TOWARDS THE INTEROPERABILITY OF BIM AND GIS BY BUILDING ONTOLOGIES USING SEMANTIC WEB TECHNOLOGY

AMAN ULLAH USMANI

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

MASTER OF SCIENCE

GRADUATE PROGRAM IN DEPARTMENT OF EARTH AND SPACE SCIENCE
AND ENGINEERING
YORK UNIVERSITY
TORONTO, ONTARIO

MAY 2021

Abstract

Over the past decade, domains of Building Information Modelling (BIM) and Geographic Information Systems (GIS) have been investigated to establish interoperability for vital geospatial information exchange. Increasing interest in this research area has driven a multitude of integration techniques that tend to incorporate other technologies like the Internet of Things (IoT). Among these methodologies, Semantic Web Technology has shown promising results towards achieving BIM-GIS semantic interoperability. With its natural ability to integrate heterogeneous information, the semantic web fabricates a mesh of data that may include IoT sources. However, before integrating heterogeneous information of BIM and GIS, presenting it in the semantic web is challenging. The core of semantic web enables building vocabularies as ontology by using Web Ontology Language (OWL) and store data in Resource Description Framework (RDF). Defining semantically rich ontologies is a complicated, time-consuming and error-prone procedure. In contrast, rich ontology models are essential for accurately mapping cross-domain features to address the interoperability problem. While there are approaches available to produce OWL models, they lack readily available tools or require excessive efforts in their implementations. Our study proposes a comprehensive conceptual framework to establish interoperability between BIM and GIS data-formats, link-able with sensors information using semantic web technology stack. From the three-module framework, study focuses specifically on the first-module of formal and automatic ontology generation using XML Schema Document (XSD) to OWL transformation patterns. It implements Janus and PIXCO frameworks from the ontology generation state-of-the-art with improvements and enhancements in their XSD to OWL transformations. To validate this study, a prototype named as EPIXCO (Enhanced Pattern Identification for XSD Conversion to OWL) is implemented to evaluate and analyze generated ontology models. The EPIXCO prototype utilizes a defined ground truth matrix for results evaluation and comparative analysis with standard ontologies, stating generated models have rich ontology axioms. These generated ontology models can be sourced for aligning information to establish cross-domain ontologies and integrated geospatial information. The future applications of this mesh of data lead to knowledge-graph and smart geospatial data, available for urban environment analysis and smart city applications.

April 20th, 2021

To my lovely parents!

Acknowledgements

I have so many people to acknowledge for not just being a part of this journey, but more importantly, I am thankful for them being part of my life. I wouldn't be able name each and everyone of them here, but in my life scroll they are well written, with bold fonts.

First of all, I would praise ALLAH Almighty. Alhamdulillah, I wouldn't have achieved anything in my life without HIS will and blessings.

I would like to express deepest gratitude to my supervisor and mentor, Dr. Mojgan Jadidi, for believing in me right from the first interview. Her supervision, support and a constant source of inspiration has paved the path to achieve this goal. Its been a pleasure.

I would take this opportunity to acknowledge the sheer guidance and throughout motivation of Dr. Gunho Sohn, my supervisory committee member, with his vital and remarkable knowledge of my research. I am very humble and thankful to my exam committee members, Dr. Magdalena Krol and Dr. Manos Papagelis, for their acceptance to assess and adding value to my research.

Dr. Muhammad Usman, this journey wouldn't even have started without mentioning this person. A brother, dearest old friend, mentor, and a motivational catalyst.

Thank you to a friend, Maryam Jameela, for being a power house and a constant support system. Afnan Ahmad, an old friend reunited, for being adventurous, supportive, and bearing my personalities. A powerful couple, Zahra Arjmandi and Ali Akbar, thank you for sharing wonderful life perspectives while preparing delicious Iranian food and tea.

Also, thank you to the York University for giving me the opportunity and my GeoVA lab colleagues and friends; Sepideh Dibadin, Xuyang Han, Amirhossein Nourbakhshrezaei, Hamid Kiavarz Moghaddam, and Sarah Kaykhosravi for sharing wonderful inspirations.

A huge shout-out to the amazing people; Ima Ituen, Myrto Tzamali, Mahya Rad, Madiha Khalid, Saim Mehmood, Dr. Jungwon Kang, Dr. Ali Baligh, Kivanc Babacan, Sun Park, Jacob Yoo, Hyungju Lee, Muhammad Kamran, Azad Maghooli, Cosette Gilmour and Agata Szeremeta.

The two brothers from another mother, Umair Khan and Uzair Khan, were home away from home. A heartwarming thank you to my genius best friends, Hassan Ali Anjum, Syed Muhammad Umair, and Imran Khalil, a.k.a. Chaar Yaar Charon Bekaar (Four Friends All Four Useless) for moulding me as a better person and a programmer.

Last, but not the least, and humbly above everyone, my little big family. I can't be thankful enough of the parents I have been blessed with. My father, Mr. H. Usmani, guiding me in every bright and dark moments of life. My mother, Mrs. Uzma Tayyaba, her love, trust and strength made me what I am today. My six siblings; Nazia, Asma, Sheharyar, Huma, Saba and Sana, for being thrusters of my life. I wouldn't have made it this far without my amazing family. Love you all!

Table of Contents

List of Tables

List of Figures

List of Algorithms

List of XSD Listings

Chapter 1

Introduction

Over the past decade, virtual city applications have been reshaping the future of the metropolitan cities. They hold great potential in the formulation of sustainable, effective strategies and policies by creating, modelling and visualizing 3D virtual environments with substantial geospatial information. The information available from such diverse sources is a collectively efficient and integral part of Smart Cities (Amini et al. 2019). Smart City is an urban development vision that uses data and technology for managing financial, environmental and social impacts of recent and future growth in the metropolitan regions by visualizations, simulations and drawing knowledge from the information systems (Carneiro et al. 2019). A substantial provision for such emergence is the data collected from cutting-edge technologies to source information and lay the foundations of a smart city. Such foundations necessitate the fusion of data from broad-spectrum domains revealing Smart Data (Howell et al. 2017), which unravel not only as an idea of the smart city but also machine-understandable, semantic information.

Motivation and Aim

The future of urban planning and development is unfolded as an important research area to build smart cities (Jamei et al. 2017). The purpose of a smart city manifests virtual representation of detailed geospatial and sensor information as a digital twin, which relies on data collected from Architecture, Engineering, Construction and Facility Management (AEC/FM), geospatial and Internet of Things (IoT) domains.

The open-standard data collection in these domains and conventional representations using cutting-edge technologies, like Geographic Information Systems (GIS) and Building Information Modelling (BIM), has accelerated the emergence of this research area (Fosu et al. 2015, Ma and Ren 2017, Wang et al. 2019). Not to mention, information collected from internet-connected devices – referred to as IoT – has equally engaged incorporating sensors data with BIM and GIS technologies. Interlinking information from smart devices within building and geospatial context are favourable. Though, an extensively integrated system of BIM, GIS and IoT requires preceding integrated knowledge of BIM and GIS. Several studies have recently investigated the benefits of effective integration of BIM and GIS (even with IoT) in the perspective of urban planning, development, and analysis (Song et al. 2017). However, achieving integration among two distinct domains of BIM and GIS itself is quite challenging (Liu et al. 2017). Traditional methods for integrating BIM and GIS have surfaced issues like data incompatibility, misinterpretation, and absence of information.

Although the foundations of BIM and GIS systems have been developed to address their respective AEC/FM and geospatial problem domains, they have progressed, individually, stemming overlapping features with the technological advancements and user specifications. Data in each of these systems represents vital information. Thus, to address the redundant data and vitalize cross-domain features, bridging the gap between their heterogeneous data-formats without an information loss becomes a critical challenge (Liu et al. 2017).

On the contrary, Semantic Web Technology methods have shown promising results due to their natural ability towards heterogeneous data integration and eventually devising Smart Data. However, existing semantic integration methods are mainly limited to the manual or semi-automatic processes for a core feature of the semantic web, ontology. Relatively, mostly automatic processes overlook ontology particularities, which is the main building block in the semantic web. Devising a comprehensive automated method of generating BIM and GIS ontologies to achieve interoperability between these domains remains unanswered. This study aims to provide a novel framework using semantic web technique to achieve interoperability among BIM and GIS, with a focal point of the automatic ontology generation process.

Problem Domain

The process of ontology development is very complex, where expressing correct semantics of data in an ontological representation itself requires base-knowledge. The literature of ontology generation highlights semi or even fully-automated processes; however, their frameworks are primarily manifested for corpus-based approaches and have a minimal extension of supporting existing ontologies. Also, the methods lack the related tools readily available for ontology generation. Therefore, creating complicated geospatial ontology is a laborious task.

Furthermore, for fusing information of heterogeneous datasets, diverse ontologies of building and geospatial information systems require mapping techniques that fulfill interlinking of their entities to obtain cross-domain integrated ontology, which characterizes cross-domain integrated data for information analysis and knowledge-graph applications. Such ontology mapping approaches are generally limited to corpus-based studies, which further requires investigating alignment knowledge for considerable information in the building and geospatial domain. Most of these approaches adapted for the geospatial ontology alignment are either manual or lacking in mapping across entities.

Hence, the ontology generation and ontology alignment of BIM and GIS data-formats are essential courses of action to investigate BIM and GIS interoperability before being integrated with the IoT information for smart city applications and smart data solutions.

Broad Research Goals

The semantic web approach establishes information exchange across independent and fundamentally incompatible data formats in the multitude of BIM and GIS integration methodologies. The general goal of this research project aims for a semantically interoperable and seamlessly integrated system of BIM and GIS data with IoT by emanating semantic web technology approach that can be divided in three goals. However, in this study the focus is narrowed down and explicit to the integration of BIM and GIS as the first goal with its in-depth implementation and later provides a brief synopsis on two following goals proposed for future work:

- Generating ontology of BIM and GIS data: To achieve semantic integration, ontologies of BIM and GIS need to be generated for schema structures of their standardized XML-based data-formats, IFC-XML and CityGML, respectively. A formal method with schema modelling, and defined patterns to transform elements from XML schema format to corresponding ontological representation, will be determined to induce enriched ontologies of BIM and GIS.

- Mapping of cross-domain ontologies: The generated ontology models of BIM and GIS data will be further used to identify corresponding mappings between their entities and properties. The machine learning techniques will investigate the linking of cross-domain elements to obtain integrated geospatial ontology, serving as a foundation for saturating information from cross-domain building and geospatial datasets.

- Generation and optimization of semantic graph: Finally, to manifest integrated information, data from IFC-XML and CityGML is adjoined with integrated cross-domain ontology resulting graph in RDF format. Furthermore, IoT data will be inter-linked with generated RDF graphs for semantically rich geospatial and sensor information. RDF graphs are sizeable and require optimization. Thus, we propose clustering and community optimization techniques for manipulating RDF graphs.

Research Objectives

This study focuses on first-goal of the broad research spectrum (mentioned earlier) for semantic interoperability of BIM and GIS. The major research objectives of this study are to establish a formal methodology to transform XML schemas into ontology models. The formal method addresses the problem of automatic ontology generation for large XML Schema Document (XSD), like ifcXML and CityGML of BIM and GIS, using defined correspondence rules. The general framework extracts information from XSD schema mapped with defined sets of patterns to build ontology models. Perspective of framework objectives can be viewed as determination and implementation of three major steps: define formal models of extracted information from XSD document and transformation patterns, identifying pattern among given XSD schema and generating the respective cohesive ontology model.

Research Contributions

This research outcome mainly contributes towards the investigation of interoperability between BIM and GIS in the following collaborations and publications:

- *The ISPRS-EuroSDR GeoBIM Benchmark 2019* – Participation to investigate interoperability of BIM and GIS for GeoBIM Benchmark project (more details in

Chapter 4).

- *Automatic Ontology Generation of BIM and GIS Data* (Usmani et al. 2020) – XML schema-based formal procedure for ontology generation (more in Chapter 5).

- *Formal Transformation of XML Schema to Ontology Models* – Publication in preparation for the developed EPIXCO framework to build ontology models automatically (more in Chapter 6).

Thesis Outline

This Chapter provides an introduction to the research project with motivation to achieve interoperability of BIM and GIS data and potentially provides a solution to the automatic ontology generation problem. The outline of this thesis for the rest of the document is organized as follows:

- Chapter 2 presents an introduction to BIM and GIS technologies. It highlights the need and trends of the research studies conducted for the BIM and GIS integration with a fundamental perspective of their integration gap and enlisting comprehensive literature with the role of the semantic web in achieving integration.

- Chapter 3 delves with a brief overview of ontology, semantic web and its technology stack as a promising technique for achieving interoperability among BIM and GIS. It portrays the XML schema components to understand XSD designs and the approaches proposed for XSD to OWL transformations.

- Chapter 4 features a study that investigates standard IFC and CityGML data-formats with uncovering the role of existing preparatory software tools and procedures on these data standards towards BIM and GIS interoperability.
- Chapter 5 features a novel framework design of BIM and GIS data integration using semantic web technologies. It defines an ontology generation process, as the core of this research study, an extensive framework to transform XML schema into an OWL model.
- Chapter 6 provides the implementation of the proposed framework for ontology generation and validates the methodology with experiments and results of selected XML schema structures.
- Finally, Chapter 7 addresses the conclusion of this research study by featuring main contributions in this dissertation, highlighting the limitations of the implemented system, and possible recommendations to possible future extensions.

Chapter 2

Background and Related Work

This chapter presents the theoretical background of the focused technologies, BIM and GIS, in this dissertation. It explicitly focusing on their integration area of interest with modest involvement of sensors technology, the motive of their integration, and the applications of research conducted this study. The Chapter is organized as follows: introduction to the fundamentals of BIM and GIS domains is provided in Section 2.1; Section 2.2 discusses dissimilarities and rudimentary factors of each domain and their supportive data-formats; Section 2.3 delves in extensive literature review of BIM and GIS integration methods and ensues the potential of their integrated applications in Section 2.4; Section 2.5 specifies the role of Semantic Web Technology adapted in approaches towards achieving interoperability among BIM and GIS; and lastly, in the Section 2.6, the Chapter is summarized by highlighting problems in literature integration methods, and recognizes an integration approach with possible potential BIM-GIS interoperability solution.

2.1 Fundamentals of BIM and GIS

2.1.1 Building Information Modelling (BIM)

BIM is the process to create, store and manage information related to infrastructure, particularly buildings, throughout their life cycle (Eastman et al. 2011). It comprises a set of interacting policies and technologies highly used by the Architecture, Engineering, Construction, and Facility Management (AEC/FM) industry for infrastructures with rich 3D geometric and semantic information (Azhar 2011, Zhu et al. 2018).

Besides efficient tools for modelling, analyzing and managing detailed 3D models, based on its enhanced design and construction techniques, BIM offers additional dimensions such as cost, schedule, accessibility, security, maintainability and energy simulation (Taylor and Bernstein 2009). BIM methodologies are also characterized by their *Level of Development* (LOD), different from *Level of Details* (LoD) in GIS, and defines LOD100-LOD500 to monitor the design progress. Figure 2.1 presents basic BIM models with LOD200 information. BIM moved to mainstream technologies over a while, disrupting the traditional building and construction designing platforms.

The buildingSmart Industrial Foundation Class (IFC) is the most comprehensive and popular open-standard data format for BIM models widely accepted and supported by most BIM-related software in the AEC industry (Deng et al. 2016). IFC models are presented in EXPRESS data specification language, where the IFC entities in these models are referred by line number, and the most commonly used IFC formats are IFC2x3

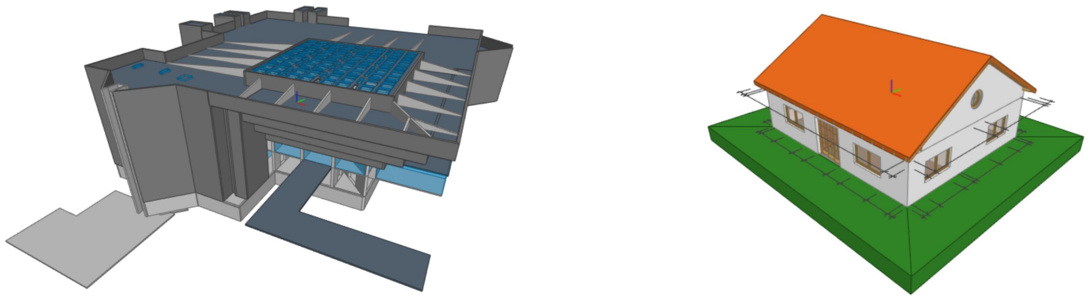

(a) (b)

Figure 2.1: BIM models example visualization in BIM Vision: (a) Scott Library (IFC2x3) at York University Campus and (b) FZK Haus (IFC 4) by Applied Computer Science (IAI) at the Karlsruhe Institute of Technology (KIT).

and IFC4 (Pauwels and Terkaj 2016). The information exchange format of IFC also has an Extensible Markup Language (XML) version, ifcXML, more flexible for BIM methodologies in XML-based environments. However, it is not widely used in industry as EXPRESS-based IFC due to the verbosity of XML document format and performance limitations (Pauwels et al. 2017a). The buildingSmart releases for the latest versions[1] of IFC4 specifications have also made steps towards interoperability with GIS by including new elements like *"IfcGeographicElement"* and *"IfcGeographicElementType"* (Liu et al. 2017). However, BIM and its supporting modelling formats do not include surrounding information and have limited spatial inquiry (Irizarry and Karan 2012a). Nevertheless, IFC is a complete but complex standard, providing several entities for describing the buildings and numerous solutions available to model BIMs.

[1]https://technical.buildingsmart.org/standards/ifc/ifc-schema-specifications/

2.1.2 Geographic Information Systems (GIS)

GIS is an information system with a framework to collect, manage, analyze, and present spatial data for performing spatial analysis and visualization for better and informed decision-making practices (Hanchette 2003). The GIS decision-support system provides powerful tools for storing and manipulating spatial information and has been long used for modelling large environments, integrating and visualizing assets information with geo-referencing (Breunig et al. 2020). The association of location information with data provides GIS components to feature spatial and temporal analysis (Song et al. 2017).

In the geospatial domain, City Geographic Markup Language (CityGML) (Gröger and Plümer 2012) is currently the most comprehensive standard of urban information with semantic 3D modelling. It has emerged as a prominent exchange format towards integrating BIM and GIS (Amirebrahimi et al. 2016, Malinverni et al. 2019). CityGML, specified as XML-based Geographic Markup Language (GML) grammar, is an open-data standard by Open Geospatial Consortium (OGC) that structures the information about cities and contextual features. The detailed standard introduces the concept of *Level of Detail* (LoD), including four levels LoD0-LoD4, that ranges objects and data from landscape to interior architectural model. Compared with the first generation of CityGML, the second generation offers richer 3D models not just of buildings but also tunnels and bridges, hence closing the gap with the BIM infrastructural elements (Yao et al. 2018).

With significantly growing computing power, tool and technologies, and advancements in data acquisition methods, the geospatial context in GIS systems have developed from 2D information to complex macro-scale 3D models with automated workflows (Becker et al. 2009, Biljecki et al. 2015).

2.2 BIM and GIS as Correlated Technologies

The principle system design of BIM and GIS are to exploit their domain-oriented information for enhanced designs, management and analysis (Song et al. 2017). Data in 3D-space collected using laser point cloud technology provides virtual models of the physical environment utilized in 3D GIS to create city models with explicit geometry and semantics (Yao et al. 2018). Eventually providing adequate information and visualization for simulations and analytical studies of buildings and surrounding infrastructures in an as-built environment (Biljecki et al. 2016b).

Mutual Extrapolation: BIM is a digital representation of detailed infrastructure elements in the AEC domain. On the contrary, GIS focuses on collecting, storing and analyzing geospatial data at a large scale. Both of these domains were developed with fundamentally different purposes. However, over the last decade, increased computing power and growing user requirements of linking cross-domain specific information have led tools and technologies of both BIM and GIS domains to incorporate and process element features at the outset across domain (El-Mekawy and Östman 2010). Multiple applications in AEC domain require adjoining information for pre and post-construction phases, involving GIS data. For example, BIM processes incorporate surrounding features alongside

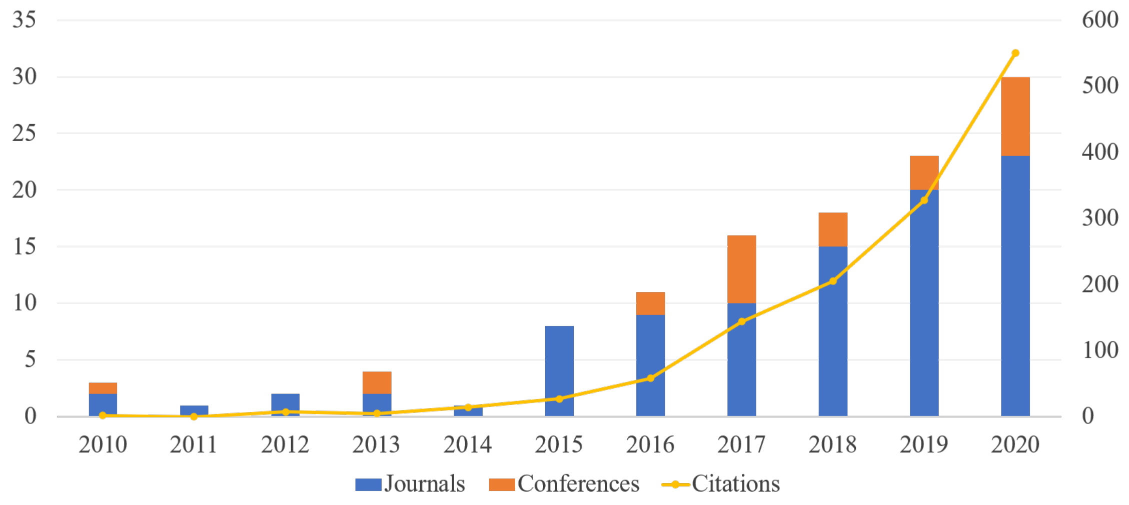


Figure 2.2: Web of Science (https://apps.webofknowledge.com/) report of publications and citations regarding BIM and GIS Integration.

detailed infrastructural elements to support analysis of infrastructure with environmental information, generally retrieved from other spatial or existing GIS tools (Irizarry and Karan 2012b, Zhao et al. 2019). In contrast, GIS models are increasingly embodying detailed structural and building information in city-level information (Amirebrahimi et al. 2016) to share semantics of individual elements — traditional domain of BIM.

Such requirements enabled increase in remodelling and mirroring of similar features in either domain. Nonetheless, it leads towards extra efforts of duplicating information for the overlapping features, as both BIM and GIS systems store and represent data in different formats (Wang et al. 2019). Therefore, these necessary conditions have accelerated the focus of bringing BIM and GIS domains together. During the last decade, many studies (Figure 2.2) have been conducted to achieve integration of these heterogeneous systems.

Divergent Features: Interoperability of BIM and GIS standard formats, IFC and CityGML, is the mapping between their key schemas, as they have different elementary development purposes, concepts and structures (Deng et al. 2016, El-Mekawy and Östman 2010, Gröger and Plümer 2012). The semantic mismatch among key schema elements of these two systems is a critical barrier towards interoperability. Correspondingly, the geometric representation types like Boundary Representation (B-Rep) are defined in IFC and CityGML with their own geometric configurations supported by respective tools, which further contributes to obstruction in geometrical information exchange (Zhu et al. 2020). Nevertheless, even with the data or domain-oriented differences elevating semantic and geometrical complexities, integration of BIM and GIS is considered as great potential to bring benefit in multiple application areas (Liu et al. 2017) like facility management and energy management applications to fulfill demands of decision making (details in Section 2.4). The section below highlights the perspectives of methodologies undergone the integration for BIM and GIS.

Integration with IoT: Consequently, BIM integrated with the Internet of Things (IoT) devices presents a powerful paradigm, but its integration still appears to be at nascent stages (Tang et al. 2019). Furthermore, including GIS with BIM and IoT provides the capability of its applications towards smart management (Park et al. 2018) including Augmented Reality (AR)/Virtual Reality (VR) technologies (Carneiro et al. 2019). However, even if BIM is integrated with IoT, question of integrating GIS information with BIM still remains open.

2.3 Trends of BIM-GIS Integration Methodologies

A series of studies presents a critical and state-of-the-art review on the BIM and GIS integration (Liu et al. 2017, Zhu et al. 2018) by complimenting their domain and data-oriented strengths and weaknesses of most relevant integration techniques (Breunig et al. 2020, Song et al. 2017). With potentialities of integration, data interoperability efforts focusing on BIM and GIS integration are made using prominent information exchange formats of two domains, IFC and CityGML (Sani and Rahman 2018). Furthermore, benchmarking studies have been conducted to investigate the BIM and GIS formats, and existing methodologies for their integration (Noardo et al. 2020a,b).

