addition, regarding the validation of transformed data and data models, work still remains. First, it is difficult to test if unidirectional transformations do not lose semantic or structural information. For this reason, a bidirectional transformation of ISO 19150-2 must be proposed to verify whether a round trip transformation can produce the same UML model as input to understand the limits of the proposed mappings [JOH19]. These transformations must also produce better ontological restrictions to ensure that the intended behavior of UML models cannot be violated when instantiating the transformed ontologies. For example, classes in OWL created from abstract classes in UML must not be instantiated. Similarly, restrictions are also necessary for properties created from composition and aggregation association roles. Implementing these restrictions may improve the bidirectional exchange of information [JOH19].

Also, while Semantic Web technologies have proven effective as an interoperable basis for storing and sharing geospatial data on the web, these technologies are still maturing in the domain of 3D data. Currently, this approach only uses existing ontological standards to *store* 3D urban data. 3D geospatial standards like 3DTiles are still required to facilitate the querying of 3D data. Approaches like BimSPARQL [ZBV18] have shown promise in permitting 3D geometric and topological queries of building data, but a more generic approach is required for all possible 3D geospatial linked data. In addition, the OGC has begun development of 3D geospatial support as a part of a future version of the GeoSPARQL standard[2,3,4,5], which—as a standards-based approach—should be taken into consideration as these features are developed.

Furthermore, while the complementary views of nD urban data through knowledge graph and 3D representations proposed in sections 6.2 and 6.2 facilitates visualization, navigation, and interaction of these data for the user, the integration of these

[2] https://github.com/opengeospatial/ogc-geosparql/issues/19
[3] https://github.com/opengeospatial/ogc-geosparql/issues/20
[4] https://github.com/opengeospatial/ogc-geosparql/issues/416
[5] https://github.com/opengeospatial/ogc-geosparql/issues/429

data can be improved. Enhancing the proposed integration approach is important for improving applications for comprehending city evolution, urban planning, and facilitating citizen participation in urbanization projects [CK19; SH20]. To this end, a collaboration was initiated with Corentin Gautier—another doctoral student of the VCity project studying "Dynamic, virtual and tangible representations of the city"—during the development of the work presented in section 6.2 [Col+23] to improve the integration of urban knowledge graphs and unstructured multimedia documents such as archival images and videos. Potential future works in this area include an improvement of the existing integration approaches for visualizing and navigating the evolution of 3D urban morphology.

Finally, with the advent of big data, scalability is an important challenge in non-relational database approaches to storing and retrieving data (such as relying on Semantic Web technologies) [PJ21]. This challenge is especially present in urban digital twin applications which often require performant solutions for analyzing large quantities of urban data [Lei+23]. In this subject, research in more *scalable integration* of the existing urban knowledge graph data, and how the data itself has evolved over time, is envisioned in collaboration with Jey Puget Gil—another doctoral student of the VCity project studying methods for storing, accessing, and reasoning over "Knowledge hub for evolving cities."

Bibliography

[INS17] ARE3NA project “INSPIRE Re3ference Platform Phase 2”. *Guidelines for the RDF encoding of spatial data*. EN. July 2017 (cit. on pp. 46, 47).

[All84] James F Allen. “Towards a general theory of action and time”. In: *Artificial intelligence* 23.2 (1984). Publisher: Elsevier, pp. 123–154 (cit. on pp. xiii, 33, 34).

[AK03] Colin Atkinson and Thomas Kühne. “Model-driven development: a metamodeling foundation. IEEE Software 20(5), 36-41”. In: *Software, IEEE* 20 (Oct. 2003), pp. 36–41 (cit. on pp. xii, 7, 19, 20, 23, 24, 27).

[Avi91] D. E. Avison. “MERISE: A European methodology for developing information systems”. In: *European Journal of Information Systems* 1.3 (Aug. 1991), pp. 183–191 (cit. on p. 18).

[BN03] Franz Baader and Werner Nutt. “Basic Description Logics”. In: *The description logic handbook: Theory, implementation and applications*. Cambridge university press, 2003, pp. 47–100 (cit. on pp. xii, 29, 164).

[Bal+22] Pasquale Balsebre, Dezhong Yao, Gao Cong, and Zhen Hai. “Geospatial Entity Resolution”. In: *Proceedings of the ACM Web Conference 2022*. WWW ’22. New York, NY, USA: Association for Computing Machinery, Apr. 2022, pp. 3061–3070 (cit. on p. 52).

[Bar+14] Luciano Barbosa, Kien Pham, Claudio Silva, Marcos R. Vieira, and Juliana Freire. “Structured Open Urban Data: Understanding the Landscape”. In: *Big Data* 2.3 (Sept. 2014), pp. 144–154 (cit. on pp. 3, 6).

[Bat+17] Sotiris Batsakis, Euripides G. M. Petrakis, Ilias Tachmazidis, and Grigoris Antoniou. “Temporal representation and reasoning in OWL 2”. en. In: *Semantic Web* 8.6 (Jan. 2017), pp. 981–1000 (cit. on p. 53).

[BK12] Robert Battle and Dave Kolas. “Enabling the geospatial Semantic Web with Parliament and GeoSPARQL”. en. In: *Semantic Web* 3.4 (2012), pp. 355–370 (cit. on pp. 31, 50).

[Bat18] Michael Batty. “Digital twins”. en. In: *Environment and Planning B: Urban Analytics and City Science* 45.5 (Sept. 2018). Publisher: SAGE Publications Ltd STM, pp. 817–820 (cit. on pp. 3, 4, 34, 37).

[BBK20] F. Beck, A. Borrmann, and T. H. Kolbe. “The need for a differentiation between heterogeneous information integration approaches in the field of “bim-gis integration”: A literature review”. In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* VI-4/W1-2020 (2020), pp. 21–28 (cit. on pp. xiii, 41, 42).

[Bed+11] Ivan Bedini, Christopher Matheus, Peter F. Patel-Schneider, Aidan Boran, and Benjamin Nguyen. “Transforming XML Schema to OWL Using Patterns”. In: *2011 IEEE Fifth International Conference on Semantic Computing*. Sept. 2011, pp. 102–109 (cit. on pp. 21, 47–49, 63, 69).

[Ber19] Camille Bernard. “Immersing evolving geographic divisions in the semantic Web”. en. PhD thesis. Université Grenoble Alpes, Nov. 2019 (cit. on pp. 32, 34, 52, 54).

[BHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. “The semantic web”. In: *Scientific american* 284.5 (2001), pp. 34–43 (cit. on pp. xii, 30).

[BKN18] Filip Biljecki, Kavisha Kumar, and Claus Nagel. “CityGML Application Domain Extension (ADE): overview of developments”. In: *Open Geospatial Data, Software and Standards* 3.1 (Aug. 2018), p. 13 (cit. on pp. 12, 42).

[Bil+15] Filip Biljecki, Jantien Stoter, Hugo Ledoux, Sisi Zlatanova, and Arzu Çöltekin. “Applications of 3D City Models: State of the Art Review”. en. In: *ISPRS International Journal of Geo-Information* 4.4 (Dec. 2015). Number: 4 Publisher: Multidisciplinary Digital Publishing Institute, pp. 2842–2889 (cit. on pp. 4, 37).

[Bis+14] Stefan Bischof, Athanasios Karapantelakis, Cosmin-Septimiu Nechifor, et al. “Semantic Modelling of Smart City Data”. en. In: Berlin, Germany: W3C, 2014 (cit. on p. 105).

[BA05] Hannes Bohring and Sören Auer. “Mapping XML to OWL ontologies”. en. In: *Marktplatz Internet: Von e-Learning bis e-Payment, 13. Leipziger Informatik-Tage (LIT 2005)*. Gesellschaft für Informatik e. V., 2005, pp. 147–156 (cit. on pp. 9, 20, 48, 49, 63, 64, 69).

[Bon+19] Mathias Bonduel, Anna Wagner, Pieter Pauwels, Maarten Vergauwen, and Ralf Klein. “Including widespread geometry formats in semantic graphs using RDF literals”. eng. In: *Proceedings of the 2019 European Conference for Computing in Construction*. ISSN: 2684-1150. European Council on Computing in Construction, 2019, pp. 341–350 (cit. on p. 75).

[Bri18] Linda Brink. “Geospatial Data on the Web”. PhD thesis. Oct. 2018 (cit. on p. 6).

[Bri+14] Linda Brink, Paul Janssen, Wilko Quak, and Jantien Stoter. “Linking spatial data: automated conversion of geo-information models and GML data to RDF”. In: *International Journal of Spatial Data Infrastructures Research* 9 (Oct. 2014), pp. 59–85 (cit. on pp. 30, 45, 46, 50, 69, 77, 78, 106).

[Bui+] Carlos Buil Aranda, Corby Olivier, Souripriya Das, et al., eds. *SPARQL 1.1 Overview* (cit. on p. 29).