IFC and CityGML are widely accepted open-standard formats from BIM and GIS community and used in integration methods. Various efforts have been conducted to classify BIM and GIS integration, such as; semantic or geometric level, unidirectional or bidirectional conversions, and commercial or open-source software (Fosu et al. 2015). Irizarry et al. (2013) categorized the integration methods into two interrelated levels: fundamental level and application level, where the fundamental level focuses on data exchange standards and interoperability at the data level, while application-level focuses on developing new methods with full potential to exploit BIM and GIS system together. Kang and Hong (2015) presents BIM and GIS integration have undergone various perspectives and classified these approaches into five groups based on similar subject keywords: schema mapping (Deng et al. 2016, El-Mekawy and Östman 2010), integrated web services (Cruz et al. 2004, Karan and Irizarry 2015), ontological modelling (Hor et al. 2016,

Integration Methods	Effectiveness	Extensibility	Effort	Flexibility
New standards and models	case by case	case by case	case by case	case by case
Conversion, translation and extension (manual)	medium	high	high	medium
Conversion, translation and extension (semi-automatic)	medium	medium	medium	medium
Semantic web technologies	high	high	high	medium
Services-based methods	high	low	high	low
Application focused methods	case by case	low	low	low

Table 2.1: Amirebrahimi et al. (2016)'s three-level categorization of BIM-GIS integration solutions (Data , Process , Application levels) and 'EEEF' comparison criteria by Liu et al. (2017).

Karan and Irizarry 2014, Peachavanish et al. 2006), data transformations and schema extensions (El-Mekawy et al. 2012). Furthermore, a significant three-leveled framework classified by Amirebrahimi et al. (2016) is presented in Table 2.1 which categorizes the integration studies into application, process and data level.

Data Level Integration: At the data level, models and structures are modified or extended to meet requirements. A wide range of studies conducted at the data level achieves interoperability among BIM-GIS data formats with promising results. These include linking, translation/conversion, extension and meta-models (mediation) and can be further divided into geometric and semantic level integration.

Process Level Integration: Approaches following process-level integration involves data standards from BIM and GIS to be simultaneously adopted in workflow and collaboration, providing flexibility, still underlying with the challenge of data interoperability among systems. Semantic Web Technology has proven to be promising with its flexibility of integrating heterogeneous data formats among the modification and introduction of new models at semantic level integrations. Thus, semantic web with its natural characteristics shows more interest for research towards the integration of BIM and GIS domains.

Application Level Integration: These integration methods include reconfiguring or rebuilding the new application with integrated BIM-GIS or extending the existing application. This classification group involves extensions using plugins or built from scratch to support BIM functionalities in GIS systems or GIS features in BIM-supported software. This approach is generally costly and inflexible, and by far, no BIM software can directly read GIS data or vice versa.

2.4 Applications of BIM and GIS Integration

As mentioned in earlier sections, the prime advantage of BIM and GIS technologies is their ability to handle detailed spatial, semantic and geometric data for a multitude of analysis, visualizations, and use case-specific or general applications. BIM was initially used for the planning and design phases of a project and now in the construction and maintenance phases for a wide range of applications. Considering these perspectives, the feasible integrated systems of BIM and GIS can extend spatial analysis capability extended in BIM implementation and detailed semantic and geometric information to be visualized and analyzed in GIS environments. Enabling information exchange and interoperability at semantic provide seamless information exchange and geometric level integration process with extension solutions for specific use-case applications such as 3D visualization of flood damage to a building (Amirebrahimi et al. 2016). The applications of 3D city models provide numerous use cases like urban planning (Chen et al. 2020), asset management (Farghaly et al. 2019), location-based solutions (Wang and Issa 2020), site selection (Irizarry and Karan 2012a, Isikdag et al. 2008), heritage management (Saygi and Remondino 2013), emergency response, facility management(Biljecki et al. 2015), and sustainable environment analysis of smart cities (Jamei et al. 2017). The BIM-GIS applications explored in different stages of construction phases include design (Isikdag et al. 2008) and operation (Irizarry et al. 2013).

2.5 Role of Semantic Web Technology in BIM and GIS Integration

In order to achieve interoperability in diverse disciplines, Semantic Web and Linked Data have been investigated by researchers as complementary for technologies in existing AEC industry (Pauwels et al. 2017b). Interoperability to improve information exchange processes, identify and link related information as well as exploit reasoning on information obtained from these sources (Ozturk 2020), and a step forward towards building ontologies in AEC (Pauwels and Terkaj 2016) and geospatial (Wang and Issa 2020) using semantic web technologies.

Traditional methods for integrating BIM and GIS have highlighted information loss, incompatibility of available software and data formats, and limitations in use-case-specific frameworks. However, the integration methods conducted based on semantic web technology have shown a promising contribution for achieving the interoperability between BIM and GIS (Liu et al. 2017). Integration methods proposed by Hor et al. (2016), Karan et al. (2016) enable enhanced data exchange and integration between BIM and

GIS from syntactic to semantic level. The developed data exchange to ensure interoperability and accessibility for facility management data for building information (Kim et al. 2018), mapping techniques of ontologies development for BIM and GIS as reference ontologies Deng et al. (2016), El-Mekawy and Östman (2010) – pivotal for semantic integration – provides more feasible integration solutions with leading potential applications of intelligent urban mobility(Hor et al. 2018), highway alignment planning (Zhao et al. 2019), and urban facility technical management (Mignard and Nicolle 2014). The applications include, but are not limited to, investigating Ontology-based integration for indoor routing (Wang and Issa 2020) and geospatial analysis of preconstruction phases (Karan and Irizarry 2015). However, these potential solutions have limitations primarily towards fundamentals of semantic web technique – devising ontology (Karan and Irizarry 2015, Pauwels et al. 2017b, Zhu et al. 2018).

Recently, studies have been conducted to develop and standardize the ontology models of IFC (Pauwels and Terkaj 2016) and CityGML (Métral et al. 2013, Zalamea et al. 2013). The Pauwels and Terkaj (2016) provides extensive framework for EXPRESS based IFC to OWL and CityGML ontology generated by Métral et al. (2013) requires manual tunning. Therefore, there is room for improvement with XML-based common format to be adapted among BIM and GIS, i.e. ifcXML and CityGML, to provide solutions comparable within the same context. More details about semantic web role in ontology generation and ontology alignment (mapping) are provided in Chapter 3.

Although semantic integration of BIM and GIS has emerged as an important research area, expressiveness in both domains is different for 3D modelling structures and their semantics. To benefit from the built-in capabilities across file boundaries of BIM and GIS and integrate different vocabularies like IoT to harness reasoning and perform standardized queries, semantic web technologies have come into more focus of research efforts.

2.6 Summary

It can be evident from the extensive literature of BIM and GIS technologies and their integration approaches presented in previous sections of this Chapter that establishing interoperability among these technologies provides a promising future of their information exchange in smart cities applications. However, due to the data formats of cross-domain elements are geometrically and semantically inconsistent, providing seamless exchange still remains a question. Above that, incorporating IoT data becomes overhead for not a fully integrated set of information. This Chapter sketches previous and undergoing studies for the integration process of BIM and GIS, including IoT, their characteristics and limitation, and provides a brief synopsis of semantics web as key for integration for the broad goal of engineering Smart Data.

Similarly, another benchmark study, Noardo et al. (2020a) analyzes conversions techniques, available procedures and state-of-the-art tools for building and geospatial domains by utilizing open-standard exchange formats (IFC to CityGML and CityGML to IFC) to evaluate and highlight limitations of procedures, software systems, their incompatibility and discrepancy in data sets itself as well. The benchmark study provides supplementary grounds for the semantic integration-based solution. More details of this benchmarking activity are discussed in Chapter 4.

Much research is currently ongoing in this area of interest. However, the connection with the world of practice and the availability of mainstream technical solutions is limited. Liu et al. (2017) by EEEF selection criteria state semantic web as much more promising solution than other methodologies for the integration of BIM and GIS. Henceforth, with all literature and related work conclusions, the vision of semantic web-driven technology and tools is selected for this research study. The remaining thesis further elaborates and presents its framework designed for this study.

Chapter 3

XML Schema in Semantic Web for Ontology Generation and Alignment

In this chapter, semantic web technology details are described with their role to enable interoperability among information systems by building ontology and mapping their entities across heterogeneous information systems. It also discusses the purpose and challenges in accomplishing these approaches. The Chapter presents ontology generation and alignment association by bringing information from Extensible Markup Language (XML) documents into the semantic web and explicitly discusses the XML schema components of ifcXML and CityGML. This research exploits these data-oriented XML-based open-standards of BIM and GIS, respectively.

This Chapter is divided into sections as follows: Section 3.1 starts with the vision of ontology and its importance in representing information that is essential for integration of information systems; Section 3.2 delves in details about the semantic web domain and the importance of its technology stack for building mesh of data; Section 3.3 provides insights on XML Schema Documents (XSD), their components and design styles; Section 3.4 highlights the schema components of data formats used in this study; Section 3.5 provides in close association of semantic web with XML/XML schema documents in-depth details for their consorting on ontology generation as well as ontology alignment, Section 3.6 identifies and discusses approaches promising for automatic ontology generation; and lastly, Section 3.7 provides summary of this Chapter by capturing influence of semantic web on XML schema and highlighting the possible approaches for ontology generation and alignment, that can be adapted in this research study.

3.1 What is an Ontology?

An ontology defines a common vocabulary of a domain to have shared information with machine-interpretable definitions of domain concepts and relations between them (Noy and McGuinness 2001). The primary purpose of an ontology is to create formal models of knowledge representation with some logical constraints. There have been multiple attempts to define what an ontology is (Noy and McGuinness 2001), but the best-known definition (in computer science) is due to Gruber (Gruber 1992):

"An ontology is a *formal*, *explicit specification* of a *shared conceptualization*."

where, the *conceptualization* is an abstract model and simplified view of some real-world aspects *shared* as consensual knowledge, and defined properties and constraints for the concepts and their relationships is *explicit specification*. The *formal*, in this context, means that model should be specified in some unambiguous language, making it amenable to read and process by machines and humans.

An ontology constitutes Classes (or concepts), Individuals, Relations, Datatypes, Attributes, Restrictions, and Axioms. Ontologies define terms in annotations using a set of pre-defined concepts and transform them into semantic annotations. Hence, an ontology together with a set of individual instances of classes constitutes a knowledge base. It can formally be represented in a tuple O = (C, R, I, D, binary relation) (Bedini et al. 2010a) and underlines a strong foundation of Description Logic (Horrocks 2007) for formal description of concepts and their roles (relations). It provides inference with ontology to perform specific reasoning to provide maintenance, consistency and classification to knowledge bases. The ontology and domain knowledge's potential applications are undoubtedly reliant on its designer, influencing the ontology design choices. Henceforth, an ontology's quality can be better assessed using it in the application it is designed for.

3.2 Domain of Semantic Web

Semantic Web (Jiehan et al. 2006), or Web 3.0, is an extension design of the World Wide Web (WWW) to represent information in a well-defined common data format human-readable and understandable by computer systems. The semantic web, known as the mesh of data, enables creating data storages, building vocabularies, encoding the semantics from vocabularies with data, and defining rules for handling it. The semantic web technology stack is empowered by multiple layers of technologies such as OWL, RDF and SPARQL (illustrated in Figure 3.1). The technology stack plays a substantial role in achieving semantic interoperability and is published as Wide Web Consortium (W3C) recommendations to be exploited by its applications. The key technologies of OWL, RDF and SPARQL are briefly described in the following sections:

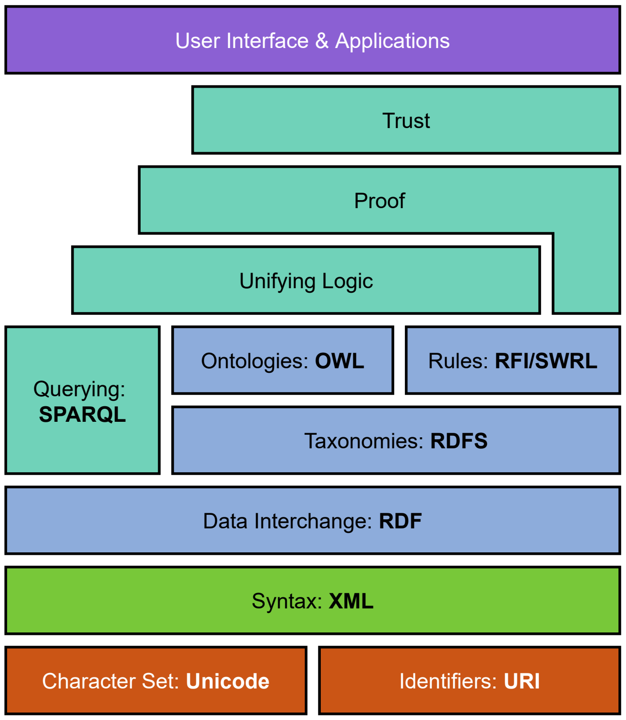


Figure 3.1: Semantic Web stack of technologies - layout inspired by Lu and Asghar (2020).

3.2.1 Resource Description Framework (RDF)

RDF (Cyganiak et al. 2014) is a graph format at the core of the semantic web that holds a flexible and generic language to enable the combination and representation of information from diverse knowledge domains. A W3C standard (Schreiber and Raimond 2014) describe data as collection of three-part statement (called triple) as subject-predicate-object (S-P-O) (presented in Figure 3.2) or as an (entity identified-attribute name-attribute value). Here the subject and predicates are URIs (Uniform Resource Identifiers) to identify them uniquely. Objects can either be literals or other URIs; hence, objects of some triples connect with subjects of other triples or literals to create a network of linked nodes called a graph. RDF works like a framework to manage and represent ontologies.

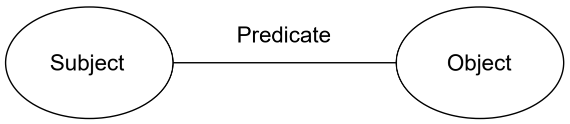


Figure 3.2: An example of RDF triple S-P-O format.

An RDF graph can be serialized using various syntax including RDF/XML (`.rdf`), N-Triples (`.nt`), Turtle (`.ttl`) and Notation-3 (`.n3`) (Schreiber and Raimond 2014). Turtle syntax format is simple to write and human-readable format in which RDF graphs can be expressed in URIs that are abbreviated as a prefix to all RDF statements (see Figure 6.2). Any data like relational databases, XML documents etc., can be expressed as a collection of triples gives RDF a natural ability to express data from any format in the semantic web. Also, it provides the ability to express all semantics and data to be captured in triple format – producing minimal loss of information. Suppose some properties do not exist in the given vocabulary. In that case, RDF has the flexibility to mix and match different vocabularies or create a new vocabulary with the given domain name and add associated properties to enable customization and standardization. It also allows multiple properties of the same type to be associated with a single subject. For example, an object A can be `rdfs:subClassOf` of multiple objects X and Y by associating `rdfs:range X` and `rdfs:range Y` with object A, which is not easy to achieve in relational databases.

3.2.2 Web Ontology Language (OWL)

OWL provides the key to expressiveness in the semantic web. Recently OWL, along with RDF on which it is based, has become popular standards for data representation and exchange. The OWL language's semantic expressiveness is specified in W3C, where the first OWL version of W3C Recommendation is superseded by the OWL2 language specification (Group 2012). The relevant references to the semantics specified to OWL in this document refer to the OWL2 specifications.

OWL Profiles: Similar to preceding OWL version, OWL2 also has number of so-called *profiles*; namely OWL 2-EL, OWL 2-QL and OWL 2-RL (Motik et al. 2012).

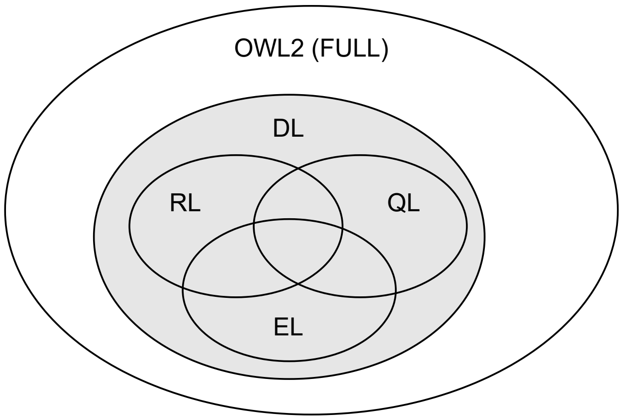


Figure 3.3: A Venn Diagram by Group (2009) of OWL2 profiles EL, QL and RL exhibiting syntactic sub-subset, each profile trades off different aspects of OWL's expressive power (more information, less performance).

Figure 3.3 provides an overview and relationship between these key profiles. Motik et al. (2012) indicates an OWL2 profile "*is a trimmed down version of OWL2 that trades some expressive power for the efficiency of reasoning*". In short, several statements that can be used in OWL 2-DL are not allowed in each of the given OWL2 profiles. By not allowing these statements and thus sacrificing some expressiveness, each profile achieves efficiency differently and is useful in different application scenarios. While more of the information can be found in (Motik et al. 2012) for the expressiveness of each profile, the following is a summary for referenced profiles:

- **OWL 2-EL:** This profile is particularly useful in applications employing ontologies with very large numbers of properties and/or classes. The expressive power captured by this profile is used by many such ontologies for which basic reasoning problems can be performed in time (`PTIME`) that is polynomial with respect to the size of the ontology. Although, OWL 2-EL places restrictions on types of classes and supports set of axioms like class expressions (`subClassOf`, `subPropertyOf`), important constructs such as universal restrictions (`allValuesFrom`), cardinality restrictions (`maxCardinality`, `minCardinality`, `exactCardinality`), disjunction (`unionOf`), enumerations (`oneOf`), and property related expressions (`disjoint`, `functionalProperty`, `symmetricProperty`), are not supported.

- **OWL 2-QL:** This profile is aimed at applications that exploits very large volumes of instance data, and where query answering performance is the most important reasoning task. Based on size of the data and execution technique, query answering can be performed in `LOGSPACE`. The overall expressive power of this profile is quite limited as it excludes constructs, among others, existential quantification (`someValuesFrom`), cardinality restrictions (`maxCardinality`, `minCardinality`), enumerations (`oneOf`) and property inclusions (`subPropertyOf`). Compared to OWL 2-EL, some property-related expressions are allowed (`inverseOf`, `disjoint`).

- **OWL 2-RL:** This profile is recommended to be used for applications that require scalable reasoning without sacrificing too much expressive power. Adopting ontologies in this profile is a good choice whenever reasoning is involved. The expressive power of OWL 2-RL is quite close to OWL 2-DL and is designed to accommodate OWL2 applications that can trade the full expressivity of the language for efficiency. It supports all axioms of OWL2 (except for disjoint unions of classes and reflexive object property axioms). However, few syntactic restrictions are required to be taken into account for an ontology to be in OWL 2-RL profile (Motik et al. 2012).

From the profiles mentioned above, OWL 2-RL is being used in this research study for the expressiveness of semantics of XML information. As mentioned earlier, less information provides more performance, which is feasible at this stage of the study.

3.2.3 SPARQL

SPARQL Protocol And RDF Query Language - Query Language for RDF is a W3C standard (Lopes et al. 2010) that helps retrieve data from RDF graphs. Similar to RDF Turtle syntax, SPARQL also supports defining stand-in abbreviated prefixes with URIs to shorten queries. The *WHERE* clause describes which triples are to be retrieved from querying datasets and provides substitution of wildcard variable for one, two or all three of triple query statement, also referred to as the triple pattern. Results of the SPARQL query are according to the triple pattern in *WHERE* clause based on the subject (URI), predicate(property) or object (URI or Literal). The *SELECT* clause indicates which variable values are to be enlisted in results. *A ** represents all of the queried variables to be displayed in the result. Results are returned based on triples statements from the collection based on the triple pattern in *WHERE* clause. In a single query, multiple triple patterns can be added for retrieving results with more information.

SPARQL has different keywords and multiple available query options, among which *CONSTRUCT* can be essential to create triples in turtle format as a result of SPARQL query for that is helpful in data integration. SPARQL provides data type and language tags along with sorting and aggregating results and enables data modification (add, update or delete functions).

3.2.4 Synthesis

The semantic web technology stack provides ontology as RDF/OWL for knowledge representation and is one of the most potent formalization languages to be defined based on Description Logic. There exist other languages for formalization, but OWL is the most feasible and widely adapted. Hence, for this research study, RDF/OWL is adapted to represent information. The aim is to exploit OWL syntax and express semantics in generated vocabulary as much as possible. The RDF graphs use these vocabularies in OWL representation and provide query-able information through SPARQL. The process leads to investigating the possible way of ontology development and information exchange by leverage least human intervention as possible.

3.3 XML Schema Documents, Components and Design Styles

The Extensible Markup Language (XML) (Bray et al. 2012), along with XML Schema Document (XSD) (Gao et al. 2012), is likely the description and specification formalism mainly designed as a general-purpose data storage and exchange format. XML documents are widely exploited to represent and manage (semi-structured) data in information systems, where XSD formally delegates XML with data modelling. XML provides a format that is both human-readable and machine-interpretable at the same time. The format also fits well in its simplicity and suppleness of usage with most application information exchange requirements. Furthermore, XML introduces Document Type Definition (DTD) and XSD formalism for a clean separation between meta-data and instances containing the actual data to be exchanged. However, XML remains, in certain senses, too open and let to the extent of dialects that tend to overload its primary usage and meanings.

The structure of the XSD document is composed of multiple XSD elements arranged in a way to precisely describe the XML language. As stated early, it checks the validity of the structure and vocabulary of an XML document against rules of appropriate XML language. An XSD structure can be majorly categorized into 6 groups: (i) Elements: ap-

Design Style	`element` and `attribute` Declaration	`type` Definition
Russian Doll	Local	Local
Salami Slice	Global	Local
Venetian Blind	Local	Global
Garden of Eden	Global	Global

Table 3.1: Classification of XML schema design styles listed by Brahmia et al. (2019).

pears in an XML document; (ii) Attributes: can be used for Elements in XML document; (iii) Simple and Complex Types: for defining element and/or attributes; (iv) Derived Types: extensions of Simple and Complex Types; (v) Grouped XSD Components: group of Elements or Attributes and (vi) Annotations: for the documentation and labelling.

The XSD syntax varies based on overall representation and changes involving schema designs resulting from different XML data structures. The XML language's flexibility derives from different arrangements of XSD components, which determines the syntax and complexity of an XSD design in their global or local scope for a given namespace forming specific XSD patterns. A global XSD component is an immediate sub-element of the root `<xs:schema>` element. It is also associated with `targetnamespace` of an XSD, making it accessible to other XML schemas using a given namespace. However, a local component is not defined as an immediate sub-element of `<xs:schema>`, instead of nested to an XSD element. Thus, not accessible outside the given schema definition.

Brahmia et al. (2019) recommends that any XML schema definition can be organized according to one of these five design styles (listed in Table 3.1): (i) *Russian Doll*, (ii) *Salami Slice*, (iii) *Venetian Blind*, (iv) *Garden of Eden* and (v) *Bologna*. The design styles are classified in the way for their definition and declaration, globally or locally, of XML schema components; `element`, `attribute`, `simpleType` and `complexType`. An example XSD is selected for better understanding of different design styles, stated below. Elements in below descriptions refers to Elements as well as Attributes:

- **Russian Doll:** An XML schema design with only one global `element` declaration that nests all other possible declarations of local components (that nest further local components), and local definitions of `simpleType` and `complexType`. An example of Russian Doll design is presented in Listing 3.1.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Thesis">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Title" type="xs:string"/>
        <xs:element name="Author" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Listing 3.1: Russian Doll style XSD.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Title" type="xs:string"/>
  <xs:element name="Author" type="xs:string"/>
  <xs:element name="Thesis">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Title"/>
        <xs:element ref="Author"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Listing 3.2: Salami Slice style XSD.

- **Salami Slice:** An XSD design corresponds to having all of Elements declarations in global namespace and then referencing the Element, while all `simpleType` and `complexType` are locally defined, see example listed in Listing 3.2.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:simpleType name="Title">
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="Author">
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="Thesis">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Title" type="Title"/>
        <xs:element name="Author" type="Author"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Listing 3.3: Venetian Blind style XSD.

- **Venetian Blind:** Similar to Russian Doll, the Venetian Blind design corresponds to having a single global `element` declaration that nests local XSD elements (that further nests local elements). However, local Elements uses types (`simpleType` and `complexType`), when needed, that are defined within global namespace. Listing 3.3 presents design of same example as Venetian Blind.

- **Garden of Eden** A normalized XML schema format, combination of Venetian Blind and Salami, that incorporates all possible Elements declarations and types definitions in the global namespace, with the Elements referenced as needed. Listing 3.4 presents Garden of Eden design of same example.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:simpleType name="Title">
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="Author">
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="Title" type="Title"/>
  <xs:element name="Author" type="Author"/>
  <xs:element name="Thesis">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Title"/>
        <xs:element ref="Author"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Listing 3.4: Garden of Eden style XSD.

- **Bologna:** A design pattern that is actually not a defined style and uses combination of global and local Elements, by default. This style conform otherwise to prior designs can be considered as Bologna design style as shown in Listing 3.5.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Title" type="xs:string"/>
  <xs:element name="Thesis">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Title"/>
        <xs:element name="Author" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Listing 3.5: Bologna style XSD.

Table 3.1 compares main XSD design styles proposed by XML schema community (Darr et al. 2011, McBeath and Hinkelman 2004) are adapted accordingly to the reusability of XSD components that are made available at different levels. StyleVolution (Brahmia et al. 2019) proposed a suite of procedures, featuring conversions from an XSD design style to another, that can facilitate the reuse and exchange of schema specifications encoded using the XML schema language. However, style conversion operations are complicated, error-prone, and impact XML schema specifications if not carefully performed.

3.4 XSD Specifications of ifcXML and CityGML

In this section, the XSD components of ifcXML and CityGML formats are inspected to understand their structure, complexity and design style. Eventually, these factors affect the process that considered XSD documents for formalization and mapping to other formats like OWL, which will be discussed later in Section 3.6. To comprise rich semantic and geometric BIM and GIS information, XML-based data formats like ifcXML and CityGML models are devised with complex XSD structures, respectively. Based on the previous section's design styles, proceeding XSD designs for their schema structures are scrutinized as a hybrid in design styles. Listing 3.6 shows a snippet XSD schema from IFC 4X1 format that displays elements' composition bisects designs of Russian Doll, Garden of Eden, Venetian Blind and Salami Slice. Similarly, a snippet of cityGMLBase XSD schema in Listing 3.7 shows schema style characterized more as Bologna design style, which is not a recommended XSD design style (Brahmia et al. 2019).