[Cha+21] Arkadiusz Chadzynski, Nenad Krdzavac, Feroz Farazi, et al. “Semantic 3D City Database — An enabler for a dynamic geospatial knowledge graph”. In: *Energy and AI* 6 (2021), p. 100106 (cit. on pp. 41, 74, 75, 78, 104).

[CK19] Kanishk Chaturvedi and Thomas Kolbe. “A Requirement Analysis on Extending Semantic 3D City Models for Supporting Time-Dependent Properties”. In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* IV-4/W9 (Sept. 2019), pp. 19–26 (cit. on pp. 4, 34, 37, 39, 139).

[Cha+17] Kanishk Chaturvedi, Carl Stephen Smyth, Gilles Gesquière, Tatjana Kutzner, and Thomas H. Kolbe. “Managing Versions and History Within Semantic 3D City Models for the Next Generation of CityGML”. In: *Lecture Notes in Geoinformation and Cartography*. Ed. by Alias Abdul-Rahman. Advances in 3D Geoinformation. Springer, 2017, pp. 191–206 (cit. on pp. 34, 95, 127).

[CGP22] Serge Chávez-Feria, Raúl García-Castro, and María Poveda-Villalón. “Chowlk: from UML-based ontology conceptualizations to owl”. In: *The Semantic Web: 19th International Conference, ESWC 2022, Hersonissos, Crete, Greece, May 29–June 2, 2022, Proceedings*. Springer. 2022, pp. 338–352 (cit. on p. 45).

[Che76] Peter Pin-Shan Chen. “The entity-relationship model—toward a unified view of data”. In: *ACM Transactions on Database Systems* 1.1 (Mar. 1976), pp. 9–36 (cit. on p. 17).

[Cla20] Christophe Claramunt. “Ontologies for geospatial information: Progress and challenges ahead”. In: *Journal of Spatial Information Science* (June 2020) (cit. on pp. 26, 36, 37).

[Cod70] Edgar F Codd. “A relational model of data for large shared data banks”. In: *Communications of the ACM* 13.6 (1970). Publisher: ACM New York, NY, USA, pp. 377–387 (cit. on p. 17).

[Col+22] C. Colin, J. Samuel, S. Servigne, C. Bortolaso, and G. Gesquière. “Creating Contextual View of Cmms Assets Using Geospatial 2D/3D Data”. In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* X-4/W2-2022 (2022), pp. 37–44 (cit. on pp. 14, 108, 114).

[Col+23] Clément Colin, Corentin Gautier, Diego Vinasco-Alvarez, et al. “UD-SV : Plateforme d’exploration de données urbaines à n-dimensions — Espace, Temps, Thématiques”. fr. In: *Mappemonde. Revue trimestrielle sur l’image géographique et les formes du territoire* 135 (Mar. 2023). Number: 135 Publisher: UMR Espace (cit. on pp. xvi, xx, 107, 114, 134, 138, 139).

[Col+20] Elisabetta Colucci, Valeria De Ruvo, Andrea Lingua, Francesca Matrone, and Gloria Rizzo. “HBIM-GIS Integration: From IFC to CityGML Standard for Damaged Cultural Heritage in a Multiscale 3D GIS”. en. In: *Applied Sciences* 10.4 (Jan. 2020). Number: 4 Publisher: Multidisciplinary Digital Publishing Institute, p. 1356 (cit. on p. 39).

[Com20] Wikimedia Commons. *File:Vue aérienne de la cité des Gratte-Ciel en 1936.jpg — Wikimedia Commons, the free media repository*. 2020 (cit. on pp. xvi, 116).

[Com22] Wikimedia Commons. *File:Usine de chaufferie du campus de la Doua, à Villeurbanne..jpg — Wikimedia Commons, the free media repository*. [Online; accessed 17-March-2023]. 2022 (cit. on pp. xii, 19).

[CA17] Isabelle Comyn-Wattiau and Jacky Akoka. “Model driven reverse engineering of NoSQL property graph databases: The case of Neo4j”. In: *2017 IEEE International Conference on Big Data (Big Data)*. Dec. 2017, pp. 453–458 (cit. on p. 44).

[Cox13] Simon J D Cox. “An explicit OWL representation of ISO/OGC Observations and Measurements”. en. In: Jan. 2013, pp. 1–18 (cit. on pp. 46, 47, 78, 99, 106).

[CN08] Christophe Cruz and Christophe Nicolle. “Ontology Enrichment and Automatic Population From XML Data”. In: Jan. 2008, pp. 17–20 (cit. on p. 49).

[CH06] K. Czarnecki and S. Helsen. “Feature-based survey of model transformation approaches”. In: *IBM Systems Journal* 45.3 (2006), pp. 621–645 (cit. on pp. xiii, 44).

[De +17] Dieter De Paepe, Geert Thijs, Raf Buyle, Ruben Verborgh, and Erik Mannens. “Automated UML-Based Ontology Generation in OSLO2”. In: Nov. 2017, pp. 93–97 (cit. on pp. 45, 46).

[DMF13] Francisco Delgado del Hoyo, M. Mercedes Martínez-González, and Javier Finat. “An evaluation of ontology matching techniques on geospatial ontologies”. In: *International Journal of Geographical Information Science* 27 (June 2013), pp. 2279–2301 (cit. on p. 51).

[DZS21] Tianhu Deng, Keren Zhang, and Zuo-Jun (Max) Shen. “A systematic review of a digital twin city: A new pattern of urban governance toward smart cities”. en. In: *Journal of Management Science and Engineering* 6.2 (June 2021), pp. 125–134 (cit. on p. 3).

[Dre+20] Urška Drešček, Mojca Kosmatin Fras, Jernej Tekavec, and Anka Lisec. “Spatial ETL for 3D building modelling based on unmanned aerial vehicle data in semi-urban areas”. In: *Remote Sensing* 12.12 (2020), p. 1972 (cit. on p. 9).

[ES13] Jérôme Euzenat and Pavel Shvaiko. *Ontology Matching*. 2nd ed. 2013. Berlin, Heidelberg: Springer Berlin Heidelberg : Imprint: Springer, 2013 (cit. on pp. 41, 51).

[FZT04] Matthias Ferdinand, Christian Zirpins, and David Trastour. “Lifting XML schema to OWL”. In: vol. 3140. July 2004, pp. 354–358 (cit. on pp. 21, 48, 49, 63).

[Gal05] Emmanuelle Gallo. “La réception et le quartier des gratte-ciel, centre de Villeurbanne, ou pourquoi des gratte-ciel à Villeurbanne en 1932?” In: *Docomomo Journal* (2005) (cit. on p. 116).

[GKK13] George Garbis, Kostis Kyzirakos, and Manolis Koubarakis. “Geographica: A benchmark for geospatial rdf stores (long version)”. In: *The Semantic Web–ISWC 2013: 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part II 12*. Springer, 2013, pp. 343–359 (cit. on p. 129).

[Gas+04] Dragan Gasevic, Dragan Djuric, Vladan Devedzic, and Violeta Damjanovi. “Converting UML to OWL ontologies”. In: *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*. WWW Alt. ’04. New York, NY, USA: Association for Computing Machinery, May 2004, pp. 488–489 (cit. on p. 45).

[Gau+15] Aditya Gaur, Bryan Scotney, Gerard Parr, and Sally McClean. “Smart City Architecture and its Applications Based on IoT”. en. In: *Procedia Computer Science*. The 6th International Conference on Ambient Systems, Networks and Technologies (ANT-2015), the 5th International Conference on Sustainable Energy Information Technology (SEIT-2015) 52 (Jan. 2015), pp. 1089–1094 (cit. on p. 105).

[GDG22] C. Gautier, J. Delanoy, and G. Gesquière. “Integrating Multimedia Documents in 3D City Models for a Better Understanding of Territories”. In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* X-4/W2-2022 (2022), pp. 69–76 (cit. on pp. xvi, 108, 113).

[Gli+14] Birte Glimm, Ian Horrocks, Boris Motik, Giorgos Stoilos, and Zhe Wang. “HermiT: An OWL 2 Reasoner”. en. In: *Journal of Automated Reasoning* 53.3 (Oct. 2014), pp. 245–269 (cit. on p. 54).

[Gra+08] Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, et al. “OWL 2: The next step for OWL”. In: *Journal of Web Semantics* 6.4 (2008), pp. 309–322 (cit. on p. 41).

[Gri14] Michael Grieves. “Digital Twin: Manufacturing Excellence through Virtual Factory Replication”. In: (Mar. 2014) (cit. on p. 3).

[GP12] Gerhard Gröger and Lutz Plümer. “CityGML - Interoperable semantic 3D city models”. en. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 71 (July 2012), pp. 12–33 (cit. on pp. 5, 11, 38).

[GOS09] Nicola Guarino, Daniel Oberle, and Steffen Staab. “What Is an Ontology?” en. In: *Handbook on Ontologies*. Ed. by Steffen Staab and Rudi Studer. International Handbooks on Information Systems. Berlin, Heidelberg: Springer, 2009, pp. 1–17 (cit. on pp. xii, 18, 26–28).

[HBC15] Mokhtaria Hacherouf, Safia Nait Bahloul, and Christophe Cruz. “Transforming XML documents to OWL ontologies: A survey”. In: *Journal of Information Science* 41.2 (2015), pp. 242–259 (cit. on pp. 47–49).

[HNC19] Mokhtaria Hacherouf, Safia Nait-Bahloul, and Christophe Cruz. “Transforming XML schemas into OWL ontologies using formal concept analysis”. en. In: *Software & Systems Modeling* 18.3 (June 2019), pp. 2093–2110 (cit. on pp. 48, 49).

[Har15] Benjamin Harbelot. “Continuum : un modèle spatio-temporel et sémantique pour la découverte de phénomènes dynamiques au sein d’environnements géospatiaux”. Theses. Université de Bourgogne, Dec. 2015 (cit. on p. 54).

[HRB20] Elio Hbeich, Ana Roxin, and Nicolas Bus. “Previous BIM-GIS Integration Approaches: Analytic Review and Discussion”. In: Oct. 2020 (cit. on pp. 25, 37–39).

[HHP12] Ivan Herman, Ian Horrocks, and Peter F. Patel-Schneider, eds. *OWL 2 Web Ontology Language Document Overview (Second Edition)*. Dec. 2012 (cit. on p. 28).

[HS14] Hlomani Hlomani and Deborah Stacey. “Approaches, methods, metrics, measures, and subjectivity in ontology evaluation: A survey”. English. In: *International Journal on Semantic Web and Information Systems* 1.5 (2014), pp. 1–11 (cit. on p. 54).

[Hog+21] Aidan Hogan, Eva Blomqvist, Michael Cochez, et al. “Knowledge graphs”. In: *ACM Computing Surveys (CSUR)* 54.4 (2021), pp. 1–37 (cit. on pp. 22, 27, 30).

[HE00] Kathleen Hornsby and Max J. Egenhofer. “Identity-based change: a foundation for spatio-temporal knowledge representation”. In: *International Journal of Geographical Information Science* 14.3 (2000), pp. 207–224. eprint: https://doi.org/10.1080/136588100240813 (cit. on p. 32).

[Hor+04] Horrocks, Ian, Patel-Schneider, et al. “SWRL: A Semantic Web rule language combining OWL and RuleML”. In: *W3C Subm* 21 (Jan. 2004) (cit. on pp. 27, 30).

[ISO14] ISO/TC 211. *ISO 19101-1:2014 Geographic information — Reference model — Part 1: Fundamentals*. en. 2014 (cit. on pp. 7, 18, 19).

[ISO15a] ISO/TC 211. *ISO 19103:2015*. en. 2015 (cit. on p. 25).

[ISO15b] ISO/TC 211. *ISO 19109:2015, Geographic information - Rules for application schema*. 2015 (cit. on p. 25).

[Jai20] Vincent Jaillot. “3D, temporal and documented cities : formalization, visualization and navigation”. Theses. Université de Lyon, Sept. 2020 (cit. on p. 39).

[Jai+21] Vincent Jaillot, Valentin Rigolle, Sylvie Servigne, John Samuel Samuel, and Gilles Gesquière. “Integrating multimedia documents and time-evolving 3D city models for web visualization and navigation”. In: *Transactions in GIS* 25.3 (Mar. 2021), pp. 1419–1438 (cit. on pp. xi, xvii, 5, 108, 117, 118).

[JSG20] Vincent Jaillot, Sylvie Servigne, and Gilles Gesquière. “Delivering time-evolving 3D city models for web visualization”. In: *International Journal of Geographical Information Science* (2020). Publisher: Taylor & Francis, 25 p. (Cit. on pp. 4, 14, 34, 37, 108, 114, 118, 122).

[JOH19] Knut Jetlund, Erling Onstein, and Lizhen Huang. “Adapted Rules for UML Modelling of Geospatial Information for Model-Driven Implementation as OWL Ontologies”. In: *ISPRS International Journal of Geo-Information* 8.9 (2019) (cit. on pp. xii, 25, 26, 45–47, 106, 138, 162).

[Jon+20] David Jones, Chris Snider, Aydin Nassehi, Jason Yon, and Ben Hicks. “Characterising the Digital Twin: A systematic literature review”. en. In: *CIRP Journal of Manufacturing Science and Technology* 29 (May 2020), pp. 36–52 (cit. on p. 3).

[Jul+18] Arttu Julin, Kaisa Jaalama, Juho-Pekka Virtanen, et al. “Characterizing 3D City Modeling Projects: Towards a Harmonized Interoperable System”. en. In: *ISPRS International Journal of Geo-Information* 7.2 (Feb. 2018). Number: 2 Publisher: Multidisciplinary Digital Publishing Institute, p. 55 (cit. on pp. 3, 37).

[KWB03] Anneke G. Kleppe, Jos Warmer, and Wim Bast. *MDA Explained: The Model Driven Architecture: Practice and Promise*. USA: Addison-Wesley Longman Publishing Co., Inc., 2003 (cit. on p. 43).

[Kol09] Thomas Kolbe. “Representing and Exchanging 3D City Models with CityGML”. In: *3D Geo-Information Sciences*. Journal Abbreviation: 3D Geo-Information Sciences. Jan. 2009, pp. 15–31 (cit. on p. 38).

[KD21] Thomas H. Kolbe and Andreas Donaubauer. “Semantic 3D City Modeling and BIM”. In: *Urban Informatics*. Ed. by Wenzhong Shi, Michael F. Goodchild, Michael Batty, Mei-Po Kwan, and Anshu Zhang. Singapore: Springer Singapore, 2021, pp. 609–636 (cit. on p. 3).

[KRZ13] Roman Kontchakov, Mariano Rodriguez-Muro, and Michael Zakharyaschev. “Ontology-based data access with databases: A short course”. In: *Reasoning Web. Semantic Technologies for Intelligent Data Access: 9th International Summer School 2013, Mannheim, Germany, July 30–August 2, 2013. Proceedings* (2013), pp. 194–229 (cit. on p. 41).

[Kra+15] Thomas R. Kramer, Benjamin H. Marks, Craig I. Schlenoff, et al. “Software Tools for XML to OWL Translation”. en. In: (July 2015). Last Modified: 2018-11-10T10:11-05:00 (cit. on pp. xiii, 49, 50, 63).

[Küh06] Thomas Kühne. “Matters of (Meta-) Modeling”. en. In: *Software & Systems Modeling* 5.4 (Dec. 2006), pp. 369–385 (cit. on pp. xii, 18, 23, 24).

[Kut16] Tatjana Kutzner. “Geospatial data modelling and model-driven transformation of geospatial data based on uml profiles”. PhD Thesis. Technische Universität München, 2016 (cit. on pp. xii, xiii, 5, 7–9, 18–20, 42–44, 47–49).

[KCK20] Tatjana Kutzner, Kanishk Chaturvedi, and Thomas H. Kolbe. “CityGML 3.0: New Functions Open Up New Applications”. en. In: *PFG - Journal of Photogrammetry, Remote Sensing and Geoinformation Science* 88.1 (Feb. 2020), pp. 43–61 (cit. on pp. xiii, xviii, 33, 34, 39, 127, 169).

[Kut+23] Tatjana Kutzner, Carl Stephen Smyth, Claus Nagel, et al., eds. *OGC City Geography Markup Language (CityGML) Part 2: GML Encoding Standard*. Draft. Jan. 2023 (cit. on pp. xvii, 127, 128).

[KKK12] Kostis Kyzirakos, Manos Karpathiotakis, and Manolis Koubarakis. “Strabon: A semantic geospatial DBMS”. In: *The Semantic Web–ISWC 2012: 11th International Semantic Web Conference, Boston, MA, USA, November 11-15, 2012, Proceedings, Part I 11*. Springer Berlin Heidelberg. 2012, pp. 295–311 (cit. on pp. 33, 50, 108).

[Kyz+18] Kostis Kyzirakos, Dimitrianos Savva, Ioannis Vlachopoulos, et al. “GeoTriples: Transforming geospatial data into RDF graphs using R2RML and RML mappings”. en. In: *Journal of Web Semantics* 52-53 (Oct. 2018), pp. 16–32 (cit. on pp. 41, 49).

[Lam17] Jean-Baptiste Lamy. “Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies”. In: *Artificial intelligence in medicine* 80 (2017). Publisher: Elsevier, pp. 11–28 (cit. on pp. 54, 101).

[Lei+23] Binyu Lei, Patrick Janssen, Jantien Stoter, and Filip Biljecki. “Challenges of urban digital twins: A systematic review and a Delphi expert survey”. en. In: *Automation in Construction* 147 (Mar. 2023), p. 104716 (cit. on pp. 6, 7, 9, 139).