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:ifc="http://www.
    buildingsmart-tech.org/ifc/IFC4x1/final" .. >
  ...
  <xs:element name="IfcWindow" type="ifc:IfcWindow"
     substitutionGroup="ifc:IfcBuildingElement" nillable="true"/>
  <xs:complexType name="IfcWindow">
    <xs:complexContent>
      <xs:extension base="ifc:IfcBuildingElement">
        <xs:attribute name="OverallHeight" type="ifc:IfcPositiveLengthMeasure" use="
            optional"/>
        <xs:attribute name="OverallWidth" type="ifc:IfcPositiveLengthMeasure" use="
            optional"/>
        <xs:attribute name="PredefinedType" type="ifc:IfcWindowTypeEnum" use="optional"/>
        <xs:attribute name="PartitioningType" type="ifc:IfcWindowTypePartitioningEnum"
            use="optional"/>
        <xs:attribute name="UserDefinedPartitioningType" type="ifc:IfcLabel" use="
            optional"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  ...
  <xs:group name="IfcUnit">
    <xs:choice>
      <xs:element ref="ifc:IfcDerivedUnit"/>
      <xs:element ref="ifc:IfcMonetaryUnit"/>
      <xs:element ref="ifc:IfcNamedUnit"/>
    </xs:choice>
  </xs:group>
</xs:schema>
```

Listing 3.6: Sample snippet from IFC 4x1 XSD design.

Table 3.2 provides the schema specifications of different XSD components for selected IFC and CityGML formats, along with an example set that is considered further in this research and schema of an IFC dataset from GeoBIM benchmark (Section 4.2.1). The Tables 3.2 provides the usage of XSD components for a better understanding of information that can be represented in other exchange formats. Since the IFC is a comprehensive yet complex data format, and XML stores information in large format, the schemas are considerably large in size and components (as shown in Table 3.2).

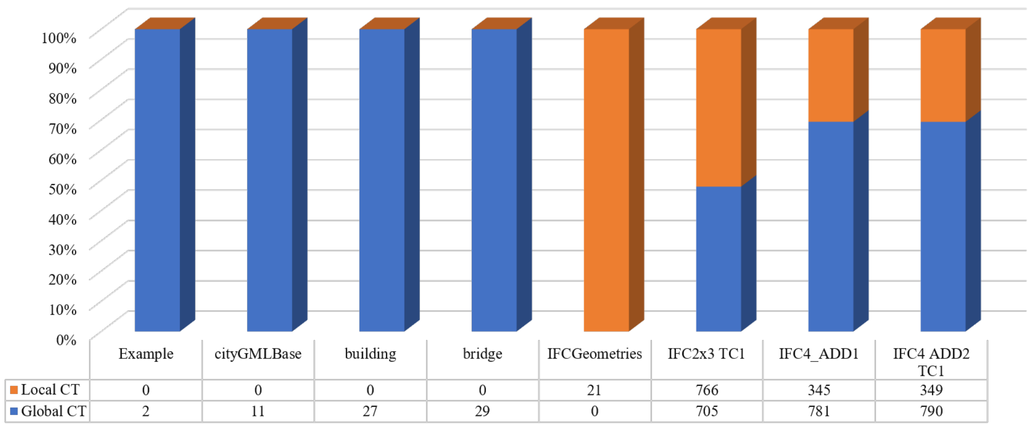
```
<xs:schema xmlns="http://www.opengis.net/citygml/2.0" xmlns:xs="http://www.w3.org/2001/
    XMLSchema" .. >
   ...
  <xs:complexType name="AbstractCityObjectType" abstract="true">
    <xs:complexContent>
      <xs:extension base="gml:AbstractFeatureType">
        <xs:sequence>
          <xs:element name="creationDate" type="xs:date" minOccurs="0"/>
          <xs:element name="terminationDate" type="xs:date" minOccurs="0"/>
          <xs:element name="externalReference" type="ExternalReferenceType" minOccurs="0"
               maxOccurs="unbounded"/>
          <xs:element name="generalizesTo" type="GeneralizationRelationType" minOccurs="0
              " maxOccurs="unbounded"/>
          <xs:element name="relativeToTerrain" type="RelativeToTerrainType" minOccurs="0"
              />
          <xs:element name="relativeToWater" type="RelativeToWaterType" minOccurs="0"/>
          <xs:element ref="_GenericApplicationPropertyOfCityObject" minOccurs="0"
              maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  ...
  <xs:complexType name="AddressType">
    <xs:complexContent>
      <xs:extension base="gml:AbstractFeatureType">
        <xs:sequence>
          <xs:element name="xalAddress" type="xalAddressPropertyType"/>
          <xs:element name="multiPoint" type="gml:MultiPointPropertyType" minOccurs="0"/>
          <xs:element ref="_GenericApplicationPropertyOfAddress" minOccurs="0" maxOccurs=
              "unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:element name="_GenericApplicationPropertyOfAddress" type="xs:anyType" abstract="
      true"/>
  ...
  <xs:complexType name="xalAddressPropertyType">
    <xs:annotation>
      <xs:documentation>Denotes the relation of an Address feature to the xAL address
          element.</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element ref="xAL:AddressDetails"/>
    </xs:sequence>
  </xs:complexType>
  ...
</xs:schema>
```

Listing 3.7: Sample design snippet from cityGMLBase XSD.

XSD Element	Example	cityGMLBase	building	bridge	IFCGeometries	IFC2x3 TC1	IFC4_ADD1	IFC4 ADD2 TC1
annotation	-	19	18	20	-	-	-	-
attribute	1	-	-	-	20	402	1467	1479
attributeGroup	-	3	7	8	-	281	116	117
choice	-	1	-	-	-	298	61	61
complexContent	1	5	20	21	-	659	774	782
complexType	2	11	27	29	21	1471	1126	1139
documentation	-	19	18	20	-	-	-	-
element	3	35	139	153	53	2936	2044	2067
enumeration	2	11	-	-	-	1310	1638	1643
extension	1	5	20	21	-	935	888	898
group	-	-	-	-	-	222	181	181
import	-	2	2	2	-	1	-	-
length	-	3	-	-	-	-	-	
list	-	1	-	-	-	-	52	55
maxInclusive	-	2	-	-	-	-	-	-
maxLength	-	-	-	-	-	1	15	15
minInclusive	-	2	-	-	-	-	-	-
minLength	-	-	-	-	-	1	32	33
restriction	1	7	-	-	-	282	386	389
schema	1	1	1	1	1	1	1	1
sequence	1	10	27	29	11	607	399	408
simpleContent	-	-	-	-	-	279	118	120
simpleType	1	8	-	-	-	279	434	440

Table 3.2: XML schema specifications for different component of multiple sets of XML schemas.

Furthermore, the extracted information of these XSD components is filtered for only `complexType` tag and presented in Figure 3.4 to specify more the global `complexType`, easier it is to map as the local declaration is anonymous and requires custom naming. Similarly, Figure 3.5 details for the `element` tag of the selected XML schemas. More `element` declarations with `type` or `ref` results in better mappings considerations which are considered in CityGML schemas, unlike IFC formats. Since the XSD design patterns



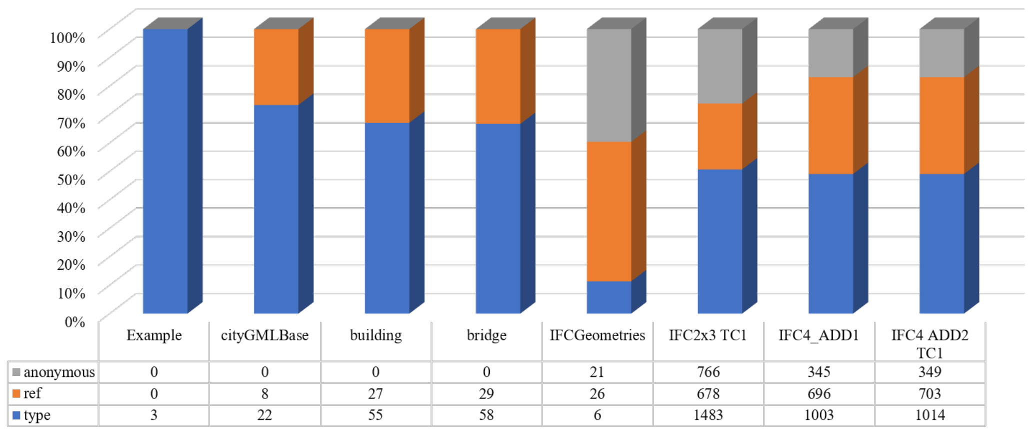
	Example	cityGMLBase	building	bridge	IFCGeometries	IFC2x3 TC1	IFC4_ADD1	IFC4 ADD2 TC1
🟧 Local CT	0	0	0	0	21	766	345	349
🟦 Global CT	2	11	27	29	0	705	781	790

Figure 3.4: Percentage distribution of the declaration of Global and Local `complexType` tags among input XML schemas.

of presented XSD schemas are defined in numerous ways, the solutions to implement a generalized XML schema pattern recognition algorithm and consequently mapping it with OWL representation becomes difficult. It may lead processes towards laborious efforts requiring manual validation.

3.5 Using XML Schema for Ontology Generation and Alignment

The previous sections provided details on the semantic web and its technology stack's potential, followed by details on XML documents validated by their XSD structures. Due to their close nature of XML environments, the XML and semantic web have been investigated for interoperability. These technologies complement one another in terms of interdisciplinary information exchange that is extensible and flexibility (Bikakis et al. 2013). Many efforts have been made for both perspectives of ontology development and mapping (alignment) of the built ontologies. The sections below provide details on each



	Example	cityGMLBase	building	bridge	IFCGeometries	IFC2x3 TC1	IFC4_ADD1	IFC4 ADD2 TC1
⬜ anonymous	0	0	0	0	21	766	345	349
🟧 ref	0	8	27	29	26	678	696	703
🟦 type	3	22	55	58	6	1483	1003	1014

Figure 3.5: Percentage distribution of `element` tag across XML schemas either with attributes `type` or `ref`, or anonymous declaration.

approach adapted for XML/XML schema:

3.5.1 Transforming XSD to OWL Models

The art of ontology development is complicated, mostly with manual approaches (Liu et al. 2017). Expressing correct semantics of data in an ontological representation itself requires domain knowledge. The literature highlights semi (Bedini et al. 2010a, Cruz and Nicolle 2008, Yahia et al. 2012) or fully-automated processes (Minutolo et al. 2014) for ontology generation. However, the frameworks are manifested mainly for B2B (Business to Business) or text-based approaches and include minimal extension towards geospatial domain ontologies (Bedini and Nguyen 2007, Hacherouf et al. 2015). Also, ontology enrichment purely depends on defined rules for semantic representation of XSD components and their relations for respective data.

As ontology generation is the key element of this document, Table 3.3 outlines what

Approach	Transformation	Consistency	Formal	XSD Elements Count
OWLMAP (Ferdinand et al. 2004)	Automatic	No	No	14
XML2OWL (Bohring and Auer 2005)	Automatic	No	No	8
XS2OWL (Tsinaraki and Christodoulakis 2007)	Automatic	No	No	7
XSD2OWL (Cruz and Nicolle 2008)	Semi-Automatic	No	No	-
X2OWL (Ghawi and Cullot 2009)	Automatic	No	No	9
JANUS (Bedini et al. 2011)	Automatic	No	No	19
EXCO (Lacoste et al. 2011)	Automatic	No	No	10
Nora Y. Approach (Yahia et al. 2012)	Automatic	No	No	9
XSD2OWL2 (EL Hajjamy et al. 2017)	Automatic	Yes	Yes	17
PIXCO (Hacherouf et al. 2019)	Automatic	Yes	Yes	24

Table 3.3: Comparison study of XML/XSD to OWL transformation approaches by Hacherouf et al. (2015, 2019) sorted by year of their publications.

options are available in terms of building an ontology. Listed studies in Table 3.3 have been conducted to define correspondences between XSD and OWL which converts given XML files to OWL format using transformation tools. As an example, Table 3.4 shows the details of mapping rules defined between two formalisms. It demonstrates that `owl:DatatypeProperty` relates to individuals with `simpleType` or literal data (e.g. strings, numbers, data time etc.). In contrast, `owl:ObjectProperty` relates to individuals with more complex relations (e.g. defined objects, external objects). While these available approaches provide promising solutions, they lack ready tools for implementing and handling complex XSD schemas with different design styles.

Apart from ontology development, mapping ontologies still requires improvements as semantic web is still not mature. Therefore for BIM and GIS domain, automatic ontology generation and mapping methods are being investigated across the studies (An and Park 2018). Although approaches are developed for improving individual ontologies for CityGML ontology (Métral et al. 2010, Zalamea et al. 2013), however, they require manual tuning (Métral et al. 2013). Only one ontology standard exists for BIM ontologies, namely ifcOWL (Pauwels and Terkaj 2016) that implements conversion patterns to convert EXPRESS into OWL. Therefore, before ontology mapping, a comprehensive methodology for ontology generation needs to be devised.

XSD	**OWL**
xs:complexType/group/attributeGroup	owl:Class
xs:simpleType/attribute	owl:DataTypeProperty
xs:element (of complex Type)	owl:ObjectProperty, rdfs:subClassOf
Element Type (or attribute) local	owl:allValuesFrom
xs:sequence/all	owl:intersectionOf
xs:choice	Boolean Expression owl:intersectionOf, owl:unionOf and owl:complementOf
minOccurs/maxOccurs	owl:minCardinality/maxCardinality/ cardinality
substituitionGroup	owl:subClassOf

Table 3.4: XSD to OWL mapping rules defined in OWLMAP (Ferdinand et al. 2004)

3.5.2 Mapping of Ontology Models

The ontology development is a step towards creating an extensive vocabulary to be mapped with other existing or newly developed ontologies. Ontology alignment, also called ontology mapping, is the key to reaching interoperability over cross-domain ontologies (Raad and Evermann 2015). It has been investigated for several years with specialized studies to formally integrate ontologies or knowledge-bases formed in different domains (Farah et al. 2016, Giunchiglia et al. 2012). Heterogeneous domain data like BIM and GIS requires linking entities (concepts) before cross-domain information is available for processing. The integration of these distributed ontologies establishes cross-domain integrated ontology for information analysis and knowledge-graph applications. However, these approaches are generally limited to corpus-based studies, which further requires investigating alignment knowledge for entities and information specific to other domains like geospatial. Some of these approaches adapted for geospatial ontology alignment are either manual or lacks in mapping across entities (Deng et al. 2016, El-Mekawy and Östman 2010, Hbeich and Roxin 2020).

The literature highlights the significance of integrating semantic data from heterogenous sources (Keeney et al. 2011), majorly investigating interoperability among XML documents using semantic web (Bikakis et al. 2013). The semantic enrichment process for urban analysis involves data heterogeneity towards multi-jurisdiction analysis and comparison (Chen et al. 2020). For this study, we investigate an innovative approach towards mapping geospatial data, which mainly applies semantic-based *Word2vec* algorithm (Mikolov et al. 2013) and structure-based *Node2vec* algorithm (Grover and Leskovec 2016) for ontology alignment of BIM and GIS ontologies. Further details of these approaches are presented in Section 5.1.2.

3.6 Potential Solution for Ontology Generation

In sections earlier, the Chapter presented detailed literature on ontology generation and alignment techniques. However, before semantic alignment, a fully automated procedure for generating ontology from XML schema remains in question. In this section, Janus (Bedini et al. 2010b) and PIXCO (Hacherouf et al. 2019) methods of XSD to OWL methodologies are presented from literature with the potential of solving the automatic ontology generation problem. These methods are briefly presented in the below sections; then, revised solutions of the considerable transformation patterns proposed by JANUS and improvements on three-steps of PIXCO are identified – that are critical for different XSD designs and overlooked during the transformation process.

Janus

Bedini et al. (2011) proposed a process to generate OWL2-RL ontology by transforming XML schema using patterns and a developed prototype tool called Janus for its validation. A transformation pattern is a corresponding representation of an XSD sub-structure to an equivalent OWL ontology model. The method presents 40 transformation patterns while considering 19 XSD elements compared to similar approaches presented in Table 3.3. Janus provides Table II-VII with six groups presenting 40 patterns. The first columns present a particular XSD schema structure, and the corresponding OWL model representation is shown in the second column. The correspondence rules are promising and are adapted in the PIXCO study presented below.

#	XSD Constructs	OWL construct (Turtle syntax)
1	`⟨simpleType name="st_name"⟩`	`ag:st_name rdf:type rdfs:Datatype .`
3	`⟨complexType name="ct_name"⟩`	`ag:ct_name rdf:type owl:Class .`
27	`⟨complexType name="ct_name"⟩` `⟨attribute name="attr_name"` `type="st_name"/⟩`	`ag:has_attr_name rdf:type` `owl:DatatypeProperty ;` `rdfs:domain ag:ct_name ;` `rdfs:range ag:st_name .`
34	`⟨attributeGroup` `name="attr_grp_name"⟩` `⟨attribute name="attr_name"` `type="attr_type"/⟩`	`ag:has_attr_name rdf:type` `owl:ObjectProperty ;` `rdfs:domain ag:attr_grp_name ;` `rdfs:range ag:attr_type .`

Table 3.5: A subset of mappings presented correspondence rules of XSD to OWL transformation patterns from Janus Bedini et al. (2011).

PIXCO

The PIXCO methodology formally implements Janus patterns and extends patterns for XSD components like `key, unique, keyref` that are not considered in Janus. The method increases the count to 43 for transformation patterns while processing 24 XSD elements. PIXCO framework also proposes a formal method of XSD to OWL transformation by representing patterns using Formal Concept Analysis (FCA) context and a mathematical model $\mathcal{FS}(\mathcal{XS})$ to manipulate XSD constructs. These formal models are the basis for algorithms outlined in PIXCO methodology for pattern identification among XSD constructs and generating a corresponding OWL model of the identified pattern. Table 3.5 presents the concept of how a transformation pattern represents XSD construction to an OWL construction.

Janus and PIXCO approach majorly focuses on corpus-based and Business to Business (B2B) domain information transformation and exercise more towards representing key concepts and relations in ontology development from XML schemas. However, it becomes challenging to transform complex XML schema structures where XSD components are tightly associated with their defined, derived, or anonymous types having discrete relations such as in ifcXML and CityGML schemas (presented in Section 3.4).

XSD components can be mapped to different OWL expressions based on `type` of components for certain transformation pattern. For example, the Table 3.5 shows the `owl:DatatypeProperty` is corresponding to `attribute` construction for pattern #27 which incorporates `type` attribute value as XSD native datatype (`xs:int, xs:string` etc.) or refer to `simpleType`. Similarly, `attribute` construct in #34 is represents as `owl:ObjectProperty` that is formulated for `complexType`, leaving incorrect representation. Therefore, not all of the patterns provided in Janus handles similar edge cases and as the XSD structure gets extensive the PIXCO algorithm omits its compliance with transformation. Since, Janus considers maximum possible XML schema constructions, therefore, these approaches are adapted in a manner to be implemented and extended to provide richness in pattern identification for maximum and improved transformation.

3.7 Summary

The semantic web's ultimate goal is to allow data to be processed automatically by tools and shared effectively by wider communities. Therefore, the semantic web has a natural ability to integrate information from different sources as it aims to provide machine-accessible semantics to annotations using rich ontologies. The semantic web integration at the process level does not change the data format and structure from both domains, limiting the information loss.

Therefore, the semantic web technology stack is adapted for this research study, and this Chapter presents a descriptive overview of its technology stack. The Chapter also presented a brief overview of XML and XML schema documents to understand their components and design styles. They are widely used for achieving semantic interoperability by first transforming XML to OWL and further mapping information from generated ontology models. The Chapter also presented XSD specifications of the selected set of XML schema for better understanding as they are evaluated further in Chapter 6 for the evaluation process. Finally, the Chapter concludes with selecting the most appropriate and comprehensive frameworks, Janus and PIXCO, for the ontology development process from literature to extend and further implement the enhanced framework.

Translating XML schema models to RDF/OWL ontologies through an automated process offers a significant advantage. It can reduce the human work necessary when designing an ontology and the effort required to transform the heterogeneous data-formats like IFC and CityGML into a common-format of RDF in the semantic web.

Though, as literature in Chapter 2 shows that the integration of BIM-GIS is a challenging topic. However, obtaining interoperability between BIM and GIS is promising using semantic web technology (Herle et al. 2020). The biggest challenges of these methods are the significant efforts required at the early stage and the isolated development of ontologies as semantic web domains are still underdeveloped and lack maturity. Defining semantic models are mainly manual and time-consuming development processes. It does define ontology that can be available for future use.

Furthermore, as both systems have different contents and data structures, developing a seamless integration system as part of the semantic web, common data format becomes challenging to be mapped. For such, a reference ontology or cross-domain integrated ontology must be designed to take a global view of both domains by extending and specializing both BIM and GIS domain and other high-level ontologies (upper ontologies). Although a comprehensive research framework is presented in Chapter 6, the reason for such complex nature of research, the scope of this thesis is constrained to only the ontology generation process, and remaining features are presented for future work.

Chapter 4

GeoBIM Benchmark Project: Investigating Interoperability between BIM and GIS

4.1 Overview

In Chapter 2, fundamental problems of information loss and geometric variance associated with the integration of BIM and GIS are substantiated. It underlined limitations of adapted and undergoing integration solutions. Moreover, it showed limited correspondence of the available mainstream technical solutions with the world of practice. To address this linking challenge, several leading projects have been instigated to investigate and fill the interoperability gap between GIS and BIM, such as OGC Future City Pilot[2],

[2]https://www.ogc.org/projects/initiatives/fcp1/

GeoBIM at TUDelft[3], and more recently, the GeoBIM Benchmark[4]. The benchmarking initiative involves recruiting participants from a variety of backgrounds, with different expertise, skills and interests (e.g. BIM, GIS and more), to test the available software tools and procedures with adequate datasets. A project aims to provide insights into the current state-of-the-art against the implementation of open-standards in the 3D Geo and BIM domains and identify compatibility issues and points for improvement. The benchmark investigates four topics:

- **Task 1:** What is the *support for IFC* within BIM (and other) software?
- **Task 2:** What options for *geo-referencing BIM* data are available?
- **Task 3:** What is the *support for CityGML* within GIS (and other) tools?
- **Task 4:** What options for *conversion (IFC* $\longleftrightarrow$ *CityGML)* are available?

This chapter features participation in the GeoBIM Benchmark project (Noardo et al. 2020a) to utilize existing software tools to assess the given 3D GIS and BIM open-standards, IFC and CityGML, respectively. Moreover, their integration and formulate recommendations in-order for further developing the standards and the tools that implement them. Although each task encompasses the analysis of standard functionalities like reading, visualize, import, manage and analyze. This study mainly focuses on the contribution in topic of **Task 4** – "What options for *conversion (software and procedural) (both*

[3]https://3d.bk.tudelft.nl/projects/geobim/

[4]https://3d.bk.tudelft.nl/projects/geobim-benchmark/

IFC to CityGML and CityGML to IFC) are available?" discussing additional features of export and convert. This section provides an overview of this chapter. The rest of the chapter is outlined as follows: Section 4.2 focuses on benchmark materials, including IFC and CityGML datasets and results template provided for a given task; Section 4.3 delves with the series of experiments and results carried out using proprietary software tools and benchmark datasets and lastly, Section 4.4 concludes this chapter.

4.2 Benchmark Materials

The GeoBIM Benchmark project permits adequate datasets and results templates as online submission forms (Noardo et al. 2019b), to direct benchmarking activity without biases and ensure consistency in benchmark results, standardizing the responses from participants as far as possible. Thus, any anomalies in submitted results can be linked back to specific software, procedure or tool handling the particular data—the subsections below further elaborate on the materials distributed for the activity.

4.2.1 Provided Data for Benchmark

The project recognizes adequate provisioning data for consistent results across tasks as one of the most critical challenges. Potentially, the datasets are identified and prepared (pre-processed) for this activity to serve the benchmarking purpose effectively. Several IFC and CityGML datasets (Noardo et al. 2019a) from real-world practice and within the academic environment were provisioned for this activity as a critical component to

IFC Data		
Name	**File Size**	**Comments**
Myran	27.78 MB	Small georeferenced architectural building model in Sweden to test main software functionalities.
Savigliano	22.07 MB	Building model in Italy to test IFC4 support by procedures and tools.
UpTown	246.82 MB	Large complex building data in Rotterdam for hardware and software related performance and support.
IFCGeometries	31 KB	Set of modelled geometries in IFC2x3 to test interpretation and support of specific geometric types.
	27 KB	IFC4 version of similar modelled geometries to test the geometrical support behavior.
CityGML Data		
Buildings	1.36 MB	Procedurally modelled buildings in LoD3 to test tool and related classes support.
Rotterdam	34.72 MB	Texturized district model in Rotterdam with LoD1 and LoD2 to test multi-LoD software support.
Amsterdam	5.02 GB	Seamless 3D city model of Amsterdam with CityGML entities (buildings, roads, water etc.) for hardware and software tests related performance and support of city entities.

Table 4.1: Noardo et al. (2019a) presented list of datasets for GeoBIM benchmark activity.

consider all tested software, tools and procedures comparable on an equal basis. While most of datasets[5] are free to download, IFC files were accessible to participants only. Table 4.1 enlists included datasets, and sections below present the summary for the details of these models:

IFC Data

- **Myran:** A georeferenced IFC2x3 model of a 2-floor office building in Falun, Sweden representing the architectural model of BIM (Figure 4.1a). The semantics in data are employed accurately, and many attributes are filled; however, as shown in Figure 4.1b, the grouping of entities across the storeys is not consistent.

[5]https://3d.bk.tudelft.nl/projects/geobim-benchmark/data.html

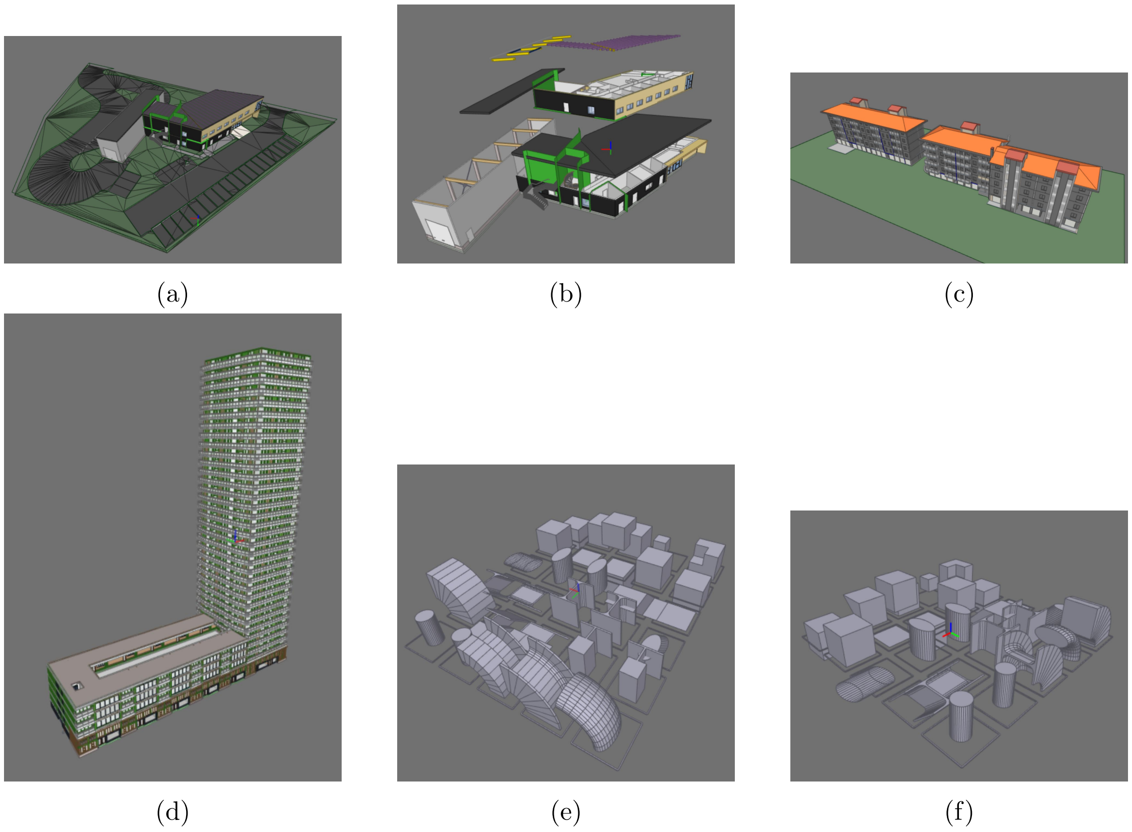


Figure 4.1: IFC data provided for benchmarking (Noardo et al. 2019a) to test software tools and procedures on equal basis: **(a)** georeferenced Myran.ifc model **(b)** entities inconsistency among stories in Myran.ifc **(c)** Savigliano.ifc model **(d)** achitectural model of UpTown building, **(e, f)** show IFCGeometries IFC2x3 and IFC4, respectively.

- **Savigliano:** It is the IFC2x3 model of a designed residential building with a series of low–rise tower blocks in Savigliano, Italy (Figure 4.1c). The data represents an architectural model of BIM and does not include georeferencing data.
- **UpTown:** An IFC4 model of a large residential tower in Rotterdam, Netherlands representing the architectural BIM model (Figure 4.1d), to test significant dimensions associated with software and hardware performances. Instead of an explicit

definition of IFC entities for distinct elements' information, the model has many generic IFC entities (*IfcBuildingElementProxy*) to cover the semantics of objects.

- **IFCGeometries:** A specific set of geometries modelled using a range of modelling alternatives allowed in IFC, which are usually less supported or incorrectly interpreted by software. Two versions of geometries' models are provided IFC2x3 and IFC4 (Figure 4.1e and Figure 4.1f), specifically selected to test geometric definitions and assess the diverging compliance within software tools supporting BIM.

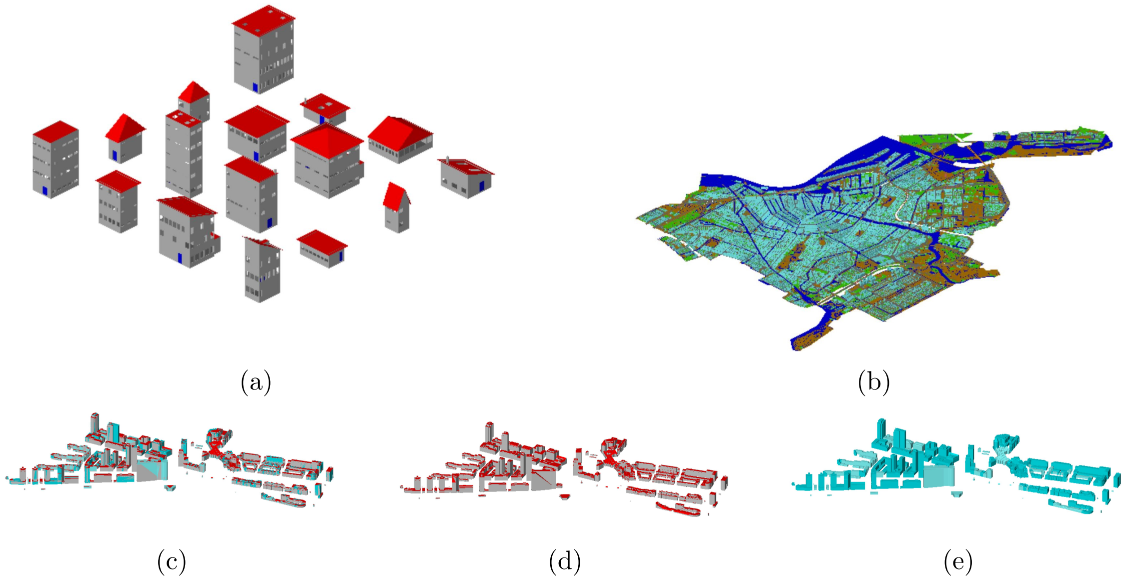


Figure 4.2: CityGML data provided for benchmarking (Noardo et al. 2019a) **(a)** Buildings LoD3 model **(b)** Amsterdam City LoD1 model **(c)** Rotterdam LoD1 & LoD2 Model **(d)** Separate Rotterdam LoD2 model (e) Rotterdam LoD1 model.

CityGML Data

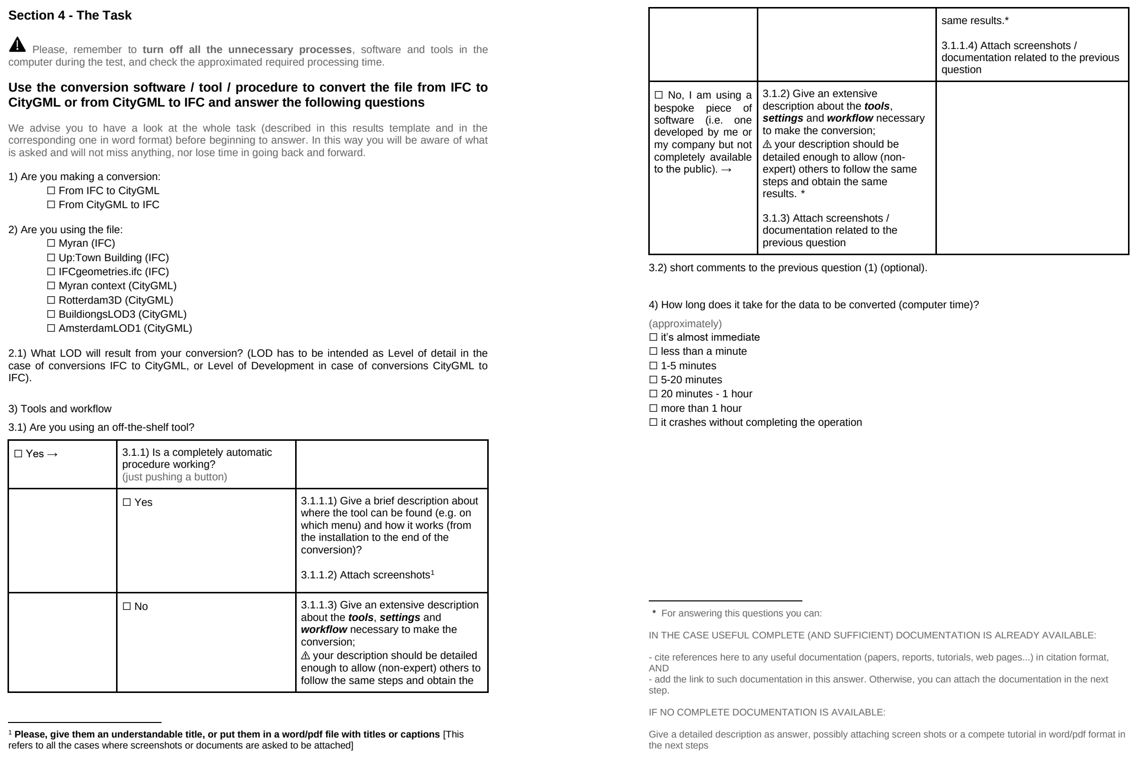
- **Buildings in LoD 3:** A procedurally generated CityGML model by the open-source tool Random3DCity (Biljecki et al. 2016a) with more details on the facades (Figure 4.2a). The generated model encompasses georeferenced data and is free of geometric errors.

- **Rotterdam LoD 2 and LoD 1:** A multi-Level of Detail (LoD) CityGML model produced and provided by the City of Rotterdam (Figure 4.2c) representing the surrounding area of the UpTown building (one of provided IFC models). It is selected to test software handling for extruded building footprints alongside more detailed structures. The dataset also includes some errors, kept to test software behaviours against these errors.

- **Amsterdam LoD 1:** It is a 3D model of the entire city of Amsterdam with all CityGML entities (buildings, roads, vegetation, etc.) in LoD 1, see Figure 4.2b. The open-data included a reference system and was selected as a part of the benchmark to test software tools dealing with large models, such as one entire (medium-size) city.

4.2.2 Results Template for Benchmark Tasks

To facilitate the comparison of obtained results from a wide participating community and range of software packages, tools and procedures, the results template was designed and provided for each benchmark topic (mentioned in Section 4.1) as online forms. Templates detailing the instructions to guide through tests and collecting the answers and results data systematically were divided into five sections: section 1 is for participants information; section 2 asks details for tested software or tool; section 3 is for the computer hardware to compare performances; section 4 is for the detailed description of the performed conversion task and section 5 finalizing the result submission with further information or comments. While sections 1-3, 5 are common in all benchmark topics, section 4 is unique for each task. Since this study and subsequent chapter focuses only on **Task 4**[6], a template of its section 4 is presented in Figure 4.3.



Section 4 - The Task

⚠ Please, remember to **turn off all the unnecessary processes**, software and tools in the computer during the test, and check the approximated required processing time.

Use the conversion software / tool / procedure to convert the file from IFC to CityGML or from CityGML to IFC and answer the following questions

We advise you to have a look at the whole task (described in this results template and in the corresponding one in word format) before beginning to answer. In this way you will be aware of what is asked and will not miss anything, nor lose time in going back and forward.

1) Are you making a conversion:
- ☐ From IFC to CityGML
- ☐ From CityGML to IFC

2) Are you using the file:
- ☐ Myran (IFC)
- ☐ Up:Town Building (IFC)
- ☐ IFCgeometries.ifc (IFC)
- ☐ Myran context (CityGML)
- ☐ Rotterdam3D (CityGML)
- ☐ BuildiongsLOD3 (CityGML)
- ☐ AmsterdamLOD1 (CityGML)

2.1) What LOD will result from your conversion? (LOD has to be intended as Level of detail in the case of conversions IFC to CityGML, or Level of Development in case of conversions CityGML to IFC).

3) Tools and workflow

3.1) Are you using an off-the-shelf tool?

☐ Yes →	3.1.1) Is a completely automatic procedure working? (just pushing a button)	
	☐ Yes	3.1.1.1) Give a brief description about where the tool can be found (e.g. on which menu) and how it works (from the installation to the end of the conversion)? 3.1.1.2) Attach screenshots[1]
	☐ No	3.1.1.3) Give an extensive description about the ***tools***, ***settings*** and ***workflow*** necessary to make the conversion; ⚠ your description should be detailed enough to allow (non-expert) others to follow the same steps and obtain the same results.* 3.1.1.4) Attach screenshots / documentation related to the previous question
☐ No, I am using a bespoke piece of software (i.e. one developed by me or my company but not completely available to the public). →	3.1.2) Give an extensive description about the ***tools***, ***settings*** and ***workflow*** necessary to make the conversion; ⚠ your description should be detailed enough to allow (non-expert) others to follow the same steps and obtain the same results. * 3.1.3) Attach screenshots / documentation related to the previous question	

[1] **Please, give them an understandable title, or put them in a word/pdf file with titles or captions** [This refers to all the cases where screenshots or documents are asked to be attached]

3.2) short comments to the previous question (1) (optional).

4) How long does it take for the data to be converted (computer time)?

(approximately)
- ☐ it's almost immediate
- ☐ less than a minute
- ☐ 1-5 minutes
- ☐ 5-20 minutes
- ☐ 20 minutes - 1 hour
- ☐ more than 1 hour
- ☐ it crashes without completing the operation

* For answering this questions you can:

IN THE CASE USEFUL COMPLETE (AND SUFFICIENT) DOCUMENTATION IS ALREADY AVAILABLE:

- cite references here to any useful documentation (papers, reports, tutorials, web pages...) in citation format, AND
- add the link to such documentation in this answer. Otherwise, you can attach the documentation in the next step.

IF NO COMPLETE DOCUMENTATION IS AVAILABLE:

Give a detailed description as answer, possibly attaching screen shots or a compete tutorial in word/pdf format in the next steps

Figure 4.3: Section 4 of the Task 4 results template for GeoBIM Benchmark submission.

[6] https://3d.bk.tudelft.nl/projects/geobim-benchmark/task4.html

4.3 GeoBIM Benchmark: Task 4

The specified task investigates *options for conversions* (software and procedural) of data from IFC to CityGML and CityGML to IFC. Off-the-shelf proprietary software and tools, such as Feature Manipulation Engine (FME) by Safe Software[7] and FZKViewer[8] were selected for the study of the conversion procedures and respective software performances. Alongside conversion results were analyzed with FME Data Inspector, BIMvision[9] and FZKViewer. Below are the experiments conducted for this benchmarking activity, and they present their submitted results with an online template submission form.

4.3.1 IFC to CityGML Conversion

To perform IFC to CityGML conversion, two approaches were adapted based on separate tools on FME platform: FME Quick Translator[10] and FME Workbench[11]. Following describes IFC to CityGML conversion techniques with selected tools.

4.3.1.1 FME Quick Translator

The Quick Translator tool is designed to perform fast and easy data conversions between numerous formats. A defined number of steps are established to use the translator tool

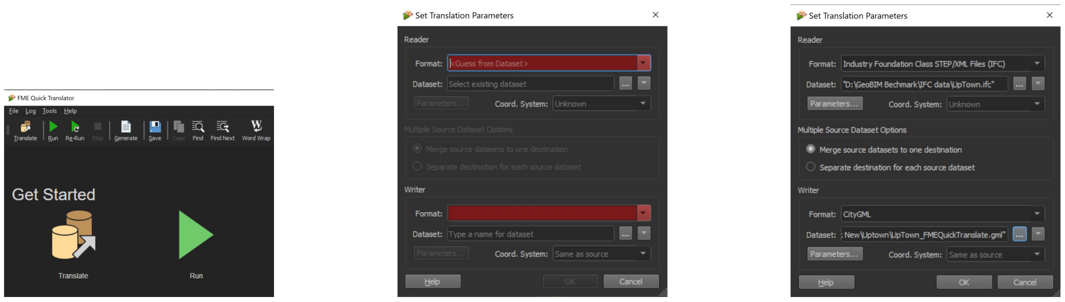
[7]https://www.safe.com/fme/

[8]https://www.iai.kit.edu/english/1648.php

[9]https://bimvision.eu/en/

[10]https://docs.safe.com/fme/html/FME_Desktop_Documentation/FME_QuickTranslator/Home_qt.htm

[11]https://docs.safe.com/fme/html/FME_Desktop_Documentation/FME_Workbench/Home.htm


FME Quick Translator
File Log Tools Help
Translate Run Re-Run Stop Generate Save Copy Find Find Next Word Wrap
Get Started
Translate
Run

Set Translation Parameters

Reader
Format: <Guess from Dataset>
Dataset: Select existing dataset
Coord. System: Unknown

Multiple Source Dataset Options
Merge source datasets to one destination
Separate destination for each source dataset

Writer
Format:
Dataset: Type a name for dataset
Coord. System: Same as source

Set Translation Parameters

Reader
Format: Industry Foundation Class STEP/XML Files (IFC)
Dataset: "D:\GeoBIM Bechmark\IFC data\UpTown.ifc"
Coord. System: Unknown

Multiple Source Dataset Options
● Merge source datasets to one destination
○ Separate destination for each source dataset

Writer
Format: CityGML
Dataset: : New\Uptown\UpTown_FMEQuickTranslate.gml"
Coord. System: Same as source

(a) (b) (c)

Figure 4.4: Procedural steps to perform an automatic IFC to CityGML conversion using off-the-shelf FME Quick Translator tool.

and applied across the IFC datasets to produce respective CityGML models. Below are the steps performed for conversion using this tool:

1. Open FME Quick Translator → Getting Started → Translate (see Figure 4.4a).
2. In *Translation* parameters select IFC as format and IFC model in *Reader*'s dataset. Select CityGML as *Writer*'s format and browse to select dataset output path and choose filename. Click *OK* to start conversion, as shown in Figure 4.4c).
3. Converted CityGML file is created at output path.

The steps mentioned above are followed for all IFC data models (listed in Section 4.2.1) when converting from IFC to CityGML, the procedure produces very generic models (Figure 4.5). In fact, with most (or all) entities being the generic one in the resulting standard data model, all entities from IFC are converted to CityGML *GenericCityObject*, instigating loss of associated semantic details (attributes and entity types) and elements' hierarchy. The procedure eliminates the reverse (bidirectional) conversion possibilities of the model (converting it back to IFC), thus discouraging interoperability.

Specifically for IFCGeometries data, not all geometry types like *IfcRevolvedAreaSolid*, *IfcExtrudedAreaSolid* and *IfcSweptDiskSolid*, were transformed by IFC reader of FME

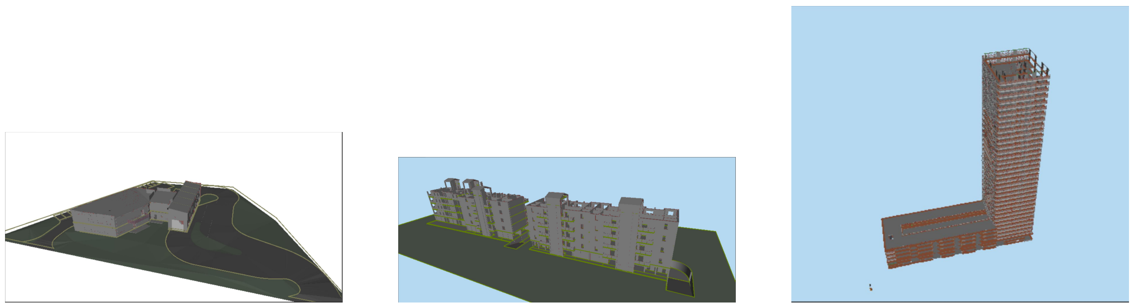


Figure 4.5: Quick translation results of IFC data into LoD4 CityGML models where not all geometries were transformed, and minimal semantics were kept with CityGML *GenericCityObject* feature type.

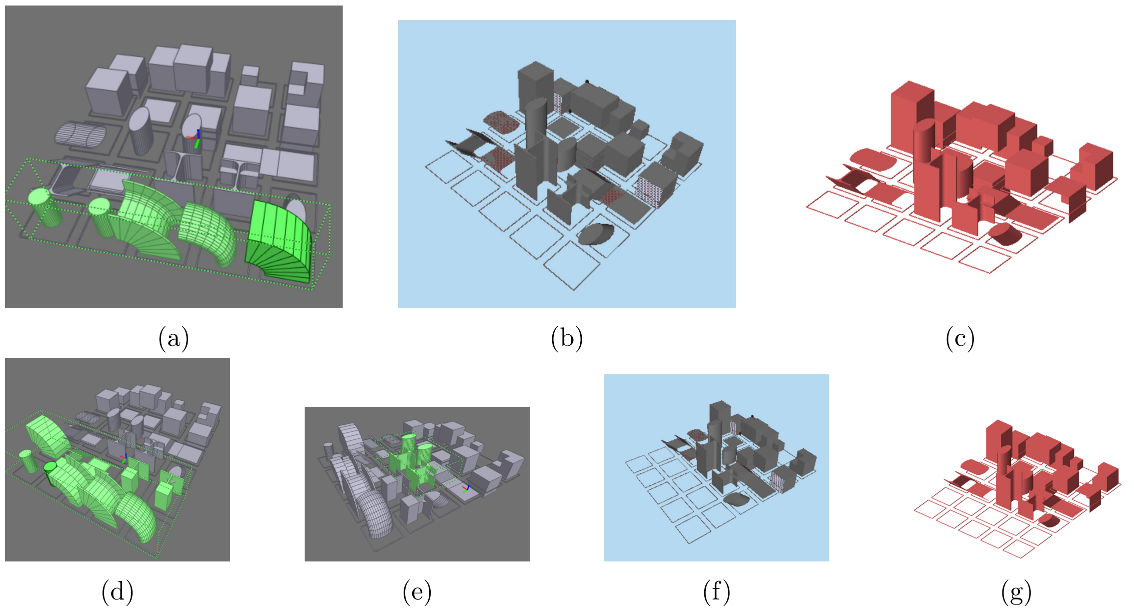


Figure 4.6: FME Quick Translator tool conversion attempt for IFCGeometries **(a)** IFC4 **(d, e)** IFC2x3, where geometry types *IfcRevolvedAreaSolid*, *IfcExtrudedAreaSolid* and *IfcSweptDiskSolid* highlighted are not converted; **(b, c)** and **(f, g)** show FME Data Inspector and FZKViewer visualizations respectively.

Quick Translator. Complex geometries in Figure 4.6 also highlight the converted geometry of the same type that have different dimensions (height) compared with originals. Only minimal semantics like *Guid*, *ifc_unique_id*, *ifc_parent_id* and *Name* attribute are kept after conversion.

4.3.1.2 FME Workbench

FME is a potent Extract Transform Load (ETL) platform that allows users to define a workspace fabricating translation or transformation procedures. Advanced users can execute custom or detailed mapping with a specific workflow to process data through the workbench interface. As a participant, the expertise level of using the FME platform was very beginner, therefore for this task, an existing FME workspace[12] (presented in Figure 4.7) was sourced, modified based on errors and workflow for testing the conversion of great detail IFC model to CityGML with LoD4. Below are the main steps performed for transformation using the FME workbench:

1. Open FME Workbench → Open custom workspace .fmv file and click *Run*.
2. In *User Parameters* select benchmark IFC model as source IFC file and set the path for destination CityGML document and output folder. Start the conversion process by clicking *OK*.
3. If successful, converted CityGML file will be generated at output path.

[12] https://knowledge.safe.com/articles/1025/bim-to-gis-intermediate-ifc-lod-300-to-lod-4-cityg.html

Last step of above the procedure is unreliable as it might be interrupted during the transformation, and only processed features are converted. The workspace comprises of numerous *transformers* that are defined for specific purposes, and can also be customized. Each transformer is an ETL component which takes input(s) and upon execution generates specific transformed output. Other transformers or writer components further uses output as an input to required transformation. For example, transformer *BinaryEncoder*

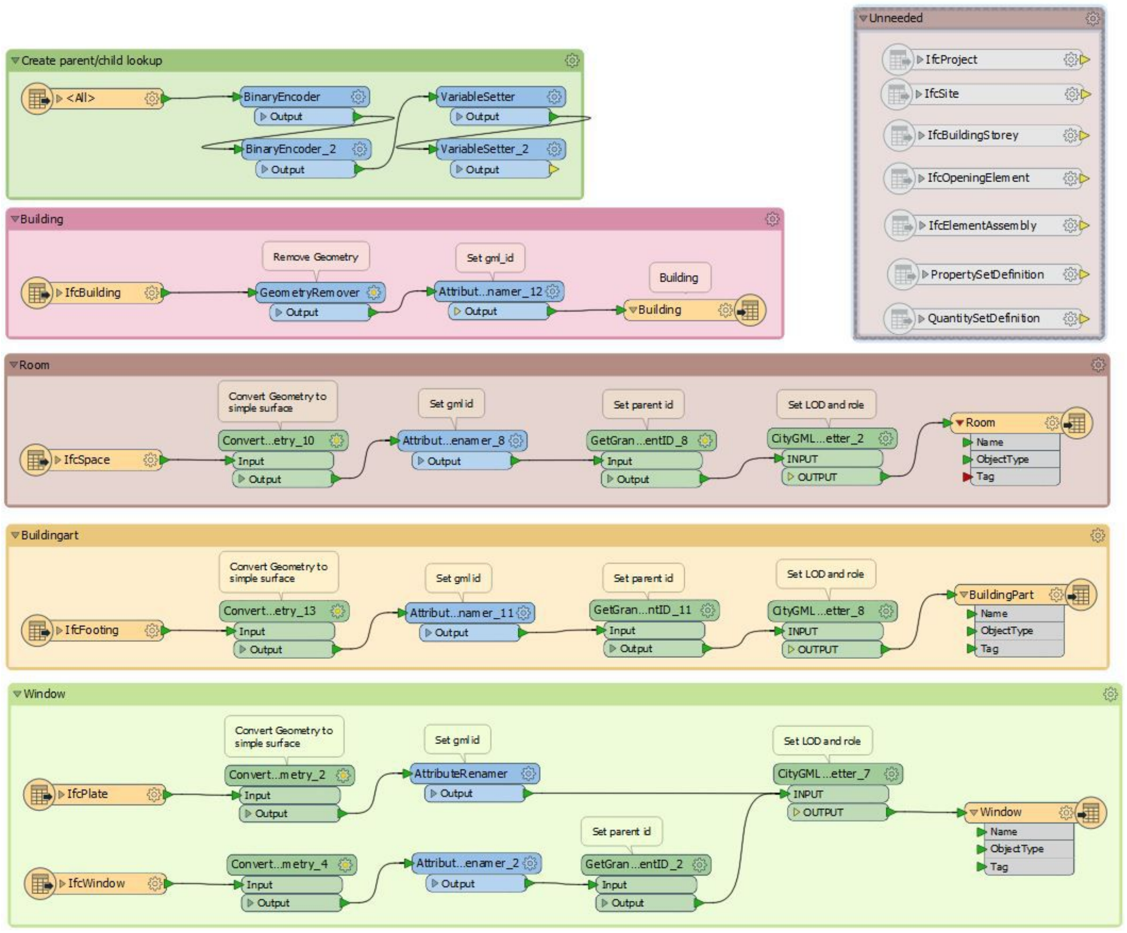


Figure 4.7: Partial visualization of a sourced example tutorial workspace for converting IFC model to CityGML with LoD4 using FME workbench where each block represents feature type with respective reader and writer, connected with FME transformers.

can convert attributes containing any data by encoding binary data to text using Base64 or Hex encoding methods.

Although the workspace presented in Figure 4.7, it describes comprehensive enough to handle most architectural IFC datasets, it gave only decent results for Myran.ifc data, while other data IFC models have missing features and interrupted conversion process by specific features. For example, *IfcMember* was not readable by the IFC reader, which highlights if certain features read are not supported by software (tool version or workspace), the translation will not be complete and/or suspended. The errors that occurred during translation and transformation do not provide much detail about the workspace process. Even transformation of processed features has different CityGML tags, e.g. BridgeWallSurface instead of BuildingWallSurface.

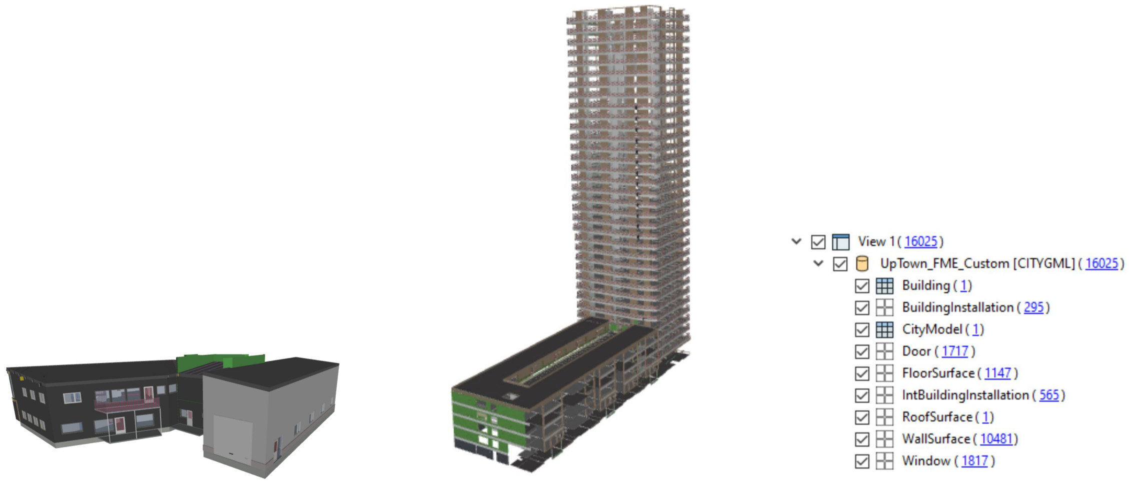


Figure 4.8: Conversion results of IFC data into LoD4 CityGML models using FME Workbench where limited features were transformed, and related semantics (attributes, entity types and hierarchy) were kept with respective CityGML feature types.

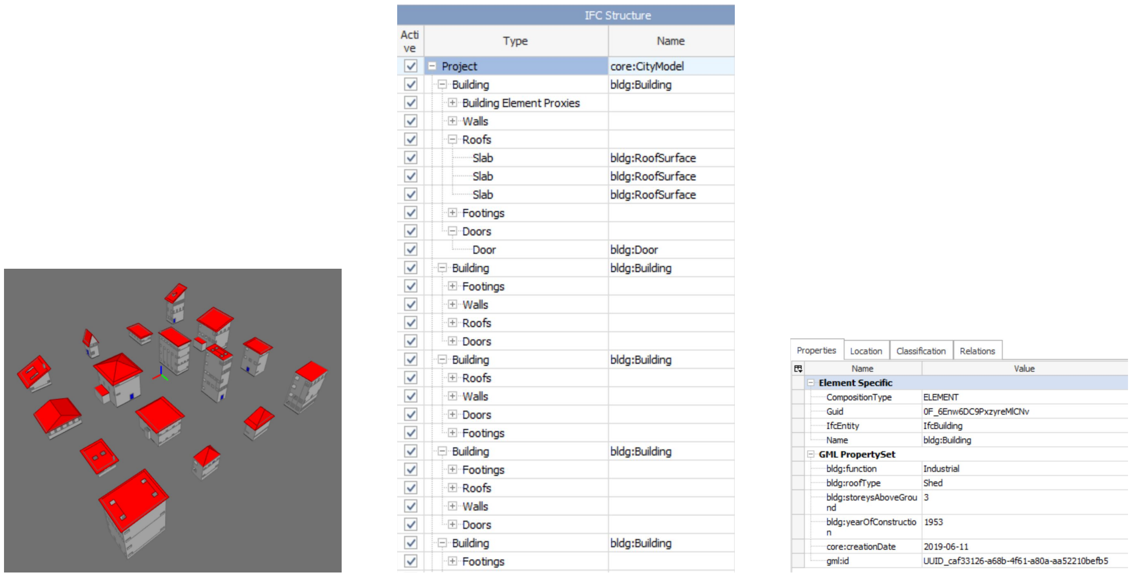
IfcMember feature types in IFC data are usually composed of complex or unsupported geometries. FME Workbench runs its IFC readers sequentially, but other transformers in the workspace attached with a reader's features run arbitrarily. Hence, the feature transformation process after reader till writer is random, unless done by feature caching. Therefore, if the transformation process of *IfcMember* or any other feature encounters any fatal error, it terminates the process at a random time. Nevertheless, disabling that feature transformer continues the translation process.

The conversion for Savigliano.ifc is not properly performed by extended workspace because of geometrical conversion error where FME workbench is unable to convert *IFMEAggregate* to *IFMEMultiSurface*, and terminated process results only *Door* features. However, CityGML writer in FME indicates *IFMEAggregate* is convertible format for *lod4multisurface* with *IFMEMultiSurface* as valid geometries. Similarly, *IFMEMultiCurve* geometric conversion error in addition to IFC reader limitation of supporting solids like *IfcAdvancedBrep* and *IfcSurfaceCurveSweptAreaSolid* and feature types of *IfcTypeObject* (i.e. *IfcDoorStyle* and *IfcWindowStyle*) for UpTown.ifc data results in partial building features (Figure 4.8).

4.3.2 CityGML to IFC Conversion

Initially, CityGML to IFC conversion experiments using FME Quick Translator resulted in IFC models with only hierarchical information with no semantic or geometric details, and error log of unable to write objects of CityGML type like *WallSurface*, *RoofSurface* and *GroundSurface*. Hence, another proprietary software of FZKViewer was investigated for the CityGML to IFC conversion task and below are the conversion steps performed using this selected tool:

1. Lanuch FZKViewer → Open → Open GML File.
2. Browse CityGML dataset (file mentioned in Section 4.2.1). Click *Open* (here Spatial Reference System is already set to EPSG-28992 - Amersfoort / RD New).
3. Click File → Export → IFC.
4. Select destination folder and add filename. Click *Save*. Before *Save* in Options you can choose export version (i.e. IFC2x3 or IFC4) and even IFC specification (i.e. *.ifc*, *.ifcxml* or *.ifczip*).

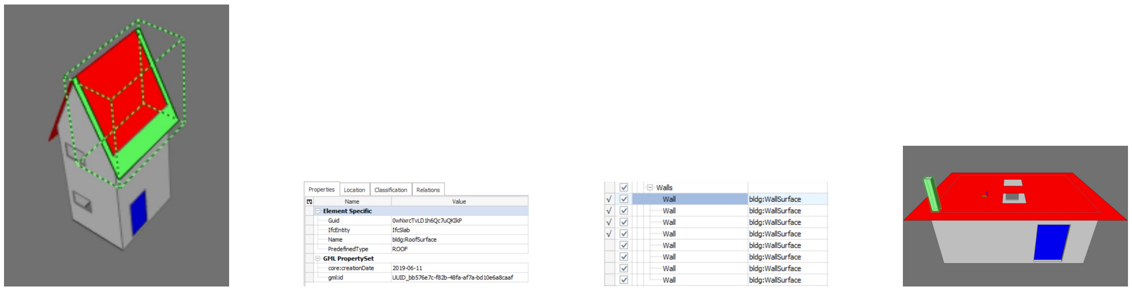


IFC Structure

Active	Type	Name
✓	Project	core:CityModel
✓	Building	bldg:Building
✓	Building Element Proxies	
✓	Walls	
✓	Roofs	
✓	Slab	bldg:RoofSurface
✓	Slab	bldg:RoofSurface
✓	Slab	bldg:RoofSurface
✓	Footings	
✓	Doors	
✓	Door	bldg:Door
✓	Building	bldg:Building
✓	Footings	
✓	Walls	
✓	Roofs	
✓	Doors	
✓	Building	bldg:Building
✓	Roofs	
✓	Walls	
✓	Doors	
✓	Footings	
✓	Building	bldg:Building
✓	Footings	
✓	Roofs	
✓	Walls	
✓	Doors	
✓	Building	bldg:Building
✓	Footings	

Name	Value
Element Specific	
CompositionType	ELEMENT
Guid	0F_6Enw6DC9PxzyreMlCNv
IfcEntity	IfcBuilding
Name	bldg:Building
GML PropertySet	
bldg:function	Industrial
bldg:roofType	Shed
bldg:storeysAboveGround	3
bldg:yearOfConstruction	1953
core:creationDate	2019-06-11
gml:id	UUID_caf33126-a68b-4f61-a80a-aa52210befb5

Figure 4.9: BIMvision visualization of converted IFC model from Buildings LoD3 CityGML data using FZKViewer shows correct semantic and geometric transformation of features.



Properties | Location | Classification | Relations

Name	Value
Element Specific	
Guid	0wNxrcTvLD1h6Qc7uQKIkP
IfcEntity	IfcSlab
Name	bldg:RoofSurface
PredefinedType	ROOF
GML PropertySet	
core:creationDate	2019-06-11
gml:id	UUID_bb576e7c-f82b-48fa-af7a-bd10e6a8caaf

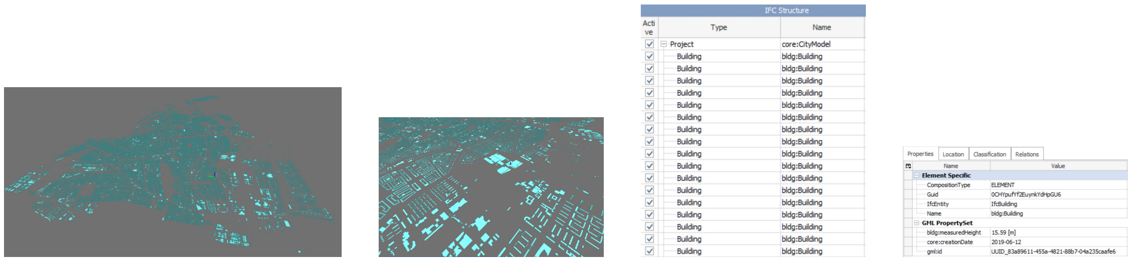
	✓	Walls	
√	✓	Wall	bldg:WallSurface
√	✓	Wall	bldg:WallSurface
√	✓	Wall	bldg:WallSurface
√	✓	Wall	bldg:WallSurface
	✓	Wall	bldg:WallSurface
	✓	Wall	bldg:WallSurface
	✓	Wall	bldg:WallSurface
	✓	Wall	bldg:WallSurface

Figure 4.10: Feature details of converted Buildings LoD3 IFC model shows correct mapping of entities.

The results of automatic procedural steps mentioned above for FZKViewer produces correct semantics (attributes and entity types) and present relatively correct changes in geometry for CityGML to IFC conversion. The accurate mapping of converted *BuildingsLoD3.gml* data presented in Figure 4.9 shows the hierarchy of detailed features is kept in the converted model, and correct mapping entities are assigned, as shown in the Figure 4.10. Even generic attributes from GML file are converted to *GML:PropertySet* in IFC attributes.

AmsterdamLoD1.gml is a huge CityGML dataset generally to check software and system's compatibility of handling large scale dataset. The chosen system's hardware was able to open the file and export the IFC model without crashing. The process took a considerably short time (approx. 8 minutes) to complete and generate only building footprints (no building geometry) in LOD100 IFC model models. Similarly, with semantic details, as shown in Figure 4.11, while other elements from CityGML data like bridges, roads were discarded automatically in export by FZKViewer.

Nevertheless, FZKViewer supports the CityGML to IFC conversion, but handling



IFC Structure

Active	Type	Name
✓	Project	core:CityModel
✓	Building	bldg:Building
✓	Building	bldg:Building
✓	Building	bldg:Building
✓	Building	bldg:Building
✓	Building	bldg:Building
✓	Building	bldg:Building
✓	Building	bldg:Building
✓	Building	bldg:Building
✓	Building	bldg:Building
✓	Building	bldg:Building
✓	Building	bldg:Building
✓	Building	bldg:Building
✓	Building	bldg:Building
✓	Building	bldg:Building
✓	Building	bldg:Building
✓	Building	bldg:Building
✓	Building	bldg:Building

Name	Value
Element Specific	
CompositionType	ELEMENT
Guid	0CHYpufYf2EuynkYdHpGU6
IfcEntity	IfcBuilding
Name	bldg:Building
GML PropertySet	
bldg:measuredHeight	15.59 [m]
core:creationDate	2019-06-12
gml:id	UUID_83a89611-455a-4821-88b7-04a235caafe6

Figure 4.11: CityGML to IFC conversion results of Amsterdam LoD1 using FZKViewer export functionality representing building footprints with correct attribute information.

multiple LoDs export is not valid even if the process is complete. Since there is no support in available software for IFC data with multiple LODs, the exported IFC file of *RotterdamLod2LoD1.gml* is not validated and readable by BIM software.

4.3.3 Summary

Results in previous sections show that using FME workbench tool for translations or transformations requires prior knowledge. Otherwise, with minimal expertise, either of the two tools, i.e. FME workbench or FME Quick translator, leads to irregular geometrical conversion and missing features.

Single workspace design for IFC to CityGML or CityGML to IFC data translation and transformation in FME Workbench is impracticable against different datasets. Even the execution of the same workspace across different Workbench software versions showing compatibility issues were also observed. As mentioned earlier, the purpose of the benchmark **Task4** activity was to investigate the off-the-shelf tools and procedures for the IFC to CityGML and CityGML to IFC interoperability, creating functional workspaces with workflow for discrete datasets is time-consuming research itself and out of scope for this study.

FME Quick Translation provides limited geometrical conversion from IFC to CityGML with loss in semantic and structural information. On the contrary, FZKViewer software tool that is generally used to view and analyze GIS and BIM models, not directed as an off-the-shelf conversion solution. However, it supports promising options of exporting data into multiple formats like Collada, STL, including CityGML to IFC with different IFC formats. Exporting multi–LOD IFC model against multi-LoDs CityGML data from software like FZKViewer requires extra efforts, with a possible solution to separate export model filtered on LoD.

Overall, the hardware used in this participation was adequate to run software tools for conversion procedures with a minimal number of crashes during the conversion process. Table 4.2 shows the time interval of each conversion task for IFC and CityGML model across FME Workbench, FME Quick Translate and FZKViewer tools. For large datasets like *"UpTown.ifc"* and *"Amsterdam.gml"*, the transformation takes longer to complete,

IFC to CityGML Conversion			
Name	**File Size**	**Experiment 1**	**Experiment 2**
Myran	27.78 MB	42s	~3m
Savigliano	22.07 MB	90s	–
UpTown	246.82 MB	44m 59s	57m 55s
IFCGeometries	31 KB	2	–
	27 KB	1.5	–
CityGML to IFC Conversion			
Buildings	1.36 MB	~2s	–
Rotterdam	34.72 MB	–	–
Amsterdam	5.02 GB	~8m	–

Table 4.2: Conversion time for Task4 attempts, Experiment 1 and 2 for IFC to CityGML denotes FME Quick Translator and FME Workbench conversions respectively, for CityGML to IFC Experiment 1 is for FZKViewer conversions.

but the system was able to complete the process without crashing the software.

4.4 Study Analysis and Conclusion

Although, Noardo et al. (2019b) shows prominent interest for the **Task4** participants' registration, the moderate result attempts submitted in Noardo et al. (2020a) highlights the challenges in IFC $\longleftrightarrow$ CityGML conversion procedures and room for improvements. While there were successful attempts of IFC to CityGML conversion delivered for the benchmark results, including a consistent conversion technique from a typical BIM structure (solids and detailed elements) to GIS (external, less detailed surfaces) by means of specific workflow in FME and ArcGIS Pro-Data Interoperability extension, and another attempt of using IFC2CityGML tool (Stouffs et al. 2018); other delivered conversions also produced generic models with *GenericCityObject* elements. Even with successful attempts, processes were very complex with outputting irregular geometries and involved limitations of relying upon CityGML v.3. Therefore, prior knowledge and expertise of tools like FME Workbench are required to achieve relatively successful results.

For the CityGML to IFC conversion results, geometric processing is minimal, but the semantics associated with elements changed correctly according to the destination data model. FZKViewer provides better semantic and geometric mapping and transformation of CityGML to IFC data by keeping the model structure, enabling bidirectional conversion compared to the off-the-shelf FME Quick Translation tool. Furthermore, exported models from FZKViewer show consistency in file size if experiments are to be repeated, unlike

FME Quick Translator and Workbench.

In terms of software potential, IFC reader in FME Quick Translator does not support all geometry types as well as transformers unable to interpret and translate complex geometries. Even a few converted complex geometries have different dimensions compared with the original. Visualization tools, on the other hand, provide a different rendering of normal surfaces in geometry across the software of FZKViewer and FME Data Inspector (Figure 4.6).

A conclusion drawn from this benchmarking activity is not easy, and existing software tools and procedures offer systematic interoperability between BIM and GIS. It should be also be noted from the study the standards bodies themselves and the community are focusing on this interoperability problem. The reason is the high-level complexity of standards and specific approaches to defining entities (including geometries), making them very difficult to implement, understand, deploy, and use.

Therefore, following this GeoBIM benchmark activity and literature for the need of interoperability of BIM and GIS, instead of data conversions leading to information loss or designing application-oriented extensions, further study is conducted with the aim towards achieving semantic interoperability among data models of BIM and GIS using Semantic Web Technology, as the main goal of this research. A study focused on bringing heterogeneous data in a common format of semantic web domain for seamless integration – from where information can be extracted and facilitated for substantially generalized future solutions.

Chapter 5

Methodology: Ontology Generation of an XML Schema

In this chapter, the comprehensive framework of the tailored methodology – automatic ontology generation of an XML schema – is presented as a step towards broad research to achieve interoperability of BIM and GIS data with sensors information using semantic web technology stack. The proposed transformation methodology is an extension of PIXCO (Hacherouf et al. 2019) and Janus (Bedini et al. 2011) frameworks, which implements essential components and logical associations for the maximum transformation of XSD elements into their appropriate OWL representations. The methodology focuses on generating ontology from XML-based ifcXML and CityGML schemas of BIM and GIS domain, respectively.

The methodology in this dissertation proposes the framework of ontology generation as a first-module of the conceptual framework initially designed to commence this research study for the broad research approach of semantic interoperability of BIM and GIS data linked with IoT (Internet of Things) by exploiting semantic web techniques. However, based on the complexity of BIM-GIS integration problems, limitations with nascent research in the semantic web domain, and unavailability of working prototypes for semantic integration (detailed in Chapter 2 and 3); the scope of the devised study was focused to the only first-module (ontology generation) of broad research. It focuses explicitly on establishing a formal method for the semantic representation of schema elements of ifcXML and CityGML formats. Henceforth, the first module of the preparatory framework is developed and presented in this research to achieve the seamless information exchange's further integral goal among BIM, GIS, and IoT.

The organization of this Chapter is as follows: Section 5.1 features the preparatory conceptual framework with three-modules of initial broad research approach of semantic integration of BIM and GIS with IoT data (referred to as preparatory or primary research methodology); Section 5.2 elaborates the proposed design and development first-module of the preparatory framework for automatic ontology generation from XML schemas (onwards referred as a research methodology or contemporary research); Section 5.3 delves with in-depth details of improvements made in initial framework and considerations in the framework; and in the last Section 5.4, a summary of this Chapter for the established methodology in the direction of implementation is addressed.

5.1 Preparatory Conceptual Framework for Semantic Integration

To accomplish primary research objectives, a conceptual framework for comprehensive semantic level integration of BIM and GIS further incorporating IoT data is outlined in Figure 5.1; addressed as preparatory (primary) methodology. The span of framework design is a composite of multiple processes and algorithms. Accordingly, indicated methodology is divided into three modules and presented in the following sub-sections: (i) a formal method of ontology generation utilizing XML schema documents of BIM and GIS data with former architecture named Ontology Generation for Geospatial Data (OGGD) is introduced in Section 5.1.1; (ii) mapping of discretely generated BIM and GIS ontologies as an innovative approach involving semantic and structural alignment techniques are proposed in Section 5.1.2 as Ontology Alignment for Geospatial Data (OAGD), that utilizes ontologies generated in the previous module for better mapping of entities among both domains; and finally, generation of RDF graph by extracting information from respective XML-based data-format of BIM, GIS, or IoT, and mapping it with cross-domain ontology (obtained from the second module) to achieve integrated information as the common data model is described in Section 5.1.3.

As mentioned earlier, the extensive preparatory framework primarily focuses on semantic interoperability of data from three domains to essentially address seamless information exchange. From the three-modules of the manifested framework presented in Figure 5.1, only the first-module is implemented in this research study; nonetheless, all are discussed in the following sections:

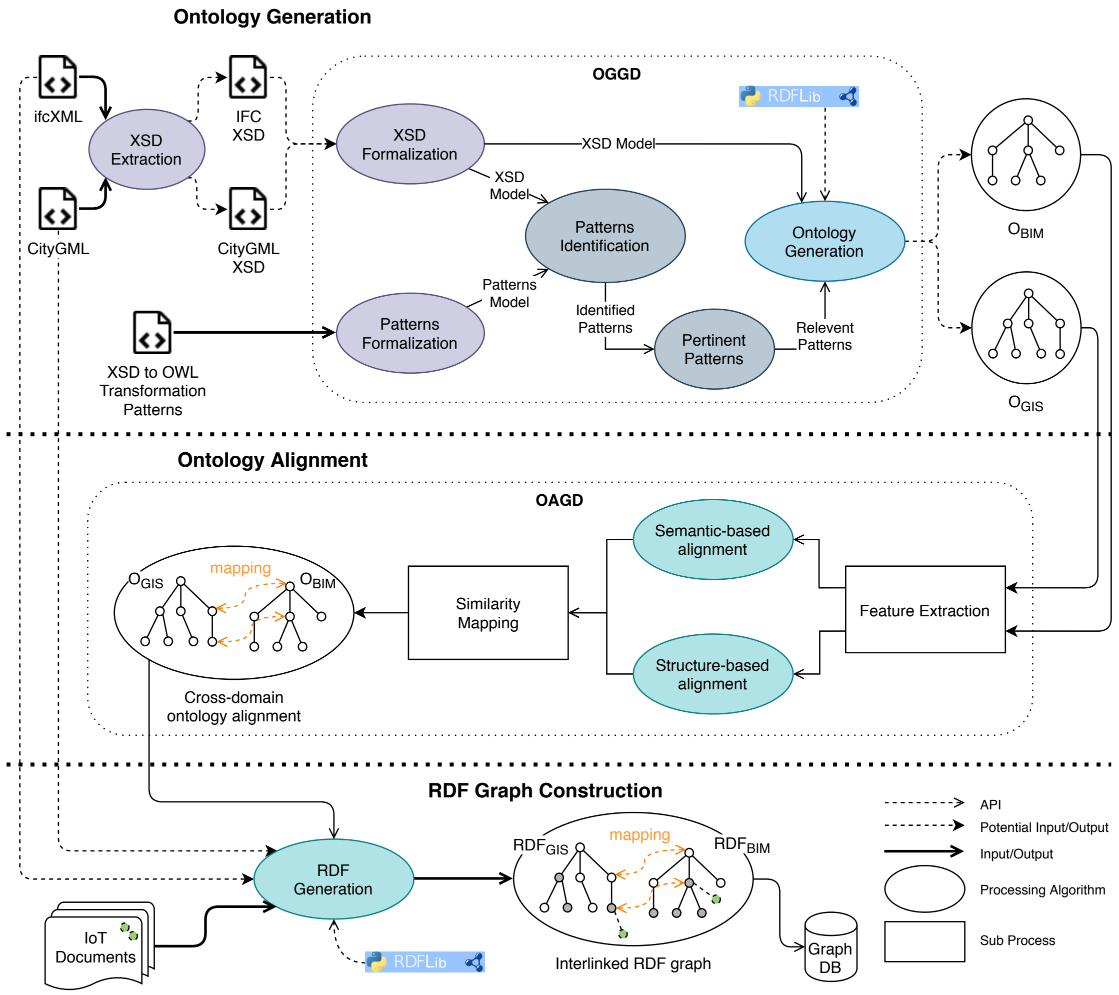


Figure 5.1: A comprehensive conceptual framework of primary methodology introduced for the seamless integration of BIM and GIS data linked with IoT.

5.1.1 Ontology Generation for Geospatial Data (OGGD)

In the primary framework illustrated in Figure 5.1, OGGD, as the first-module, defines the initial steps of automatically generating ontology for a given schema document. Ontology models are generated in an OWL format by extracting information from XSD of ifcXML and CityGML standards.

The design of OGGD itself is based on three steps (phases) (see Figure 5.1). The first step utilizes XSD constructs of XML schemas to develop a formal model named Formal Structure of XML Schema $\mathcal{FS}(\mathcal{XS})$. Alongside, it performs formalization of defined sets of XSD to OWL Janus transformation patterns in a Formal Concept Analysis (FCA) context. In the second step, patterns are identified across XSD constructs using the $\mathcal{FS}(\mathcal{XS})$ model and transformation patterns, which further proceeds to pertinent patterns for each specified XSD construct. Finally, in the last step, each XSD construct associated with an appropriately identified transformation pattern is represented as an ontology fragment, later adjoined into a consolidated OWL model. Henceforth, for a given x XSD, respective ontology models of O_x is generated from aggregated ontology fragments. In-depth details of this module are discussed in Section 5.2 as the major contribution of this research study.

5.1.2 Ontology Alignment for Geospatial Data (OAGD)

The second module design of the primary conceptual framework includes the alignment of cross-domain ontologies named OAGD; an automatic alignment (mapping) process that determines the semantic-relations (correspondences) between concepts (entities) represented as *classes* and *individuals*, among generated O_{BIM} and O_{GIS} ontologies from the previous module.

In this module, the features are extracted from O_{BIM} and O_{GIS} ontologies in a sets of *classes*, *properties* and *individuals* along with *annotations*. Further, the sets are stacked in a formal form of lists or dictionaries for initiating the efficient retrieval of specific entities and their correspondences. Next, to procure the alignment and imply potential ontology mapping relations (e.g., one-to-many, many-to-one, or many-to-many), the semantic-based *Word2vec* (Mikolov et al. 2013) and structure-based *Node2vec* (Grover and Leskovec 2016) algorithms are implied. The integration of these machine learning algorithms in OAGD will estimate the similarity between entities for better alignment results. In particular, these algorithms utilize each entity from an ontology graph to be learned and represented in *vector* formats for estimating the similarity between entities composing a correspondence. Furthermore, from these *vectors*, an aggregated confidence value representing similarity assessment between mapping entity nodes can be estimated, which potentially identifies two entities aligned by their similarity level. Such process of aggregating similarity measures is defined as *similarity aggregation*, and can be applied using *weighted average similarity* proposed in Acampora et al. (2013) which illustrates Equation 5.1 applies aggregated similarity value for c as:

$$sim_{aggregate}(c) = \sum_{i=1}^{h} w_i \times sim_i(c) \text{ subject to} \sum_{i=1}^{h} w_i = 1 \tag{5.1}$$

where for i^{th} similarity measure, w_i is the associated weight of h similarity measures and $sim_i(c)$ is the similarity value computed against each correspondence c for an alignment A with the k correspondences; such that $i = \{1, 2, ..., k\}$, for correspondence c_i.

Geng et al. (2020) is a leading study that implies alignment technique employs semantic-based and structure-based similarities aggregated together according to the *weighted av-*

erage similarity between two entity nodes determine the similarity results between the entities of two domains. Consequently, indicating confidence is estimated as *equal-to* relation between every two entities of respective ontologies. Indicated study also evaluates the semantics relation recognition by utilizing well-known information retrieval precision and recall assessment. Nevertheless, the OAGD framework built on this procedure presented for ontology alignment is comprehensive and supported by an evident study to correlate between two cross-domain entities of BIM and GIS. Due to the tangled and supplementary scope of the ontology generation process, our research was limited to only the first module of the primary framework. Furthermore, a correlation between entities in complex ontologies demands to precede enriched ontology models to be reckoned in this module. This alignment OAGD module was primed, indicating aligned entities would be investigated and integrated with the future study scope of linking cross-domain ontology of BIM and GIS systems.

5.1.3 Semantic Graph Generation

OWL representation of ontology assists reasoning and inference. In contrast, RDF is the core of the semantic web and annotates to link data. In the last module of the preparatory framework, RDF graphs will be produced for data from ifcXML and CityGML datasets utilizing inference from respective generated ontology models, O_{BIM} and O_{GIS} in the first-module, and linked cross-domain ontology from second-module.

The similarity correspondence of cross-domain ontology is to bridge the gap between

BIM and GIS. It shows the prospect of generating an interlinked RDF graph, a common data model in the semantic web. Moreover, sensor data (IoT) as linked data can potentially be interlinked with this common data model in representing it in subject-predicate-object format. Henceforth, these RDF graphs will potentially represent the information exchange of sensors adjoined with geospatial entities in a cross-domain integrated graph structure. RDF graphs produced for such semantically rich geospatial and IoT information are considerably sizable and likely requires optimizations; for querying, storing and managing operations. To address these, cluster-based and community optimization techniques can be applied for manipulating RDF graphs with exhaustive information.

Nevertheless, the primary framework modules provide a comprehensive workflow towards achieving the interoperability between BIM and GIS and using a semantic web technology stack, a potential system with seamless information exchange of integrated BIM and GIS with IoT data. However, the scope of this research study was made limited to only the first-module due to limitations mentioned earlier. This dissertation focuses only on the first module of the primary framework, delegating detailed second and third modules of the preparatory research framework to be scrutinized in succeeding studies.

5.2 EPIXCO Framework

The contemporary framework of this research study focuses explicitly on OGGD (Section 5.1.1) by proposing a detailed extension in the design of its first module of the prepara-

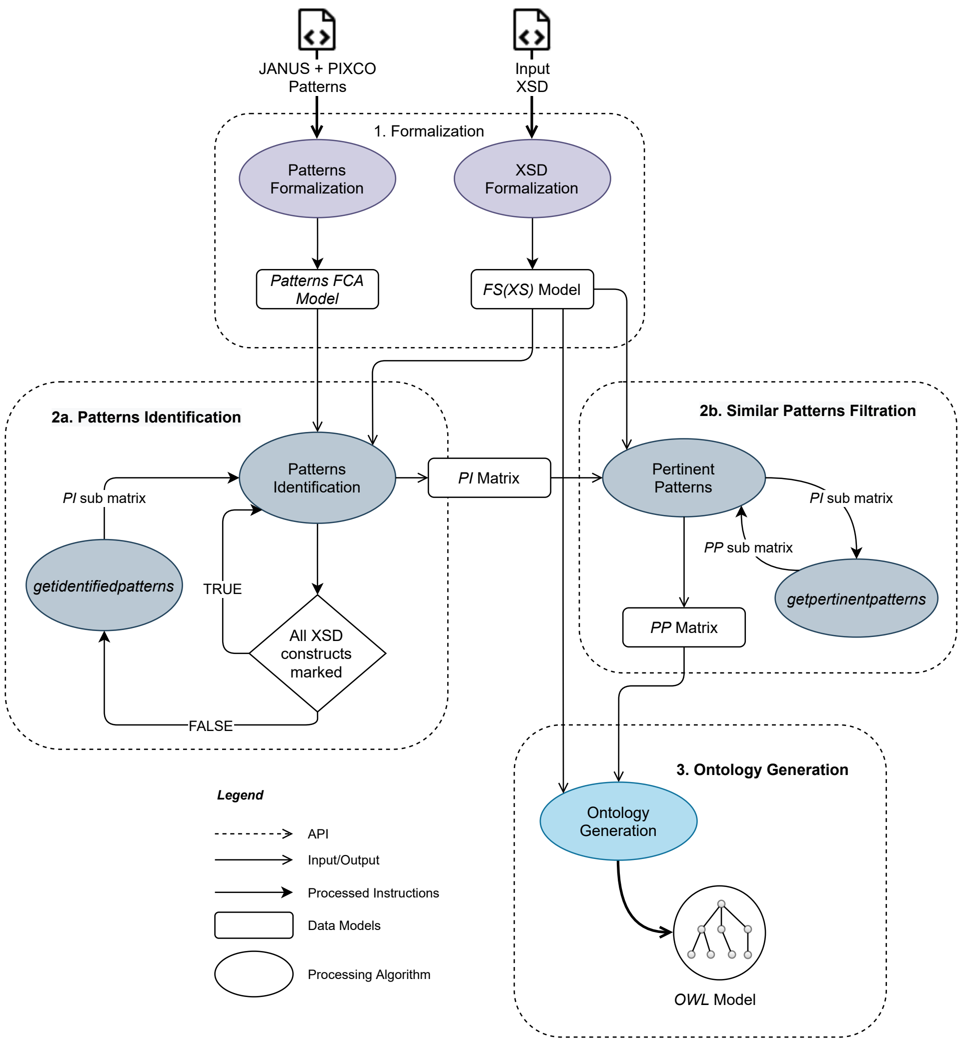


Figure 5.2: A formal methodology design of EPIXCO for automatic ontology generation of XML schema by using transformation patterns.

tory framework to address the automatic generation of ontologies from XML schemas, explicitly focusing on XSD of ifcXML and CityGML formats.

Primarily, the tentative design of OGGD underlines the formal architecture from PIXCO (Hacherouf et al. 2019) to utilize an extensive patterns-based transformation approach of ontology generation from XML schema documents. To complement this, a comprehensive framework **EPIXCO** (**E**nhanced **P**atterns **I**dentification for **X**SD **C**onversion to **O**WL) is introduced in Figure 5.2 as an improvement to the initially developed OGGD design (Usmani et al. 2020) by enabling an extensive identification of transformation patterns introduced in Janus and revamping the formalization and ontology generation methods from PIXCO.

For better illustration, going forward few terminologies will be used interchangeably: transformation patterns or simply patterns are collective of XSD to OWL correspondence patterns from PIXCO and Janus approach. XSD components or tags are elements of XML schema where each XSD construction (construct) comprises an XSD component and attributes. The EPIXCO methodology implements multiple algorithms assisted by sub-algorithms and is majorly categorized into three-phases which are elaborated in the following sections:

5.2.1 Formalization of XSD and Transformation Patterns

The formal modelling process of data and related components encourage swift information retrieval and exchange across the framework. In this phase, an input XSD schema is formalized in a model along with adapted transformation patterns. An XSD is a collection of XSD constructs in a hierarchical structure, where each construct represents an XSD element accompanied by one or more attributes. Collectively, these XSD components forming the XSD construct have several representation cases (check Section 3.3 for details). Subsequently, a transformation pattern defines a correspondence rule of transforming elements from XSD constructs into OWL axioms. Therefore, for the essential process of ontology generation by transforming XSD elements to OWL, two principle information sources, input XSD schema and patterns, are formalized in extended mathematical models of $\mathcal{FS}(\mathcal{XS})$ and context-based FCA model, respectively. They are presented as follows:

5.2.1.1 XSD Formalization

To formalize an XSD schema comprising a set of XSD constructs, a formal model introduced in PIXCO, Formal Structure of XML Schema ($\mathcal{FS}(\mathcal{XS})$), is redefined in EPIXCO as sets of 7-tuple $S = (E, A, V_A, C, H_C, L_T, L_e)$ comprising:

1. Set E containing all **elements** of an XSD: $E = \{$`all`, `attribute`, `element`, $\ldots\}$.
2. Set A containing all **attributes** of an XSD: $A = \{$`name`, `type`, `minOccurs`, $\ldots\}$.
3. Set V_A containing all **attributes values** of an XSD schema.
4. **Collection of functions** C for all **constructs** of an XSD, where each construction function represents an XSD element accompanied by one or more attributes. The collection C can be modelled as:

   $C = \{C_1, \ldots, C_n\}$ where n $\in \mathbb{N}$ and $C_i : \langle E \longrightarrow A \times V_A \rangle$
5. **Collection of functions** H_C representing a **hierarchy of constructs** in an XSD:

   $H_C : C \longrightarrow 2^C$
6. Set L_T containing all XSD **types** of an XSD schema.

7. Set L_e containing `name` **attribute values** of all `element` constructions in an XSD schema, where `element` $\in E$.

The $\mathcal{FS}(\mathcal{XS})$ model is redefined for the effective modelling of XSD components with further inclusion of L_T and L_e sets useful for transforming perplexing XSD schemas by their utilization in the succeeding processes of pattern identification and later ontology generation. L_T is the collection containing all XSD types of an XSD schema, while L_e is to track `element` components for their construction. The created $\mathcal{FS}(\mathcal{XS})$ model is exploited in succeeding phases of the contemporary framework and helps formally manipulate the extracted information of XSD components.

5.2.1.2 Patterns Formalization

To formalize XSD to OWL transformation patterns, a Formal Concept Analysis (FCA) (Bělohlávek 2008) context method is used in PIXCO to create a *concept lattice*. The FCA model has a triple form *context* (G, M, I) whose elements contain a set of *objects* (G) sharing a set of *attributes* (M) that represents a set of concepts (I), such that I is a binary relation between G and M represented as $I \subseteq G \times M$. In our context, the *objects* (G) are patterns, *attributes* (M) are XSD elements and I relates with patterns against elements or vice versa. Plainly, to formalize Janus and extended PIXCO patterns, the formal FCA model of triple form context can be represented with following considerations:

1. G as set of all patterns i.e. $G = \{1, 2, 3, \ldots, 43\}$.

2. M as set of XSD elements in G patterns e.g. $M = \{\texttt{simpleType}, \texttt{complexType}, \ldots\}$.

	simpleType	union	complexType	restriction	enumeration	minInclusive	maxInclusive
1	x						
2	x	x					
3							
4	x			x	x		
5	x			x		x	x
6			x				

Table 5.1: A partial matrix to illustrate the formal context for patterns and XSD elements.

3. gIm or $(g, m) \in I$ as interpretation function to recognize pattern $g \in G$ has element $m \in M$, for example:

   $1I$`simpleType` shows relation I with pattern 1 using XSD element `simpleType`.

4. Set $B \subseteq M$ induced with formal operator of concepts $^{\downarrow}$ to identify pattern(s) sharing all elements can be defined as:

   $^{\downarrow} : 2^M \longrightarrow 2^G$

   $B^{\downarrow} = \{\, g \in G \,|\, \text{for each}\, m \in B\ :\ (g, m)\ \in I\}$ and $B^{\downarrow} \subseteq G$

   For example, **if** $B = \{\texttt{simpleType}, \texttt{restriciton}, \texttt{enumeration}\}$ **then** $B^{\downarrow} = 4$.

The 43 transformation patterns of Janus (Bedini et al. 2011, Table I-VI) and PIXCO (Hacherouf et al. 2019, Table 4-5) includes in total 27 XSD elements resulting in a very large context matrix with 43 rows and 27 columns. A fully created formal context of patterns (rows) and XSD elements (columns) occupies large space. Therefore, limited set of patterns (*objects*) and elements (*attributes*) are considered, and their partial matrix is presented in Table 5.1 for context matrix understanding. A pattern sharing an element at matrix position (i, j) is marked (**x**), if and only if pattern i uses the XSD element j.

A complete concept lattice of transformation patterns is provided in Figure 5.3 for full

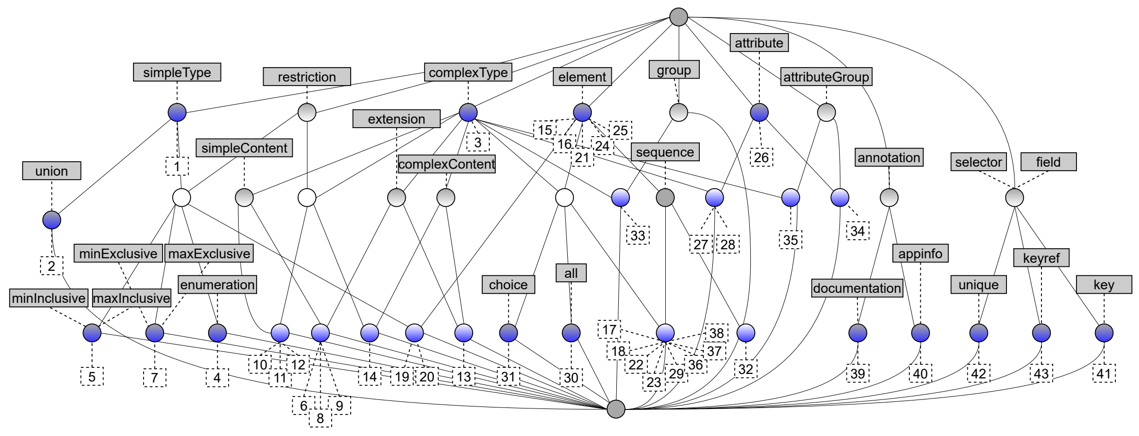


Figure 5.3: A full concept lattice of patterns in the FCA context with transformation patterns and their respective XSD elements, inspired by Hacherouf et al. (2019). Filled circle denotes intersecting node with patterns and XSD elements, half gray nodes are only XSD elements, half blue are patterns nodes, and empty nodes are simple connections.

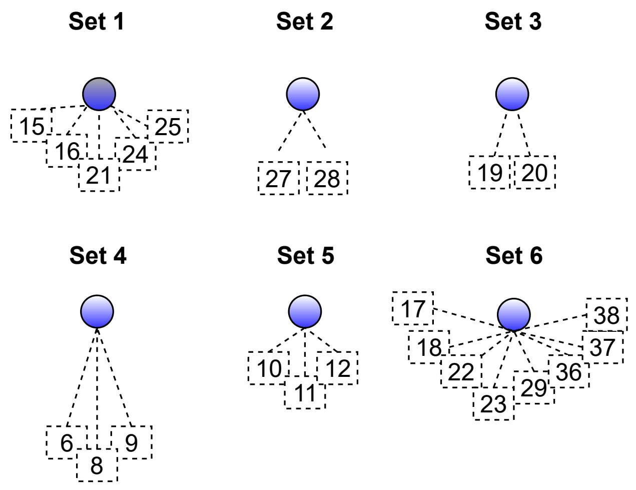


Figure 5.4: Sets of similar patterns listed in PIXCO framework extracted from concept lattice Figure 5.3.

context matrix representation of the concept linking of patterns with elements where for a given concept, a set of patterns are linked with XSD elements. For example, for concept I with [Element:`element`], set of patterns it refers to is $\{15, 16, 21, 24, 25\}$ (Bedini et al. 2011, Table IV). Moreover, if elements are shared in concept like [Element:`element`, `annotation`] implying concept that is not *closed*, then there are two paths: (a) path including XSD element `appinfo` results pattern set {40}; (b) path with `documentation` maps to pattern set {39} (see lattice in Figure 5.3). Hence, it can be implied that `element` node is directly linked with `annotation`, and indirectly linked with `appinfo` and `documentation`, with other XSD elements like `simpleType` and `complexType` as well. Henceforth, the formalization of patterns using a concept lattice is convenient to retrieve a formal concept for a given context (G, M, I).

5.2.2 Patterns Identification and Filtration

Identifying patterns for an XSD block is a complicated process and further rectifying one or two patterns precisely among 43 patterns becomes more challenging. The reason for complexity originates from several design styles of single XSD block representation and selecting a particular pattern from a high number of transformation patterns. Therefore, to generate an ontology fragment of an XSD block of constructs, it has to match exactly with a defined XSD structure rule of a pattern. Therefore, this phase is subdivided into two steps:

5.2.2.1 Patterns Identification

In this phase, appropriate patterns are identified for each block of XSD constructs by exploiting formal models from the first phase. Two algorithms (*PatternsIdentification* and *getIdentifiedPatterns*) are implemented for appropriate identification process (see phase 2a. of Figure 5.2). Both algorithms utilize helper functions (sub-algorithms listed in (Hacherouf et al. 2019, Table 9, 11) and extended in Table 5.2 and 5.3), for their exhaustive implementation.

The first algorithm *PatternsIdentification* (Algorithm 1) takes $\mathcal{FS}(\mathcal{XS})$ model of an XSD schema as an input and returns a Patterns Identified matrix (PI), where each row indicates a set of identified patterns corresponding to the set of XSD constructs (block). In principle, the algorithm iterates over global XSD constructions in a sequence while marking the iterated constructs. The first construct (i.e. `schema`) is explicitly marked (since not being handled in any pattern), and for the rest, a loop iterates until no construct in a schema is left unmarked. Within the loop, an unmarked global construct C_i is retrieved and passed to *getIdentifiedPatterns* algorithm, which returns a sub-matrix *PI*. At this point, constructions in the hierarchy of global construct C_i are marked. Lastly,

Algorithm 1: PatternsIdentification - (Extended algorithm from Hacherouf et al. (2019) introduces handling of PI sub-martix)