[LN+07] Ulf Leser, Felix Naumann, et al. “Informationsintegration”. In: *dpunkt, Heidelberg* (2007) (cit. on p. 7).

[Li+22] Wenwen Li, Sizhe Wang, Sheng Wu, Zhining Gu, and Yuanyuan Tian. “Performance benchmark on semantic web repositories for spatially explicit knowledge graph applications”. en. In: *Computers, Environment and Urban Systems* 98 (Dec. 2022), p. 101884 (cit. on p. 129).

[Liu+17] Xin Liu, Xiangyu Wang, Graeme Wright, et al. “A State-of-the-Art Review on the Integration of Building Information Modeling (BIM) and Geographic Information System (GIS)”. en. In: *ISPRS International Journal of Geo-Information* 6.2 (Feb. 2017), p. 53 (cit. on p. 39).

[Lóp+18] Facundo José López, Pedro M Lerones, José Llamas, Jaime Gómez-García-Bermejo, and Eduardo Zalama. “A review of heritage building information modeling (H-BIM)”. In: *Multimodal Technologies and Interaction* 2.2 (2018), p. 21 (cit. on pp. 34, 39).

[Mag91] David J Maguire. “An overview and definition of GIS”. In: *Geographical information systems: Principles and applications* 1.1 (1991), pp. 9–20 (cit. on p. 38).

[Mal+20] Eva Savina Malinverni, Berardo Naticchia, Jose Luis Lerma Garcia, et al. “A semantic graph database for the interoperability of 3D GIS data”. en. In: *Applied Geomatics* (Aug. 2020) (cit. on p. 36).

[Mar+22] L. Marnat, C. Gautier, C. Colin, and G. Gesquière. “Py3Dtilers: An Open Source Toolkit for Creating and Managing 2D/3D Geospatial Data”. In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* X-4/W3-2022 (2022), pp. 165–172 (cit. on pp. 14, 108, 118).

[MV06] Tom Mens and Pieter Van Gorp. “A taxonomy of model transformation”. In: *Electronic notes in theoretical computer science* 152 (2006), pp. 125–142 (cit. on p. 44).

[Mes+18] Tommy Messaoudi, Philippe Véron, Gilles Halin, and Livio De Luca. “An ontological model for the reality-based 3D annotation of heritage building conservation state”. en. In: *Journal of Cultural Heritage* 29 (Jan. 2018), pp. 100–112 (cit. on p. 39).

[MF18] C. Métral and G. Falquet. “Extension and Contextualization for Linked Semantic 3D Geodata”. English. In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. XLII-4-W10. ISSN: 1682-1750. Copernicus GmbH, Sept. 2018, pp. 113–118 (cit. on pp. 74, 75, 77, 104).

[Mün+20] Sander Münster, Florian Niebling, Jonas Bruschke, et al. “Urban History Research and Discovery in the Age of Digital Repositories. A Report About Users and Requirements”. In: *Digital Cultural Heritage*. Ed. by Horst Kremers. Cham: Springer International Publishing, 2020, pp. 63–84 (cit. on p. 39).

[Mus15] Mark A. Musen. “The Protégé Project: A Look Back and a Look Forward”. In: *AI matters* 1.4 (June 2015), pp. 4–12 (cit. on pp. 54, 101).

[NK20] S. H. Nguyen and T. H. Kolbe. “A Multi-Perspective Approach to Interpreting Spatio-Semantic Changes of Large 3D City Models in CityGML Using a Graph Database”. English. In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. VI-4-W1-2020. ISSN: 2194-9042. Copernicus GmbH, Sept. 2020, pp. 143–150 (cit. on pp. 33, 105).

[NK21] S. H. Nguyen and T. H. Kolbe. “Modelling Changes, Stakeholders and Their Relations in Semantic 3D City Models”. English. In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. VIII-4-W2-2021. Copernicus GmbH, Oct. 2021, pp. 137–144 (cit. on pp. 34, 39).

[NCM19] Ikrom Nishanbaev, Erik Champion, and David A. McMeekin. “A Survey of Geospatial Semantic Web for Cultural Heritage”. In: *Heritage* 2.2 (2019), pp. 1471–1498 (cit. on p. 39).

[NM+01] Natalya F Noy, Deborah L McGuinness, et al. *Ontology development 101: A guide to creating your first ontology*. 2001 (cit. on p. 82).

[OD10] Martin J O’Connor and Amar K Das. “A method for representing and querying temporal information in OWL”. In: *International joint conference on biomedical engineering systems and technologies*. Springer. 2010, pp. 97–110 (cit. on pp. 54, 101).

[Ope17] Open Geospatial Consortium. *Testbed-12 ShapeChange Engineering Report*. Public Engineering Report. Open Geospatial Consortium, Apr. 4, 2017 (cit. on pp. 46, 47, 49, 86–89, 99).

[Ope22] Open Geospatial Consortium. *OGC Testbed-17: UML Modeling Best Practice Engineering Report*. Engineering Report 21-031. Open Geospatial Consortium, Feb. 8, 2022 (cit. on pp. 20, 42).

[PJ21] Archana Patel and Sarika Jain. “Present and future of semantic web technologies: a research statement”. In: *International Journal of Computers and Applications* 43.5 (2021). Publisher: Taylor & Francis, pp. 413–422 (cit. on p. 139).

[Pat+14] K. Patroumpas, M. Alexakis, Giorgos Giannopoulos, and S. Athanasiou. “TripleGeo: An ETL tool for transforming geospatial data into RDF triples”. In: *CEUR Workshop Proceedings* 1133 (Jan. 2014), pp. 275–278 (cit. on pp. 9, 41, 49).

[PV12] Pieter Pauwels and Davy Van Deursen. “IFC/RDF: adaptation, aggregation and enrichment”. In: *First international workshop on linked data in architecture and construction*. 2012, pp. 1–3 (cit. on p. 114).

[PMG15] F. Pédrinis, M. Morel, and G. Gesquière. “Change Detection of Cities”. In: *3D Geoinformation Science: The Selected Papers of the 3D GeoInfo 2014*. Cham: Springer International Publishing, 2015, pp. 123–139 (cit. on pp. 14, 52, 118).

[PB11] George Percivall and Kurt Buehler, eds. *OGC Reference Model*. Dec. 2011 (cit. on pp. 5, 25).

[PH12] Matthew Perry and John Herring, eds. *GeoSPARQL - A Geographic Query Language for RDF Data | OGC*. Sept. 2012 (cit. on p. 72).

[Ras+20] Mads Holten Rasmussen, Maxime Lefrançois, Georg Schneider, and Pieter Pauwels. “BOT: the Building Topology Ontology of the W3C Linked Building Data Group”. In: *Semantic Web* (Nov. 2020) (cit. on pp. 31, 46).

[Ren00] Agnar Renolen. “Modelling the Real World: Conceptual Modelling in Spatiotemporal Information System Design”. In: *Transactions in GIS* 4.1 (2000), pp. 23–42 (cit. on pp. xiii, 32, 34, 117).

[RP16] Claude Rochet and Juan David Pinzon. “Urban lifecycle management: A research program for smart government of smart cities”. In: *Revista de Gestão e Secretariado* 7 (Aug. 2016), pp. 1–20 (cit. on p. 3).

[Rod+17] P. Rodríguez-Gonzálvez, A. L. Muñoz-Nieto, S. del Pozo, et al. “4D RECONSTRUCTION AND VISUALIZATION OF CULTURAL HERITAGE: ANALYZING OUR LEGACY THROUGH TIME”. English. In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. XLII-2-W3. ISSN: 1682-1750. Copernicus GmbH, Feb. 2017, pp. 609–616 (cit. on pp. xiii, 34, 39, 40).

[Sam+23] John Samuel, Vincent Jaillot, Clément Colin, et al. “UD-SV: Urban data services and visualization framework for sharing multidisciplinary research”. en. In: *Transactions in GIS* 27.3 (Apr. 2023), pp. 841–858 (cit. on pp. xvi, xx, 10, 107–109, 138).

[Sam+16] John Samuel, Clémentine Périnaud, Sylvie Servigne, Georges Gay, and Gilles Gesquière. “Representation and Visualization of Urban Fabric through Historical Documents”. In: Oct. 2016 (cit. on p. 39).

[SSG20] John Samuel, Sylvie Servigne, and Gilles Gesquière. “Representation of concurrent points of view of urban changes for city models”. en. In: *Journal of Geographical Systems* 22.3 (July 2020), pp. 335–359 (cit. on pp. xiii, xviii, 4, 12, 34–37, 53, 106, 117, 119, 169).

[SH20] Gerhard Schrotter and Christian Hürzeler. “The Digital Twin of the City of Zurich for Urban Planning”. en. In: *PFG - Journal of Photogrammetry, Remote Sensing and Geoinformation Science* 88.1 (Feb. 2020), pp. 99–112 (cit. on pp. xi, 3, 4, 37, 139).