```
   Input: schemaFS(XS)                                  /* $\mathcal{FS}(\mathcal{XS})$ model of an XSD schema */
   Output: $PI[m][2]$             /* where $m \leq n$ and $n$ is the total no of constructs */
1  $C_i \in C$;                                     /* XSD construct $C_i$ where $1 < i \leq n$ */
2  $Block \subset C$;                         /* proper sub-set of constructions collection */
3  $P \subset G$;                              /* proper sub-set of correspondence patterns */
4  $m = 0$;
5  markConstruction($C_1$);                           /* mark xs:schema construct explicitly */
6  repeat
7      $C_i$ = getConstructionNotMarked();           /* returns $C_i$ as a global construct */
8      $[\{Block, P\}]$ = getIdentifiedPatterns($C_i$);              /* returns $PI$ sub-matrix */
9      foreach $\{Block, P\}$ in $[\{Block, P\}]$ do
10         $PI[m][0] = Block$;
11         $PI[m][1] = P$;
12         $m = m + 1$;
13     end
14 until hasConstructionNotMarked = false;
```

the returned sub-matrix is aggregated in the PI matrix, and the loop is repeated for the next unmarked C_i.

The second algorithm *getIdentifiedPatterns* (Algorithm 2) takes C_i as an input and returns a sub-matrix PI. The global construct C_i is marked and added to a $Block_k = \{C_i\}$ where k is the possible number of blocks accumulated within a global construct. Further, a loop iterates over every construct within a hierarchy of C_i, where the next unmarked

Algorithm 2: getIdentifiedPatterns - (Improved algorithm from Hacherouf et al. (2019) with enhanced identification and intersection sub-algorithm)