[SGV06] Vivek Sehgal, Lise Getoor, and Peter D Viechnicki. “Entity resolution in geospatial data integration”. In: *Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems*. GIS ’06. New York, NY, USA: Association for Computing Machinery, Nov. 2006, pp. 83–90 (cit. on p. 52).

[Sei03] E. Seidewitz. “What models mean”. In: *IEEE Software* 20.5 (2003), pp. 26–32 (cit. on p. 23).

[SK03] Shane Sendall and Wojtek Kozaczynski. “Model transformation: The heart and soul of model-driven software development”. In: *IEEE software* 20.5 (2003). Publisher: IEEE, pp. 42–45 (cit. on p. 42).

[Ser10] Sylvie Servigne. “Conception, architecture et urbanisation des systèmes d’information”. fr. In: *Encyclopaedia Universalis* (2010) (cit. on p. 43).

[SCI18] Willington Siabato, Christophe Claramunt, and Sergio Ilarri. “A Survey of Modelling Trends in Temporal GIS”. In: *ACM Computing Surveys* 51.2 (2018), 30:1–30:41 (cit. on p. 9).

[SKH18] Bhagya Nathali Silva, Murad Khan, and Kijun Han. “Towards sustainable smart cities: A review of trends, architectures, components, and open challenges in smart cities”. en. In: *Sustainable Cities and Society* 38 (Apr. 2018), pp. 697–713 (cit. on p. 3).

[Sir+07] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. “Pellet: A practical OWL-DL reasoner”. en. In: *Journal of Web Semantics*. Software Engineering and the Semantic Web 5.2 (June 2007), pp. 51–53 (cit. on p. 54).

[SMJ02] Peter Spyns, Robert Meersman, and Mustafa Jarrar. “Data modelling versus ontology engineering”. In: *ACM SIGMOD Record* 31.4 (Dec. 2002), pp. 12–17 (cit. on p. 5).

[SK07] Alexandra Stadler and Thomas H. Kolbe. “Spatio-semantic coherence in the integration of 3D city models”. In: *Proceedings of the 5th International ISPRS Symposium on Spatial Data Quality ISSDQ 2007 in Enschede, The Netherlands, 13-15 June 2007*. Ed. by Alfred Stein. ISPRS Archives. ISPRS, 2007 (cit. on pp. xiii, 38).

[Ste+08] Chiara Stefani, Livio De Luca, Philippe Véron, and Michel Florenzano. “Reasoning about space-time changes: an approach for modeling the temporal dimension in architectural heritage”. In: *IADIS International Conference*. Netherlands, 2008, pp. 1–6 (cit. on p. 32).

[SBF98] Rudi Studer, V. Richard Benjamins, and Dieter Fensel. “Knowledge engineering: principles and methods. Data Knowl Eng 25(1-2):161-197”. In: *Data & Knowledge Engineering* 25 (Mar. 1998), pp. 161–197 (cit. on p. 27).

[TSD12] Rumyana Tonchovska, Victoria Stanley, and Samantha De Martino. *Spatial Data Infrastructure and INSPIRE*. en. Sept. 2012 (cit. on p. 6).

[Tra+20] Ba-Huy Tran, Nathalie Aussenac-Gilles, Catherine Comparot, and Cassia Trojahn. “Semantic Integration of Raster Data for Earth Observation: An RDF Dataset of Territorial Unit Versions with their Land Cover”. en. In: *ISPRS International Journal of Geo-Information* 9.9 (Sept. 2020). Number: 9 Publisher: Multidisciplinary Digital Publishing Institute, p. 503 (cit. on pp. 7, 40, 41, 49, 54).

[Tra+16] Ba-Huy Tran, Christine Plumejeaud, Alain Bouju, and Vincent Bretagnolle. “Towards a semantic framework for exploiting heterogeneous environmental data”. In: *International Journal of Metadata, Semantics and Ontologies* 11 (Dec. 2016) (cit. on pp. xiii, 6, 29, 52–54).

[TH06] Dmitry Tsarkov and Ian Horrocks. “FaCT++ Description Logic Reasoner: System Description”. en. In: *Automated Reasoning*. Ed. by Ulrich Furbach and Natarajan Shankar. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2006, pp. 292–297 (cit. on p. 54).

[Uni19] Population Division United Nations Department of Economic and Social Affairs. *World urbanization prospects: the 2018 revision*. en. ST/ESA/SER.A/420. New York: United Nations, 2019 (cit. on p. 3).

[Urb18] Michael C. Urban. *Abandoning Silos: How Innovative Governments are Collaborating Horizontally to Solve Complex Problems*. Mowat Centre for Policy Innovation, Dec. 2018 (cit. on p. 6).

[UG04] Michael Uschold and Michael Gruninger. “Ontologies and semantics for seamless connectivity”. In: *ACM SIGMod Record* 33.4 (2004), pp. 58–64 (cit. on pp. xii, 26, 27).

[UJS20] A. U. Usmani, M. Jadidi, and G. Sohn. “Automatic Ontology Generation of BIM and GIS Data”. English. In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. XLIII-B4-2020. ISSN: 1682-1750. Copernicus GmbH, Aug. 2020, pp. 77–80 (cit. on p. 49).

[UJS21] A. U. Usmani, M. Jadidi, and G. Sohn. “Towards the Automatic Ontology Generation and Alignment of BIM and GIS Data Formats”. English. In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. VIII-4-W2-2021. Copernicus GmbH, Oct. 2021, pp. 183–188 (cit. on pp. 37, 51).

[Van+15] Muriel Van Ruymbeke, Cyril Carré, Vincent Delfosse, Michelle Pfeiffer, and Roland Billen. *Towards an Archaeological Information System: Improving the Core Data Model*. en. Archaeopress, 2015 (cit. on p. 32).

[Vas09] Panos Vassiliadis. "A survey of Extract–transform–Load technology". In: *International Journal of Data Warehousing & Mining* 5.3 (2009), pp. 1–27 (cit. on p. 9).

[Vil+17] S. Vilgertshofer, J. Amann, B. Willenborg, A. Borrmann, and T. H. Kolbe. "Linking BIM and GIS Models in Infrastructure by Example of IFC and CityGML". en. In: Publisher: American Society of Civil Engineers. June 2017, pp. 133–140 (cit. on p. 51).

[Vin22] Diego Vinasco-Alvarez. "Leveraging Standards in Model-Centric Geospatial Knowledge Graph Creation". en. In: *The Semantic Web: ESWC 2022 Satellite Events*. Ed. by Paul Groth, Anisa Rula, Jodi Schneider, et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2022, pp. 224–233 (cit. on pp. xx, 81).

[Vin+21a] Diego Vinasco-Alvarez, John Samuel, Sylvie Servigne, and Gilles Gesquière. "Towards Limiting Semantic Data Loss In 4D Urban Data Semantic Graph Generation". In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* VIII-4/W2-2021 (Oct. 2021), pp. 37–44 (cit. on pp. xx, 81, 133, 134).

[Vin+20] Diego Vinasco-Alvarez, John Samuel Samuel, Sylvie Servigne, and Gilles Gesquière. *From CityGML to OWL*. Technical Report. LIRIS UMR 5205, Sept. 2020 (cit. on pp. xx, 59, 133).

[Vin+21b] Diego Vinasco-Alvarez, John Samuel Samuel, Sylvie Servigne, and Gilles Gesquière. "Towards a semantic web representation from a 3D geospatial urban data model". In: May 2021 (cit. on pp. xiv, xx, 59, 76, 118, 133, 134).

[Wac+01] H. Wache, T. Vögele, U. Visser, et al. "Ontology-Based Integration of Information - A Survey of Existing Approaches". In: *In IJCAI'01 Workshop. on Ontologies and Information Sharing*. 2001 (cit. on p. 52).

[WPL19] Hao Wang, Yisha Pan, and Xiaochun Luo. "Integration of BIM and GIS in sustainable built environment: A review and bibliometric analysis". en. In: *Automation in Construction* 103 (July 2019), pp. 41–52 (cit. on p. 38).

[Whe07] David A Wheeler. *Why open source software/free software (OSS/FS, FLOSS, or FOSS)? Look at the numbers*. 2007 (cit. on p. 124).

[Xia+18] Guohui Xiao, Diego Calvanese, Roman Kontchakov, et al. "Ontology-Based Data Access: A Survey". en. In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. Stockholm, Sweden: International Joint Conferences on Artificial Intelligence Organization, July 2018, pp. 5511–5519 (cit. on p. 41).

[Yao+18] Zhihang Yao, Claus Nagel, Felix Kunde, et al. “3DCityDB - a 3D geodatabase solution for the management, analysis, and visualization of semantic 3D city models based on CityGML”. en. In: *Open Geospatial Data, Software and Standards* 3.1 (Dec. 2018), p. 5 (cit. on pp. 108, 118).

[ZG20] Olga Zalamea and G. García. “VALIDATION OF THE BCH-ONTOLOGY”. In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLIV-M-1-2020 (July 2020), pp. 497–504 (cit. on p. 39).

[ZL12] Jesper Zedlitz and Norbert Luttenberger. “Transforming Between UML Conceptual Models and OWL 2 Ontologies.” In: *Terra Cognita@ ISWC*. Citeseer. 2012, pp. 15–26 (cit. on pp. 47, 85, 86).

[ZBV18] Chi Zhang, Jakob Beetz, and de Vries. “BimSPARQL: Domain-specific functional SPARQL extensions for querying RDF building data”. In: *Semantic Web* (Aug. 2018), pp. 1–27 (cit. on pp. 78, 129, 138).

[Zis00] A. Zisman. “An overview of XML”. en. In: *Computing & Control Engineering Journal* 11.4 (Aug. 2000). Publisher: IET Digital Library, pp. 165–167 (cit. on p. 20).

List of Listings

Appendix: Data model results

A

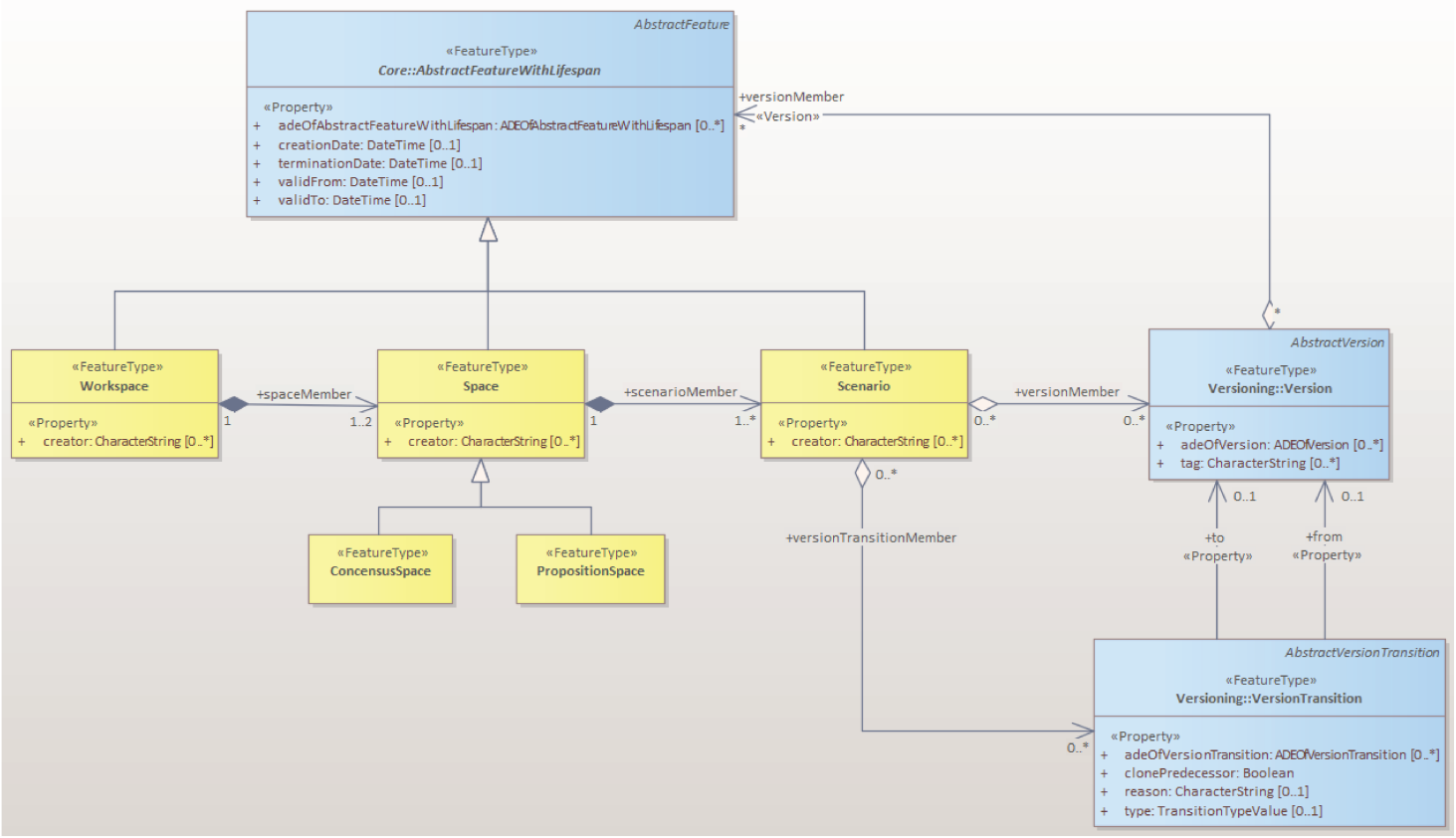

Figure A.1.: Workspace UML model as a CityGML 3.0 ADE, visualized in Enterprise Architect

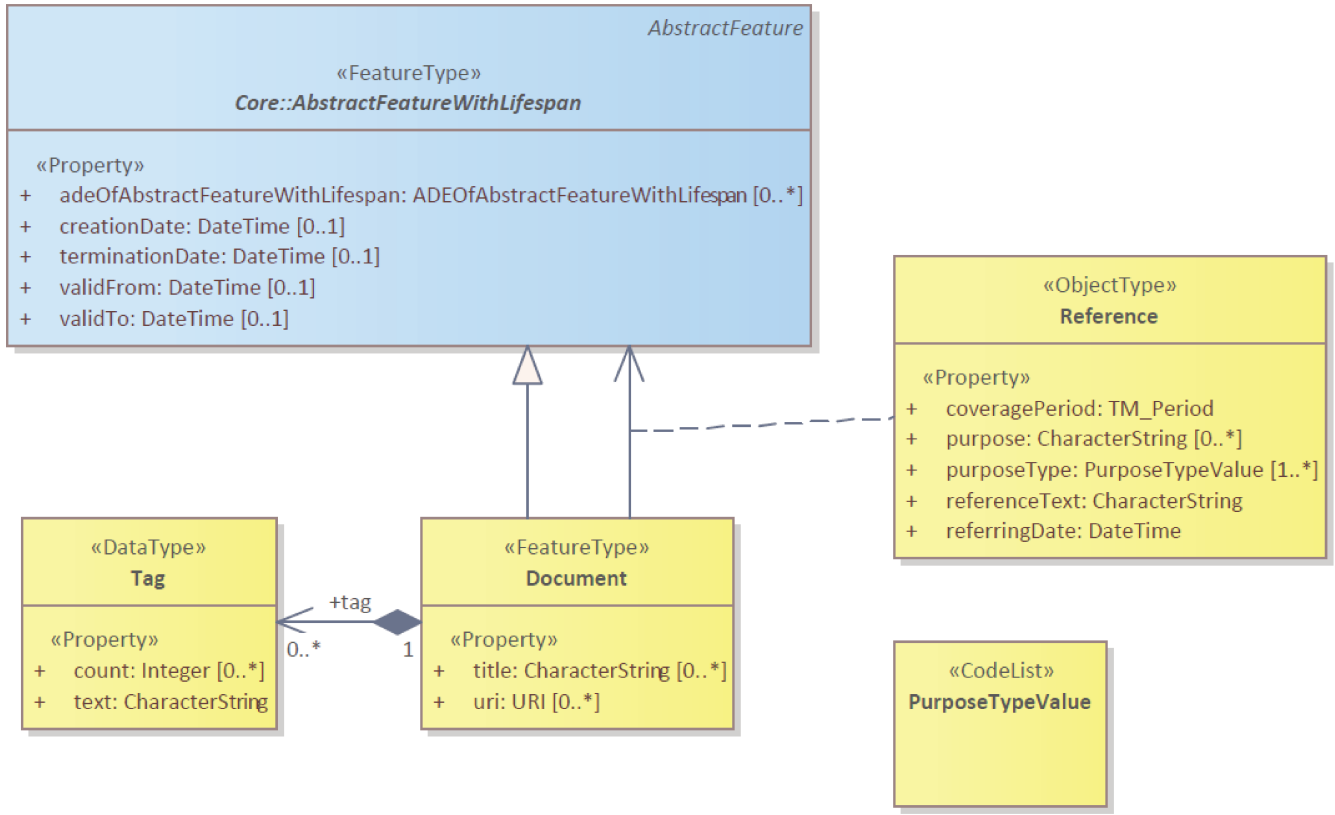

Figure A.2.: Document UML model as a CityGML 3.0 ADE

Appendix: Model-driven transformations

B

UML abstract syntax	OWL abstract syntax	Proposed conversion rule
Package	Ontology	- Name and structure as in UML - **Name and structure from tagged values**
Class	Class	- Direct conversion - **Subclass of ISO 19103 AnyFeature** - Mapping to external classes
Generalization	subclassOf	- **Direct conversion**
Abstract class		- **isAbstract annotation property** - DisjointUnion axiom
Primitive data type	Datatype-Property	- **Matched to XML Schema Datatypes**
Structured data type	Datatype-Property or ObjectProperty	- Mapping to a few external types, else new class - **Mapping to specified external types, else new class** - Mapping to all similar external types, else new class
Spatial data type	Data types defined in ISO 19107 and GeoSPARQL	- **Data types defined in the ISO 19107 ontology** - **Mapping to GeoSPARQL data types** - Extending GeoSPARQL
Enumeration	DataOneOf	- Direct conversion - **SKOS Concept Scheme**