```
Input: $C_i \in C$
Output: $[\{Block, P\}]$                                          /* sub-matrix PI */
1  $sG \subset G$;                                        /* proper sub-set of patterns */
2  $sM \subset M$;                            /* proper sub-set of elements in patterns */
3  $P = \{P_0, \ldots, P_k\}$, where $P_j \subset G$;
4  $Block = \{Block_0, \ldots, Block_k\}$, where $Block_j \subset C$;
5  $C_{next} \in C$;
6  $k = 0$;                      /* counts total blocks forming patterns in $C_i$ hierarchy */
7  markConstruction ($C_i$);                             /* mark $C_i$ global construction */
8  $Block_k = \{C_i\}$;          /* initialize constructions block with marked construct */
9  while hasNextConstruction ($C_i$) do
10     $C_{next}$ = getNextConstruction ($C_i$); /* get next unmarked construct in $H_C$ for $C_i$ */
11     markConstruction ($C_{next}$);
12     $Block_k = Block_k \cup \{C_{next}\}$;               /* add marked construct to current block */
13     $sM$ = getElements ($Block_k$);
14     $sG = sM^{\downarrow}$;                  /* find pattern against sM set elements from lattice */
       // $Block_k$ forms no pattern or $C_{next}$ is not in same hierarchy of current block
15     if $sG = \emptyset$ or ($size(Block_k) > 1$ and !IsParentHierarchy ($Block_k$, $C_{next}$)) then
16         $Block_k = Block_k - \{C_{next}\}$;      /* remove the construct from current block */
17         $k = k + 1$;
           // Find intersection of $C_{next}$ with previous blocks resulting a new block
18         $Block_k$ = getIntersection ($Block, C_{next}$);
19     end
20 end
21 for $j \leftarrow 0$ to $k$ do
22     $sM$ = getElements ($Block_j$);                        /* $Block_j$ is set of constructs */
23     $P_j$ = getPatterns ($sM$);         /* $P_j$ set of identified patterns for $sM$ elements */
       // Add $P_j$ of given $Block_j$ in a sub-matrix (collection) for input $C_i$
24     $[\{Block, P\}] = [\{Block, P\}] \cup \{Block_j, P_j\}$;
25 end
```

construct C_{next} is appended to existing $Block_k$ to identify patterns against a set of constructs' elements (sM) in $Block_k$. C_{next} is the unmarked construct connected in hierarchy H_C with C_i, i.e. it can either be a child or child's sibling construct. If no patterns are found against sM set of elements, then C_{next} is removed from $Block_k$ and a possible sequence convergence of constructs is identified that intersects with C_{next} and added as new $Block_{k+1}$. Otherwise, the loop continues for the next unmarked construct. In the last progression of this algorithm, a collection (sub-matrix PI) is produced comprising k records (PI Matrix rows), where each record contains relevant constructions block accompanied with their identified patterns. The details on demonstration of these presented *functional-style syntax* algorithms are presented in next Chapter 6. Since, there is a possibility of more than one pattern identified in this phase, the next phase provides detail on distinguishing to find pertinent patterns.

5.2.2.2 Similar Patterns Filtration

The previous step of the patterns identification process for a certain XSD block may result in a set of similar patterns based on the XSD elements of the constructs. However, they differ based on the XSD attributes and their attribute values (for better understanding, see Table 6.4). Similarly, among 43 transformation patterns, sets of similar patterns are identified and listed in Figure 5.4 which are extracted from the concept lattice shown in Figure 5.3. The PIXCO study highlights the concept lattice is favourable for clearly recognizing these sets of similar patterns. Therefore, from the identified patterns, XSD

Methods	Parameters	Returns	Description
IsParentHierarchy	$sC \subset C$, $C_{next} \in C$	True/False	Check using H_C if C_{next} is within hierarchy of sC.
checkPatternElementsLinked	$A \subset G$, $sE \subset E$	True/False	Checks if set of elements sE are subset of elements contained in set of patterns A. Duplicates in sE are considered single element.
checkConstructHierarchy	$sC \subset C$, $C_{next} \in C$	True/False	Checks in H_C if C_{next} is the connected to any construction in set sC.
getIntersection	$\{sC_0, \ldots, sC_k\} \subset C$, $C_{next} \in C$	$sC_j \subset sC$	Returns the sub-set of constructs sC_j whose elements are linked to C_{next} in concept lattice as well as connected in H_C using *checkPatternElementsLinked* and *checkConstructHierarchy* methods.

Table 5.2: Helper methods for Algorithm 1 and 2 implemented in EPIXCO framework for the improved patterns identification (details in Section 5.3.2).

Methods	Parameters	Returns	Description
getInlineElementType	$c \in C$	*type*	Returns the anonymous or inline *type* definition of construction c.
getTypeOfAttributeValue	$c \in C$, $a \in A$	*type*	Returns *type* of the value of attribute a in construct c by looking into L_T or L_e collection.

Table 5.3: Helper methods implemented in EPIXCO for treatment of similar patterns among Algorithm 3 and 4 (details in Section 5.3.2).

components of the set of constructions have to be scrutinized for pertinent patterns to select the most suitable patterns. Accordingly, two algorithms implemented in PIXCO are enhanced and extended as *PertinentPatterns* and *getPertientPatterns*.

The algorithm *PertinentPatterns* (Algorithm 3) iterates over all the records (rows) of PI matrix and inspects if the identified patterns of any record is identical to any set of constructs listed in Figure 5.4. If yes, then respective set of identified patterns and associated constructions are processed further by *getPertientPatterns*. For example, the different in set 4 = {10, 11, 12} the difference is with the attribute value of `base` attribute of `restriction` construct. On contrary, for set 5 = {27, 28}, the difference is within the

Algorithm 3: PertinentPatterns - (Extended algorithm from Hacherouf et al. (2019) with improved PP matrix by handling duplicate records of identified patterns)