Code list	Multiple options	- SKOS and allValuesFrom - **SKOS** - DataUnionOf and any value
Union	Multiple options	- Union - Intersections and Union - **Flattening** - Subproperty and ExactCardinality - DisjointClasses
Attribute and association role	Property	- Simple conversion - Globalization by similarity matching - **Globalization by prefix** - Global attributes with domain AnyFeature - Globalization by manual matching - Mapping to external properties
Simple association	Domain and range	- **Direct conversion**
Aggregation		- Hierarchy of properties - **aggregationType annotation property**
Composition		- Hierarchy of properties - InverseFunctional - **aggregationType annotation property**

Table B.1.: A summary of (GFM-based) UML to OWL conversion rules from Jetlund et al. [JOH19]. The transformation approaches chosen for formalizing the CityGML UML models are highlighted in bold.

C Appendix: Description Logic and Rules

C.1 Namespaces

Prefix	URI
xs, xsd	http://www.w3.org/2001/XMLSchema#
xslt	http://www.w3.org/1999/XSL/Transform
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs	http://www.w3.org/2000/01/rdf-schema#
owl	http://www.w3.org/2002/07/owl#
swrl	http://www.w3.org/2003/11/swrl#
skos	http://www.w3.org/2004/02/skos/core#
geo, gsp	http://www.opengis.net/ont/geosparql#
gml	http://www.opengis.net/gml
gmlowl	http://www.opengis.net/ont/gml
time	http://www.w3.org/2006/time#
time_ext	https://dataset-dl.liris.cnrs.fr/rdf-owl-urban-data-ontologies/Ontologies/Time/time-extension#
core	https://dataset-dl.liris.cnrs.fr/rdf-owl-urban-data-ontologies/Ontologies/CityGML/3.0/core#
bldg	https://dataset-dl.liris.cnrs.fr/rdf-owl-urban-data-ontologies/Ontologies/CityGML/3.0/building#
vers	https://dataset-dl.liris.cnrs.fr/rdf-owl-urban-data-ontologies/Ontologies/CityGML/3.0/versioning#
wksp	https://dataset-dl.liris.cnrs.fr/rdf-owl-urban-data-ontologies/Ontologies/Workspace/3.0/workspace#
type	https://dataset-dl.liris.cnrs.fr/rdf-owl-urban-data-ontologies/Ontologies/Workspace/3.0/transactiontype#

Table C.1.: A list of Namespace prefixes and URIs used in this article. These are used for the XML namespaces and IRI namespaces in the XML and RDF listings of this work.

C.2 A basic description logic language

A basic description logic language, $\mathcal{AL}$, is proposed in Baader and Nutt [BN03]. Given atomic concepts A and B, an atomic role R, and concept descriptions C and D, the following conceptual formalism are provided in $\mathcal{AL}$ according to the following syntax rules:

$$
\begin{aligned}
C, D \rightarrow A &\mid \text{atomic concept} \\
\top &\mid \text{universal/top concept} \\
\bot &\mid \text{bottom concept} \\
\neg A &\mid \text{atomic negation} \\
C \sqsubseteq D &\mid \text{subsumption} \\
C \sqcap D &\mid \text{intersection} \\
\forall R.C &\mid \text{value restriction} \\
\exists R.\top &\mid \text{limited existential quantification} \\
\exists R.C &\mid \text{full existential quantification}
\end{aligned}
$$

C.3 SWRL rules for inferring temporal relations

Name	SWRL Rule
Beginning and End	`time:TemporalEntity(?a), time:hasBeginning(?a, ?i1), time:Instant(?i1), time:inXSDDateTimeStamp(?i1, ?t1), time:hasEnd(?a, ?i2), time:Instant(?i2), time:inXSDDateTimeStamp(?i2, ?t2), greaterThan(?t1, ?t2) -> owl:Nothing(?a)`
Before and After	`time:hasEnd(?a, ?i1), time:Instant(?i1), time:inXSDDateTimeStamp(?i1, ?t1), time:hasBeginning(?b, ?i2), time:Instant(?i2), time:inXSDDateTimeStamp(?i2, ?t2), lessThan(?t1, ?t2) -> time:before(?a, ?b)`
Instant	`time:TemporalEntity(?a), time:hasBeginning(?a, ?i1), time:Instant(?i1), time:inXSDDateTimeStamp(?i1, ?t1), time:hasEnd(?a, ?i2), time:Instant(?i2), time:inXSDDateTimeStamp(?i2, ?t2), equal(?t1, ?t2) -> time:Instant(?a), time:inXSDDateTimeStamp(?a, ?t1)`

Proper Interval	`time:TemporalEntity(?a), time:hasBeginning(?a, ?i1), time:Instant(?i1), time:inXSDDateTimeStamp(?i1, ?t1), time:hasEnd(?a, ?i2), time:Instant(?i2), time:inXSDDateTimeStamp(?i2, ?t2), lessThan(?t1, ?t2) -> time:ProperInterval(?a)`
Instant-Interval Starts	`time:Instant(?a), time:inXSDDateTimeStamp(?a, ?t1), time:ProperInterval(?b), time:hasBeginning(?b, ?i), time:Instant(?i), time:inXSDDateTimeStamp(?i, ?t2), equal(?t1, ?t2) -> time_ext:starts(?a, ?b)`
Instant-Interval Finishes	`time:Instant(?a), time:inXSDDateTimeStamp(?a, ?t1), time:ProperInterval(?b), time:hasEnd(?b, ?i), time:Instant(?i), time:inXSDDateTimeStamp(?i, ?t2), equal(?t1, ?t2) -> time_ext:finishes(?a, ?b)`
Instant-Instant Equals	`time:Instant(?a), time:inXSDDateTimeStamp(?a, ?t1), time:Instant(?b), time:inXSDDateTimeStamp(?b, ?t2), equal(?t1, ?t2) -> time_ext:equals(?a, ?b)`
Interval before	`time:ProperInterval(?a), time:hasEnd(?a, ?i1), time:Instant(?i1), time:inXSDDateTimeStamp(?i1, ?t1), time:ProperInterval(?b), time:hasBeginning(?b, ?i2), time:Instant(?i2), time:inXSDDateTimeStamp(?i2, ?t2), lessThan(?t1, ?t2) -> time:intervalBefore(?a, ?b)`
Interval equals	`time:ProperInterval(?a), time:hasBeginning(?a, ?i1), time:Instant(?i1), time:inXSDDateTimeStamp(?i1, ?t1), time:hasEnd(?a, ?i2), time:Instant(?i2), time:inXSDDateTimeStamp(?i2, ?t2), time:ProperInterval(?b), time:hasBeginning(?b, ?i3), time:Instant(?i3), time:inXSDDateTimeStamp(?i3, ?t3), time:hasEnd(?b, ?i4), time:Instant(?i4), time:inXSDDateTimeStamp(?i4, ?t4), equal(?t1, ?t3), equal(?t2, ?t4) -> time:intervalEquals(?a, ?b)`

Interval overlaps	`time:ProperInterval(?a), time:hasBeginning(?a, ?i1), time:Instant(?i1), time:inXSDDateTimeStamp(?i1, ?t1), time:hasEnd(?a, ?i2), time:Instant(?i2), time:inXSDDateTimeStamp(?i2, ?t2), time:ProperInterval(?b), time:hasBeginning(?b, ?i3), time:Instant(?i3), time:inXSDDateTimeStamp(?i3, ?t3), time:hasEnd(?b, ?i4), time:Instant(?i4), time:inXSDDateTimeStamp(?i4, ?t4), lessThan(?t1, ?t3), lessThan(?t2, ?t4), greaterThan(?t2, ?t3) -> time:intervalOverlaps(?a, ?b)`
Interval meets	`time:ProperInterval(?a), time:hasBeginning(?a, ?i1), time:Instant(?i1), time:inXSDDateTimeStamp(?i1, ?t1), time:hasEnd(?a, ?i2), time:Instant(?i2), time:inXSDDateTimeStamp(?i2, ?t2), time:ProperInterval(?b), time:hasBeginning(?b, ?i3), time:Instant(?i3), time:inXSDDateTimeStamp(?i3, ?t3), time:hasEnd(?b, ?i4), time:Instant(?i4), time:inXSDDateTimeStamp(?i4, ?t4), equal(?t2, ?t3) -> time:intervalMeets(?a, ?b)`
Interval during	`time:ProperInterval(?a), time:hasBeginning(?a, ?i1), time:Instant(?i1), time:inXSDDateTimeStamp(?i1, ?t1), time:hasEnd(?a, ?i2), time:Instant(?i2), time:inXSDDateTimeStamp(?i2, ?t2), time:ProperInterval(?b), time:hasBeginning(?b, ?i3), time:Instant(?i3), time:inXSDDateTimeStamp(?i3, ?t3), time:hasEnd(?b, ?i4), time:Instant(?i4), time:inXSDDateTimeStamp(?i4, ?t4), greaterThan(?t1, ?t3), lessThan(?t2, ?t4) -> time:intervalDuring(?a, ?b)`
Interval starts	`time:ProperInterval(?a), time:hasBeginning(?a, ?i1), time:Instant(?i1), time:inXSDDateTimeStamp(?i1, ?t1), time:hasEnd(?a, ?i2), time:Instant(?i2), time:inXSDDateTimeStamp(?i2, ?t2), time:ProperInterval(?b), time:hasBeginning(?b, ?i3), time:Instant(?i3), time:inXSDDateTimeStamp(?i3, ?t3), time:hasEnd(?b, ?i4), time:Instant(?i4), time:inXSDDateTimeStamp(?i4, ?t4), equal(?t1, ?t3), lessThan(?t2, ?t4) -> time:intervalStarts(?a, ?b)`

Interval finishes	`time:ProperInterval(?a), time:hasBeginning(?a, ?i1), time:Instant(?i1), time:inXSDDateTimeStamp(?i1, ?t1), time:hasEnd(?a, ?i2), time:Instant(?i2), time:inXSDDateTimeStamp(?i2, ?t2), time:ProperInterval(?b), time:hasBeginning(?b, ?i3), time:Instant(?i3), time:inXSDDateTimeStamp(?i3, ?t3), time:hasEnd(?b, ?i4), time:Instant(?i4), time:inXSDDateTimeStamp(?i4, ?t4), equal(?t2, ?t4), greaterThan(?t1, ?t3) -> time:intervalFinishes(?a, ?b)`

Table C.2.: Proposed SWRL rules for inferring basic temporal relations. The reader should note that properties in these rules are written with global scope naming conventions for brevity. The actual proposed rules use locally scoped naming conventions.

Name	SWRL Rule
14a	`core:AbstractFeatureWithLifespan(?f), core:validFrom(?f, ?v1), core:validFrom(?f, ?v2) -> lessThanOrEqual(?v1, ?v2)`
14b	`core:AbstractFeatureWithLifespan(?f), core:creationDate(?f, ?e1), core:terminationDate(?f, ?e2) -> lessThanOrEqual(?e1, ?e2)`
22	`vers:Transaction(?t), vers:type(?t, ?y), vers:oldFeature(?t, ?f1), vers:newFeature(?t, ?f2), equal(?y, type:insert) -> empty(?f1), core:AbstractFeatureWithLifespan(?f2)`
23	`vers:Transaction(?t), vers:type(?t, ?y), vers:oldFeature(?t, ?f1), vers:newFeature(?t, ?f2), equal(?y, type:delete) -> core:AbstractFeatureWithLifespan(?f1), empty(?f2)`
24 (unchanged)	`vers:Transaction(?t), vers:type(?t, ?y), vers:oldFeature(?t, ?f1), vers:newFeature(?t, ?f2), equal(?y, type:unchanged), core:AbstractFeatureWithLifespan(?f1), core:validTo(?f1, ?d1), core:AbstractFeatureWithLifespan(?f2), core:validFrom(?f2, ?d2) -> lessThanOrEqual(?d1, ?d2)`

24 (reided)	`vers:Transaction(?t), vers:type(?t, ?y), vers:oldFeature(?t, ?f1), vers:newFeature(?t, ?f2), equal(?y, type:re-ided), core:AbstractFeatureWithLifespan(?f1), core:featureID(?f1, ?i1), core:validTo(?f1, ?d1), core:AbstractFeatureWithLifespan(?f2), core:featureID(?f2, ?i2), core:validFrom(?f2, ?d2) -> lessThanOrEqual(?d1, ?d2), notEqual(?i1, ?i2)`
24 (modified)	`vers:Transaction(?t), vers:type(?t, ?y), vers:oldFeature(?t, ?f1), vers:newFeature(?t, ?f2), equal(?y, type:modified), core:AbstractFeatureWithLifespan(?f1), core:featureID(?f1, ?i1), core:validTo(?f1, ?d1), core:AbstractFeatureWithLifespan(?f2), core:featureID(?f2, ?i2), core:validFrom(?f2, ?d2) -> lessThanOrEqual(?d1, ?d2), notEqual(?i1, ?i2)`
24 (fused)	`vers:Transaction(?t), vers:type(?t, ?y), vers:oldFeature(?t, ?f1), vers:newFeature(?t, ?f2), equal(?y, type:fused), core:AbstractFeatureWithLifespan(?f1), core:validTo(?f1, ?d1), core:AbstractFeatureWithLifespan(?f2), core:validFrom(?f2, ?d2) -> lessThanOrEqual(?d1, ?d2)`
24 (subdivided)	`vers:Transaction(?t), vers:type(?t, ?y), vers:oldFeature(?t, ?f1), vers:newFeature(?t, ?f2), equal(?y, type:subdivided), core:AbstractFeatureWithLifespan(?f1), core:validTo(?f1, ?d1), core:AbstractFeatureWithLifespan(?f2), core:validFrom(?f2, ?d2) -> lessThanOrEqual(?d1, ?d2)`
27	`vers:Version(?v), vers:versionMember(?v, ?f), core:AbstractFeatureWithLifespan(?f), core:validFrom(?v, ?vv1), core:validTo(?v, ?vv2), core:validFrom(?f, ?vf1), core:validTo(?f, ?vf2) -> lessThanOrEqual(?vf1, ?vv1), greaterThanOrEqual(?vf2, ?vv2)`
31	`vers:VersionTransition(?vt), vers:from(?vt, ?v1), vers:to(?vt, ?v2), vers:Version(?v1), core:validTo(?v1, ?t1), vers:Version(?v2), core:validFrom(?v2, ?t2), -> lessThanOrEqual(?t1, ?t2)`

35	`wksp:Scenario(?s), wksp:versionTransitionMember(?s, ?vt), vers:VersionTransition(?vt), vers:from(?vt, ?v1), vers:to(?vt, ?v2) -> wksp:versionMember(?s, ?v1), wksp:versionMember(?s, ?v2)`
36	`wksp:Scenario(?s), wksp:versionMember(?s, ?v1), vers:Version(?v1), wksp:versionMember(?s, ?v2), vers:Version(?v2) -> vers:VersionTransition(?vt), vers:from(?vt, ?v1), vers:to(?vt, ?v2)`
37	`wksp:Scenario(?s), wksp:versionMember(?s, ?v1), vers:Version(?v1), core:validFrom(?v1, ?vf1), core:validTo(?v1, ?vt1), wksp:versionMember(?s, ?v2), vers:Version(?v2), core:validFrom(?v2, ?vf2), core:validTo(?v2, ?vt2), equal(?vf1, ?vf2), equal(?vt1, ?vt2) -> equal(?v1, ?v2)`
38	`wksp:Scenario(?s), wksp:versionTransitionMember(?s, ?vt1), vers:VersionTransition(?vt1), core:validFrom(?vt1, ?vtf1), core:validTo(?vt1, ?vtt1), wksp:versionTransitionMember(?s, ?vt2), vers:VersionTransition(?vt2), core:validFrom(?vt2, ?vtf2), core:validTo(?vt2, ?vtt2), equal(?vtf1, ?vtf2), equal(?vtt1, ?vtt2) -> equal(?vt1, ?vt2)`
39	`wksp:Scenario(?s), wksp:versionTransitionMember(?s, ?vt1), vers:VersionTransition(?vt1), wksp:versionTransitionMember(?s, ?vt2), vers:VersionTransition(?vt2), vers:to(?vt1, ?v1), vers:from(?vt2, ?v1) -> wksp:versionMember(?s, ?v1), vers:Version(?v1)`

Table C.3.: Proposed SWRL rules for temporal validation of scenarios of urban evolution based on the CityGML 3.0 Versioning module [KCK20] and the Workspace extension [SSG20]. Rules names follow the same numbering as [SSG20]. The reader should note that properties in these rules are written with global scope naming conventions for brevity. The actual proposed rules use locally scoped naming conventions.