```
Input: $PI[m][2]$                                   /* identified patterns matrix */
Output: $PP[m'][2]$                                 /* pertinent patterns matrix */
// Sets of similar patterns
$P_{sim} = [\{15,16,21,24,25\},\{19,20\},\{6,8,9\},\{10,11,12\},\{27,28\}]$;
// Another set of similar patterns (to be processed differently)
$P_{sim6} = \{17,18,22,23,29,36,37,38\}$;
$Block = \{C_1,\ldots,C_n\}$, where $C_j \in C$ and $Block \subset C$;
$Block' \subset C$;
$P \subset G$;                                       /* set of identified patterns */
$P_{pert} \subset G$;                                /* set of pertinent patterns */
$m' = 0$;                            /* no of rows in $PP$ matrix $\implies m' \geq m$ */
for $i \leftarrow 0$ to $m-1$ do                     /* $m$ rows of the matrix $PI$ */
    $Block = PI[i][0]$;                  /* $i^{th}$ block of constructions in $PI$ */
    $P = PI[i][1]$;                   /* $i^{th}$ set of identified patterns in $PI$ */
    if $P \in P_{sim}$ then
        $P_{pert}$ = getPertinentPatterns$(Block, P)$;
        $PP[m'][0] = Block$;
        $PP[m'][1] = P_{pert}$;        /* update pertinent patterns for $Block$ in $PP$ */
        $m' = m' + 1$;
    else if $P = P_{sim6}$ then
        // Split the $Block$ starting from $3^{rd}$ construction, for each $C_j$
        for $j \leftarrow 3$ to $size(Block)$ do
            $Block' = \{C_1, C_2, C_j\}$;     /* $C_1, C_2$ are first two constructs of $Block$ */
            $P_{pert}$ = getPertinentPatterns$(Block', P)$;
            // $Block'$ possibly contains constructions already processed for $PP$
            if $\{Block', P_{pert}\} \notin PP$ then              /* To avoid duplicate entry */
                $PP[m'][0] = Block'$;
                $PP[m'][1] = P_{pert}$;  /* update pertinent patterns for $Block'$ in $PP$ */
                $m' = m' + 1$;                /* additional rows in $PP$ compared to $PI$ */
            end
        end
    end
end
```

Algorithm 4: getPertinentPatterns - (Improved algorithm from Hacherouf et al. (2019) that explicitly handles similar sets P_{sim1} and P_{sim6})

```
Input: $Block \subset C, P \subset G$
Output: $P_{pert}$
// type is one of XML native Datatype, simpleType or complexType
$P_{pert} = \{P_{pert_1}, \ldots, P_{pert_z}\}$; $Block = \{C_1, \ldots, C_z\}$ where $z \leq 3$, $type$: string;
switch $P$ do
    case $\{15, 16, 21, 24, 25\}$ do
        $type$ = getTypeOfAttributeValue ($C_1$, "type");
        // If no type attribute is defined in $C_1$, find inline type of construction
        if $type = \varnothing$ then $type$ = getInlineElementType ($C_1$);
        if $type$ = Datatype then $P_{pert} = P_{pert} \cup 15$;
        else if $type$ = simpleType then $P_{pert} = P_{pert} \cup 16$;
        else if $type$ = complexType then $P_{pert} = P_{pert} \cup 21$;
        if hasAttribute( "substitutionGroup", $C_1$) then $P_{pert} = P_{pert} \cup 25$;
    case $\{19, 20\}$ do                                /* similar to PIXCO approach */
        ...
    case $\{6, 8, 9\}$ do                               /* similar to PIXCO approach */
        ...
    case $\{10, 11, 12\}$ do                            /* similar to PIXCO approach */
        ...
    case $\{27, 28\}$ do                                /* similar to PIXCO approach */
        ...
    case $\{17, 18, 22, 23, 29, 36, 37, 38\}$ do
        if hasAttribute( "name", $C_3$) then
            $type$ = getTypeOfAttributeValue ($C_3$, "type");
            // Find inline type of construct $C_3$ if no type attribute is defined
            if $type = \varnothing$ then $type$ = getInlineElementType ($C_3$);
            if $type$ = Datatype or simpleType then $P_{pert} = P_{pert} \cup 17$;
            else if $type$ = complexType then $P_{pert} = P_{pert} \cup 22$;
        else if hasAttribute( "ref", $C_3$) then
            $type$ = getTypeOfAttributeValue ($C_3$, "ref");
            if $type$ = Datatype or simpleType then $P_{pert} = P_{pert} \cup 18$;
            else if $type$ = complexType then $P_{pert} = P_{pert} \cup 23$;
        end
        if hasAttribute( "minOccurs", $C_3$) and hasAttribute( "maxOccurs", $C_3$) then
            $x$ = getValueOfAttribute ("minOccurs", $C_3$);
            $y$ = getValueOfAttribute ("maxOccurs", $C_3$);
            if $x = y$ then $P_{pert} = P_{pert} \cup 38$ else $P_{pert} = P_{pert} \cup 36 \cup 37$;
        else if hasAttribute( "minOccurs", $C_3$) then $P_{pert} = P_{pert} \cup 36$;
        else if hasAttribute( "maxOccurs", $C_3$) then $P_{pert} = P_{pert} \cup 37$;
    end
end
```

`attribute` construct whether it comprises `name` or `ref` XSD attribute. Among all 6 sets, *Set 6* is treated distinctively by *PertinentPatterns* and *getPertientPatterns*. In *PertinentPatterns* the given block of constructions is sliced, starting from third element in the set, for distinct procedure on each sub-block (subset of constructs) in *getPertientPatterns*.

The algorithm *getPertientPatterns* (Algorithm 4) selects pertinent patterns from the similar set of patterns. Firstly, it distinguishes the set of identified patterns among six similar patterns to find respective pertinent patterns. Secondly, after differentiating among sets of similar patterns, distinct procedures exploiting helper methods mentioned in Table 5.3 is discretely adapted as each set has a distinct difference. This phase concentrates on details in components of XSD blocks to identify the most appropriate pattern for the block. In this phase, identified patterns in *PI* matrix are processed to filter particular patterns of XSD blocks resulting in *PP* matrix.

5.2.3 Ontology Generation

As the result of previous phase of the EPIXCO framework, the *PP* matrix incorporating two columns; XSD blocks column respective to their produced pertinent pattern(s) column, that is readily available to generate an OWL model. The algorithm *OntologyGeneration* (Algorithm 5) takes *PP* matrix as input. For every XSD block, an OWL block is produced by processing an associated set of the pertinent pattern(s) by correspondence/transformation patterns and further merged into a consolidated OWL model.

Each transformation pattern is associated with an appropriate method (sub-algorithm)

Algorithm 5: OntologyGeneration - (Improved ontology generation algorithm)

```
Input: $PP[m'][2]$                    /* pertinent patterns matrix from algorithm 3 */
Output: owlGraph: RDF, owlFile: File  /* graph in RDF/TTL or RDF/XML file format */
1  $type$: string;
2  $Block \subset C$;
3  $P_{pert} \subset G$;                          /* pertinent patterns against Block */
4  $st\_name$, $base\_extension$: string;
5  $enum\_list$, $name\_list$, $type\_list$: List;
6  for $j \leftarrow 0$ to $m' - 1$ do                    /* $m'$ rows of the matrix PP */
7      $Block = PP[j][0]$;                    /* $j^{th}$ block of constructions in PP */
8      $P_{pert} = PP[j][1]$;          /* $j^{th}$ set of pertinent pattern against Block */
9      foreach $p$ in $P_{pert}$ do   /* $P_{pert}$ may stack multiple patterns (e.g.{17, 37}) */
10         switch $p$ do                      /* handle each pattern $p$:$1 \le p \le 43$ */
11             case 1 do
12                 $st\_name$ = getValueOfAttribute ($C_1$, "name");
                   // Algorithm 6
13                 owlGraph = owlGraph ∪ writePatterns_1 ($st\_name$);
14             end
               ⋮
15             case 6 or 8 or 9 do
16                 $ct\_name$ = getValueOfAttribute ($C_1$, "name");
17                 $base\_extension$ = getValueOfAttribute ($C_3$, "base");
                   // Algorithm 8
18                 owlGraph = owlGraph ∪ writePatterns_6_8_9 ($ct\_name$, $base\_extension$, $p$);
19             end
               ⋮
20             case 17 do
21                 $ct\_name$ = getValueOfAttribute ($C_1$, "name");
22                 for $j \leftarrow 3$ to $size(Block)$ do
23                     $name\_list = name\_list$ ∪ getValueOfAttribute ($C_j$, "name");
24                     $type\_list = type\_list$ ∪ getValueOfAttribute ($C_j$, "type");
25                 end
                   // Algorithm 9
26                 owlGraph = owlGraph ∪ writePatterns_17_18_27_28 ($ct\_name$, $name\_list$,
                    $type\_list$);
27             end
               ⋮
28         end
29     end
30 end
   // create OWL file of consolidated owlGraph in specified file format
31 owlFile = serializeOWLGraph (owlGraph, format);     /* format is RDF/TTL or RDF/XML */
```

Algorithm 6: writePatterns_1

```
  Input: dTN: string                                      /* datatype name */
  Output: OWL Model
1 Declaration ( Datatype ( ag:dTN ) );   /* ag is placeholder prefix for current IRI */
```

Algorithm 7: writePatterns_3

```
  Input: className: string                           /* class/element name */
  Output: OWL Model
1 Declaration ( Class ( ag:className ) );   /* ag is placeholder prefix for current IRI */
```

that has either distinctive representation or can be aligned with other transformation pattern(s) in terms of OWL generation implementation, but requires handling particular edge-cases. The data of the XSD block extracted in $\mathcal{FS}(\mathcal{XS})$ model is passed as parameter to the respective to the patterns' method. For example, *writePatterns_1* (Algorithm 6) requires value of `name` attribute of `simpleType` construct to create `rdfs:Datatype` OWL fragment, whereas *writePatterns_3* (Algorithm 7) takes `name` of `complexType` construct to create `owl:Class` OWL fragment.

Furthermore, *writePatterns_17_18_27_28* (Algorithm 9) shows that multiple patterns can adhere same valid implementation of generating OWL block for the data sent as parameters. However, *writePatterns_6_8_9* (Algorithm 8) collectively implements three transformation patterns but also adds pattern-specific correspondence of parametric elements. Henceforth, for 43 transformation patterns, the framework does not create equal number of writing pattern methods. These algorithms uses the values of `targetNamespace` attribute of `schema` construct in $\mathcal{FS}(\mathcal{XS})$ model to be added and used as namespace for targeted OWL model with its relevant **prefix** (i.e. `ag:<http://example.org/autology#>`). A custom method *multiconcat* encompasses namespace concatenation with the respective

Algorithm 8: writePatterns_6_8_9

```
Input: className: string, dT: string, pattern: int    /* dT is base attribute value */
Output: OWL Model
Declaration ( Class ( ag:className ) );  /* ag is placeholder prefix for current IRI */
Declaration ( DataProperty ( ag:has_className ) );
/* Select XML namespace if dT is one of XML native datatype (e.g.  xs:string)   */
if $dT = nativeDataType$ then $ns = xmlNS$ else $ns = ag$;
if $pattern = 8$ then                          /* extended semantics for pattern 8 */
    Declaration ( DataProperty ( ag:has_dT ) );
    SubDataPropertyOf ( ag:has_className ag:has_dT );
else if $pattern = 9$ then                     /* extended semantics for pattern 9 */
    Declaration ( DataProperty ( ag:has_dT ) );
    SubDataPropertyOf ( ag:has_className ag:has_dT );
    SubClassOf ( ag:className ns:dT );
end
DataPropertyDomain ( ag:has_className ag:className );
DataPropertyRange ( ag:has_className ns:dT );
```

Algorithm 9: writePatterns_17_18_27_28

```
Input: className: string, elemNameList: List, elemTypeList: List
Output: OWL Model
Declaration ( Class ( ag:className ) );
/* Iterates a tuple of eN and eT, where len(elemNameList) = len(elemTypeList)   */
foreach eN, eT in elemNameList, elemTypeList do
    if $eT = nativeDataType$ then $ns = xmlNS$ else $ns = ag$;
    Declaration ( DataProperty ( ag:has_eN_className ) );
    DataPropertyDomain ( ag:has_eN_className ag:className );
    DataPropertyRange ( ag:has_eN_className ns:eT );
end
```

attribute values (as they were in XSD schema) for accurate correspondence and different them with other similar elements.

Since it requires much space, only a few *writePatterns* methods are presented for consideration. Section 6.1 in Chapter 6 provides an example to illustrate the workflow of EPIXCO methodology and implemented algorithms. EPIXCO framework incorporates the RDFLib package that works with RDF to create OWL models. Every fragment of OWL blocks generated from *writePattern* methods is added in an OWL graph for a consolidated OWL model and by using serializers saved in RDF/XML and Turtle syntax. Since formalized XSD components and FCA context patterns are considered sophisticated algorithmic workflow, the framework requires no human-intervention to complete the transformation process.

5.3 Enhancements to Janus and PIXCO Frameworks

The literature of ontology generation unveils the transformation of XML schema to OWL model as a perplexing process (see Section 3.5 in Chapter 3). Nevertheless, following Janus patterns, the PIXCO approach provides a good foundation for defining a formal ontology generation method with defined correspondence rules. Hence, our methodology is inspired by these approaches. However, these approaches are not readily available to be implemented; instead, they provide frameworks and algorithms that further require refinements in terms of correctness and scaling for complex XSD structures and a generalized solution, meaning reforming Janus patterns and refactoring the algorithms of PIXCO. Therefore, in the sections below, the enhancements considered in this research study of the extended framework are provided:

5.3.1 Improving Janus Transformation Patterns

The Janus prototype defines 40 patterns (Bedini et al. 2011, Table I-VI) in 6 groups distribution for maximum XSD components to be transformed into OWL model. However, some highlighted improvements are established to improve their originally defined

Pattern	Janus	EPIXCO

17/18	`ag:has_elt_name rdf:type`
	`owl:ObjectDatatype ;`
	`rdfs:domain ag:ct_name ;`
	`rdfs:range ag:st_name .`
		`ag:has_elt_name_ct_name` `rdf:type`
		`owl:DatatypeProperty ;`
		`rdfs:domain ag:ct_name ;`
		`rdfs:range ag:st_name .`

24	`⟨element name="elt_name"`
	`type="xs:nativeDataType"/⟩`
		`⟨element name="elt_name"`
		`type="ext_type"/``⟩`

32	`ag:has_elt_name1 rdf:type`
	`owl:ObjectProperty ;`
	`rdfs:domain ag:grp_name ;`
	`rdfs:range ag:elt_type .`
	`ag:has_elt_name2 rdf:type`
	`owl:ObjectProperty ;`
	`rdfs:domain ag:grp_name ;`
	`rdfs:range ag:elt_type .`
		`ag:has_elt_name1_grp_name`
		`rdf:type`
		`owl:DatatypeProperty ;`
		`rdfs:domain ag:grp_name ;`
		`rdfs:range` **`ag:elt_st_type`** `.`
		`ag:has_elt_name2_grp_name`
		`rdf:type`
		`owl:ObjectProperty ;`
		`rdfs:domain ag:grp_name ;`
		`rdfs:range` **`ag:elt_ct_type`** `.`

33	`⟨complexType name="ct_name"⟩`
	`⟨`**`sequence`**`⟩`
	`⟨group ref="grp_name"/⟩`
	`⟨`**`/sequence`**`⟩`
	`⟨/complexType⟩`
		`⟨complexType name="ct_name"⟩`
		`⟨group ref="grp_name"/⟩`
		`⟨/complexType⟩`

34	`ag:has_attr_name rdf:type`
	`owl:ObjectProperty ;`
	`rdfs:domain ag:attr_grp_name ;`
	`rdfs:range ag:attr_type .`
		`ag:has_attr_name_attr_grp_name`
		`rdf:type` **`owl:DatatypeProperty ;`**
		`rdfs:domain ag:attr_grp_name ;`
		`rdfs:range ag:attr_type .`

Table 5.4: List of few major improvements identified among XML/OWL syntax of transformation patterns in Janus (Bedini et al. 2011) that are enhanced (as **`bold`**) in EPIXCO.

correspondences that are presented in Table 5.4. First column of the table shows Janus pattern number, second column lists XML/OWL syntax for defined transformation in Janus, and third column are improved considerations in EPIXCO framework.

For example, Janus pattern #17 and #18 corresponds with `owl:ObjectDatatype`, an incorrect OWL axiom. Considering it as typo of `owl:ObjectProperty`, it still mismatches with `element` construct corresponding to `type` or `ref` attribute as `nativeDataType` or `simpleType`. Reason because an `owl:ObjectProperty` denotes `complexType`, and the `owl:DatatypeProperty` denotes `simpleType` representations (see pattern #22, #23,

#32). Hence, EPIXCO associates with `owl:DatatypeProperty`. Similarly, in pattern #33, an XSD block includes `<sequence>` within hierarchy of `<group>` construct. While (Gao et al. 2012) suggests these XSD blocks are defined without `<sequence>` tag.

Furthermore, in details to the improvements mentioned in Table 5.4, the patterns' correspondences are revised for all OWL *Properties* (`ObjectProperty` or `DatatypeProperty`). EPIXCO presents *Properties* with compound names (e.g. '`has_elt_name_ct_name`') by concatenating prefix 'has' with name of associated `element` or `attribute` followed by associated type. The reason for such compound name is to distinguish *Properties* if multiple enclosing `element` or `attribute` in XSD schema shares the same name. If not differentiated, owl graph would add related correspondences of XSD components to same node. This section provided improvements adapted in EPIXCO with Janus transformation patters. Section below provides updates on adapted PIXCO framework.

5.3.2 Refactoring PIXCO Algorithms

PIXCO framework provided formal methodology of XSD to OWL transformation by incorporating patterns specified in informal context model of Janus, and creating a formal mathematical model of input XSD schema to be incorporated in algorithmic processes for ontology generation. However, no prototype of formal transformation is available to implement and verify. Only few methodology steps are presented that also comprises of incorrectness in associations. Moreover, algorithms in PIXCO require enhancements to support complex XML schemas, like ifcXML and CityGML. Therefore, few of the major

Algorithm 10: getIntersection - (Enhanced sub-algorithm to identify correct *Block* forming pattern in EPIXCO framework)

```
   Input: $sC \subset C$, $C_{next} \in C$
   Output: $sC_j \subset sC$
1  $sC = \{sC_0, \ldots, sC_k\}$, where $sC_j \subset C$ and $sC_j = \{C_0, \ldots, C_t\}$;
   // As parent construct to $C_{next}$ maybe present in constructs sub-sets added last.
2  $k = size(sC)$;
3  for $j \leftarrow k-1$ to $0$ do                       /* Iterate $sC$ in reverse order */
4      if checkConstructHierarchy ($sC_j$, $C_{next}$) then
5          break;      /* identified $sC_j$ is sub-set of constructs with $C_{next}$ in $H_C$ */
6      end
7  end
8  $Block = \{C_{next}\}$;
   // Split the $sC_j$ for each construction $C_t \in C$ to identify possible pattern.
9  for $t \leftarrow 0$ to $size(sC_j)$ do
10     $Block = C_t \cup Block$;         /* add construct from set $sC_j$ to current block */
11     $sM$ = getElements ($Block$);
12     $sG = sM^{\downarrow}$;            /* find pattern against $sM$ set elements from lattice */
       // Check if $sM$ elements from $Block$ forms pattern or linked in lattice.
13     if $sG = \emptyset$ or !checkPatternElementsLinked ($sG$, $sM$) then
14         $Block = C_t - Block$;
15         break;
16     end
17 end
   // $Block$ of constructs in $H_C$ forming patterns and intersecting with $C_{next}$.
18 $sC_j = Block$;
```

algorithmic improvements are highlighted below:

1. In Algorithm 2, if no pattern(s) are found against $Block_k$ constructions including C_{next} construct, identification of a potential pattern in *getIntersection* dynamically with constructs of current block $Block_k$ becomes challenging. As C_{next} might not be within same hierarchy as of defined patterns. Hence, EPIXCO implements *getIntersection* (Algorithm 10) functionality to enhance and refine pattern identification process with supporting methods like *checkPatternElementsLinked* and *checkConstructHierarchy*.

Algorithm 11: getTypeOfAttributeValue - (Enhanced sub-algorithm for retrieval of correct *type* for a given attribute in EPIXCO framework)

```
Input: $c \in C$, $a \in A$                 /* attribute $a$ in construct $c$ */
Output: type: $Type$        /* native XML datatype, simpleType or complexType */
1  $v$ = getValueOfAttribute ($a$, $c$);        /* returns value of attribute $a$:$v \in V_A$ */
2  if $v$ = $nativeDataType$ then     /* XML native type i.e. {xs:string, xs:int,...} */
3      type = Datatype;
4  else
5      if $v \in L_T$ then                   /* $L_T$ is type tuple of $\mathcal{FS}(\mathcal{XS})$ model */
6          $t = L_T[v]$;                 /* type $t$ against $v$ in $L_T$ of $\mathcal{FS}(\mathcal{XS})$ model */
7          if $t$ = simpleType then type = simpleType else type = complexType ;
8      else if $v \in L_e$ then              /* $L_e$ is element tuple of $\mathcal{FS}(\mathcal{XS})$ model */
9          $c = L_e[v]$;              /* get new construct $c$ against element $v$ in $L_e$ */
           // Recursively identify attribute $a$ for referenced construct $c$ in value $v$
10         type = getTypeOfAttributeValue ($c$, $a$);
           // Check for inline construct if no type is found for attribute $a$
11         if type = $\varnothing$ then type = getInlineElementType ($c$);
12     else
13         type = $\varnothing$;
14     end
15 end
```

2. Since, design of XSD maybe complex, retrieving *getTypeOfAttributeValue* isn't straight forward as key value pair. In EPIXCO, functionality of *getTypeOfAttributeValue* in Table 5.3 is revised as Algorithm 11, providing a recursive solution to return *type* of referenced attribute a for given construction c. The method assists with more generalized implementation for XSD design with complex (nested or referenced) associations.

3. The *writePatterns* method in ontology generation phase of the framework provides a fragment of OWL graph. The fragments are customized for patterns sharing correspondence rules and namespaces. For example, in Algorithm 8 dT parameter is checked to assign the correct namespace prefix accordingly which provides maximum

correctness in transformed models.

4. EPIXCO treats the pattern 25 in Algorithm 4 separately as an `element` construct may contains attributes of `type` and `substitutionGroup` simultaneously.

5. EPIXCO includes identification of pattern 15 for *nativeDatatypes* in Algorithm 4, first case of identified patterns set $\{15, 16, 21, 24, 25\}$, that is not considered in PIXCO approach.

6. Similarly, EPIXCO implements *getInlineElementType* method referenced in Algorithm 4 for similar patterns identification of P_{sim_1} and P_{sim6}. The enhanced method is used for constructions with no `type` attribute to find appropriate `type` values.

Mentioned above are a few from the list of improvements, minor or logically complex, for the enhancements of extended algorithms implemented in EPIXCO, based on PIXCO suggested algorithms. Possibilities in which the XML schemas are incomplete or not fully validated exists. Henceforth, the EPIXCO algorithms are designed to handle edge cases like *null* or empty if attributes or attribute values are not available in extracted XSD information model $\mathcal{FS}(\mathcal{XS})$.

5.3.3 Reconstructed Mapping Rules of XSD to OWL

The mapping rules are defined as what an XSD component will correspond to in an OWL syntax (Table 3.4 highlights example mappings). However, improved correspondence rules for XSD schema elements (`element, attribute,`...) are applied in EPIXCO, presented in Figure 5.5, where a global construct of `element` is mapped to an OWL `Class` or `Datatype` depending on attribute value of `type` and `ref` attributes. Otherwise, a local `element` declaration is mapped as `ObjectProperty` or `DatatypeProperty`, based on same attribute value rules. Whereas for `element` with anonymous (inline) type declaration, local or global, mapping is with `ObjectProperty` or `DatatypeProperty` as well as `Class` or `Datatype`. This is an example of complexity that is not very well established within prior mapping approaches and needs to be addressed. A similar subset of the rule is

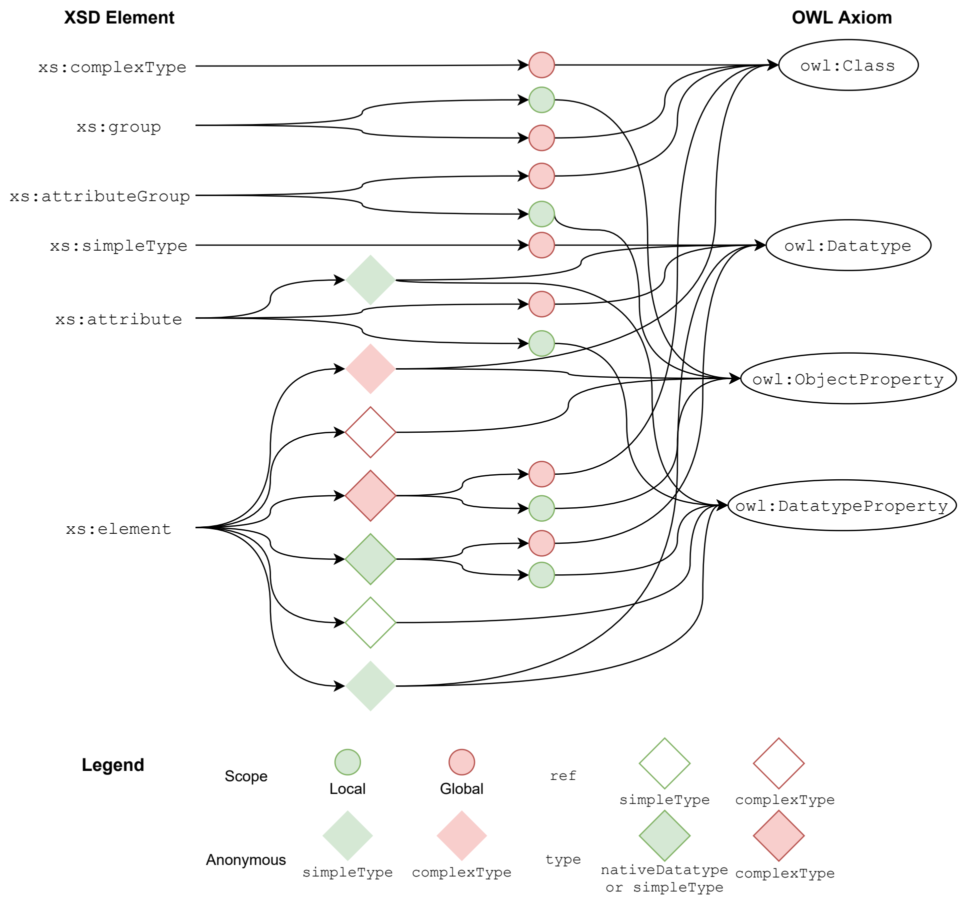


Figure 5.5: Reconstructed and enhanced correspondence rules of mapping XSD to OWL for an improved transformation design in EPIXCO.

applied with the `attribute` tag, and Figure 5.5 presents pictorial representation of all correspondence rules for better understanding.

The mapping process is more complicated than defining simple correspondence rules. Janus and PIXCO handles small segment of these mapping OWL representations, where as, EPIXCO design establishes better mapping rules (as shown in Figure 5.5) to assist in implementing algorithms and helper methods of Table 5.2 and 5.3. These methods also apply recursive retrieval to the attribute values of `type` and `ref` attributes as referring to `complexType` or `simpleType` constructions. Further, EPIXCO implements *getInlineElementType* function to explicitly identify anonymous (inline) type declarations of XSD constructs where `type` and `ref` are not present. The enhanced mapping rules transforms respective OWL models with substantial and improved ontology mappings. These mapping rules are further used in EPIXCO implementation (Chapter 6) to be defined as ground truth of OWL axioms identified for a given XSD schema.

5.3.4 General Considerations in Enhanced Framework

For understanding correct XSD construct correspondence to an OWL syntax, EPIXCO considers representation of classes (entities) and individuals linked with other ontology axioms (properties, operators etc.) from standards like Ontology Visualization Benchmark (OntoViBe) (Haag et al. 2015) and ifcOWL (Pauwels and Terkaj 2016) as standardized representation of OWL axioms and implements them accordingly. For example, in Janus patterns $\#36 - 38$, the occurrence constraints is represented with an `owl:equivalentClass` axiom in OWL namespace, while, OntoVibe and ifcOWL standard ontologies employ `rdfs:subClassOf` axiom in RDFS namespace.

The transformation of XSD to OWL is a complicated process. While patterns with defined correspondences help to identify the correct representations, it is noted that different XSD designs like Russian Doll, Salami Slice etc. (see Section 3.3) provides different arrangement styles of the same XSD components. Nonetheless, the implementation of this methodology is an iterative process resulting in an evolving ontology model. Thus, for a generic procedure and maximum transformation, some cases require special attention towards algorithmic considerations of XSD to OWL transformation for a generalized behaviour. Correspondingly, most of these considerations are employed in EPIXCO for more accurate ontology modelling by following addressed improvements in the transformation process motivated by Janus patterns and algorithmic implementation of PIXCO.

5.4 Summary

As mentioned at the beginning of this Chapter, a primary methodology is established for the broad research to adapt semantic web technologies for integrating BIM and GIS with sensors data. A focused framework to obtain seamless information exchange between sensors, building and geospatial information. A comprehensive conceptual framework (presented in Section 5.1) is developed to resolve integration problem. However, due to the complexity of the very first module, the approach for this thesis is revised with only focus on the fundamental of a preparatory seamless integration framework, i.e. towards

	PIXCO	**EPIXCO**
Formal Method	✓	✓
Extensible	✓	✓
Automatic Transformation	✓	✓
Enhanced Mapping Rules		✓
Generalized Framework for XSD Designs		✓
OWL Expressivity	n/a	ALN(D)
OWL Models Validation		✓
Results Performance Measures	✓	✓
Results Statistical Analysis		✓

Table 5.5: A summary of enhanced considerations and improvements as contribution in EPIXCO methodology compared with PIXCO Hacherouf et al. (2019) approach.

automatic ontology generation. Therefore, this study defines the ontology generation methodology for the launch version of broad research methodology.

An automatic ontology development method is defined in Section 5.2 of this Chapter that is proposed as a evolving framework. The quality of ontology relies on the accurate modelling of its constituents; concepts, properties, facets and individual instances. Any mismatch or a unique understanding of an axiom for a mapping pattern can lead to a completely different ontological representation. Therefore, the proposed methodology adapts an enhanced correspondence of rules defined in Section 5.3.3 for accurate modelling.

A formally enhanced EPIXCO transformation method is designed in Section 5.2 to extend and integrate transformation patterns from Janus and PIXCO incrementally for the automatic ontology generation process. Table 5.5 presents summary improvements in comparison of established EXPICO methodology with PIXCO approach. EPIXCO simultaneously considers different XSD structural design styles to produce OWL models in OWL 2-RL profile. Henceforth, these enrich generated conceptual models are flexible to be aligned with information from heterogeneous sources establishing integrated systems.

Chapter 6

EPIXCO Implementation: Building Ontologies Automatically

This Chapter aims to introduce EPIXCO, a system developed to provide a solution with the capability to automatically generate ontology to solve the fundamental problem towards building integration systems. The EPIXCO prototype accepts XSD schema and applies algorithms to generate an OWL file that can further facilitate the ontology alignments and integrated information systems.

The Chapter is outlined as follows: Section 6.1 introduces the prototype and provides the overall implementation system of EPIXCO framework with an example; Section 6.2 provides details on the workflow of extracting XSD information from an XML document and establishing the ground truth of given XSD schema; Section 6.3 presents the experimentation with different source XSD to evaluate this work, among them includes system scalability, performance, accuracy and comparative analysis with standard bodies to measure quality; Section 6.4 discusses the framework, in-depth difficulties of its implementation with considerations made to overcome them and its limitations; and the final Section 6.5 provides overall analysis and concludes this Chapter.

6.1 Implementation Features

In trying to answer the research question of "automatic ontology generation process" the prototype of proposed transformation methodology named **EPIXCO** (**E**nhanced **P**atterns **I**dentification for **X**SD **C**onversion to **O**WL) is implemented, as an extended

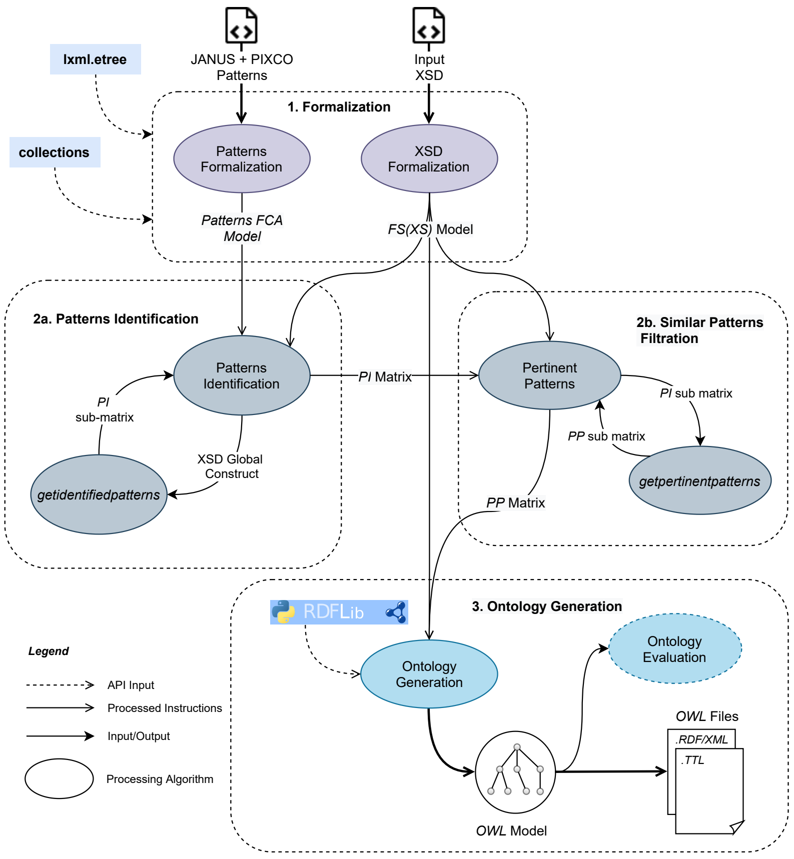


Figure 6.1: Overall architecture of EPIXCO.

framework of PIXCO (Hacherouf et al. 2019). The prototype extracts detailed information from XSD schema, and recognizing a defined set of patterns across input schema, without intervention, results in an OWL model in an output RDF/OWL file. The EPIXCO framework is developed by understanding the JANUS and PIXCO methodologies and their algorithms, as no source code was available for given methodologies. The overall architecture of the system is presented in Figure 6.1 and consists of three major phases with algorithms utilizing APIs in a Python-based environment elaborated below in detail.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="http://example.
    org/autology#">
  <xs:simpleType name="DegreeEnum">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Masters"/>
      <xs:enumeration value="Doctoral"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="Document">
    <xs:sequence>
      <xs:element name="Title" type="xs:string" maxOccurs="1"/>
      <xs:element name="Author" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="Thesis">
    <xs:complexContent>
      <xs:extension base="Document">
        <xs:attribute name="Degree" type="DegreeEnum"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:element name="Thesis" type="Thesis"/>
</xs:schema>
```

Listing 6.1: An example XSD schema inspired by example in PIXCO (Hacherouf et al. 2019) reused for better relative illustration with EPIXCO framework.

Formal modelling, in the first phase of EPIXCO framework, parsing and manipulate the XSD documents and all defined transformation patterns are sourced into a formal context model uses *lxml.etree*[13] library. The modelling process iterates over the tree structure of the XSD document from a root node to extract all attributes and attribute values of each node to create a mathematical model $\mathcal{FS}(\mathcal{XS})$. As an example, Listing 6.1 is presented to illustrate the formal model where the line numbers of example schema correspond to each XSD construction. The lines with closing tags are excluded from modelling i.e. [6, 7, 12, 13, 18, 19, 20, 22], while the remaining lines [1, 2, 3, 4, 5, 8, 9, 10,

[13]https://docs.python.org/3/library/xml.etree.elementtree.html

Tuple	**Value**
$E =$	{`xs:schema`, `xs:simpleType`, `xs:restriction`, `xs:enumeration`, `xs:complexType`, `xs:sequence`, `xs:element`, `xs:complexContent`, `xs:extension`, `xs:attribute`}
$A =$	{`xmlns:xs`, `targetNamespace`, `name`, `base`, `value`, `type`, `maxOccurs`}
$V_A =$	{`"http://www.w3.org/2001/XMLSchema"`, `"http://example.org/autology#"`, `"DegreeEnum"`, `"xs:string"`, `"Masters"`, `"Doctoral"`, `"Document"`, `"Title"`, `"1"`, `"Author"`, `"Thesis"`, `"Degree"`}
$C =$	{C_1, C_2, C_3, C_4, C_5, C_8, C_9, C_{10}, C_{11}, C_{14}, C_{15}, C_{16}, C_{17}, C_{21}}
	$C_1 = \langle$`xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="http://example.org/autology#"`$\rangle$
	$C_2 = \langle$`xs:simpleType name="DegreeEnum"`$\rangle$
	$C_3 = \langle$`xs:restriction base="xs:string"`$\rangle$
	$C_4 = \langle$`xs:enumeration value="Masters"`$\rangle$
	$C_5 = \langle$`xs:enumeration value="Doctoral"`$\rangle$
	$C_8 = \langle$`xs:complexType name="Document"`$\rangle$
	$C_9 = \langle$`xs:sequence`$\rangle$
	$C_{10} = \langle$`xs:element name="Title" type="xs:string" maxOccurs="1"`$\rangle$
	$C_{11} = \langle$`xs:element name="Author" type="xs:string"`$\rangle$
	$C_{14} = \langle$`xs:complexType name="Thesis"`$\rangle$
	$C_{15} = \langle$`xs:complexContent`$\rangle$
	$C_{16} = \langle$`xs:extension base="Document"`$\rangle$
	$C_{17} = \langle$`xs:attribute name="Degree" type="DegreeEnum"`$\rangle$
	$C_{21} = \langle$`xs:element name="Thesis" type="Thesis"`$\rangle$
$H_C =$	$\{(C_1,C_2), (C_2,C_3), (C_3,C_4), (C_3,C_5), (C_1,C_8), (C_8,C_9), (C_9,C_{10}), (C_9,C_{11}), (C_1,C_{14}), (C_{14},C_{15}), (C_{15},C_{16}), (C_{16},C_{17}), (C_1,C_{21})\}$
$L_T =$	{{`simpleType`: {`"DegreeEnum"`}}, { `complexType`: {`"Document"`, `"Thesis"`}}}
$L_e =$	{{`"Title"`: C_{10}}, {`"Author"`: C_{11}}, {`"Thesis"`: C_{21}}}

Table 6.1: The $\mathcal{FS}(\mathcal{XS})$ model defined for the information extracted from XSD schema in Listing 6.1.

11, 14, 15, 16, 17, 21] are considered. The line numbers are treated as indexes of XSD constructions, meaning each XSD construct can be referred to as $C_{linenumber}$; for example, C_1 refers to the construct of line number 1. Therefore, the formalization process of the Listing 6.1 produces set of 7-tuples expressed in Table 6.1 as $\mathcal{FS}(\mathcal{XS})$ model. Unlike PIXCO architecture, the $\mathcal{FS}(\mathcal{XS})$ model is not stored in any database and is part of the complete workflow which eventually saves an OWL model upon completion.

The second phase with of patterns identification uses Formal Concept Analysis (FCA) to create a concept lattice using *collections*[14] library on 43 Janus and PIXCO transformation patterns. EPIXCO prototype implements *PatternsIdentification* (Algorithm 1) and *getIdentifiedPatterns* (Algorithm 2) for the pattern identification processes which exploits helper methods such as *checkPatternElementsLinked*, *getIntersection* (listed in Table 5.2). To demonstrate the basic workflow of these two algorithms, Table 6.1 formalized with $\mathcal{FS}(\mathcal{XS})$ model is used. The formal model is sent as parameter to Algorithm 1 and subsequently, the Table 6.2 displays the principle state of designated algorithmic variables like C_i, XSD blocks and set of identified patterns P. The **bold** cells show determined state by algorithm for the block and upon completion the resulting PI matrix is shown in Table 6.3a, where first column presents XSD block with set of constructs respective to its identified patterns.

Further, the framework implements *PertinentPatterns* (Algorithm 3) and *getPertientPatterns* (Algorithm 4) filtration for pertinent patterns by using helper methods listed in

[14]https://docs.python.org/3/library/collections.html

C_i	***Block**$_k$*	**Block XSD Elements**	**States of *P***
C_2	{2}	–	–
	{2, 3}	{`simpleType`, `restriction`}	{4, 5, 7}
	{2, 3, 4}	{`simpleType`, `restriction`, `enumeration`}	{4}
	{2, 3, 4, 5}		**{4}**
C_8	{8}	–	–
	{8, 9}	{`complexType`, `sequence`}	{17, 18, 22, 23, 29, 36, 37, 38}
	{8, 9, 10}	{`complexType`, `sequence`, `element`}	{17, 18, 22, 23, 29, 36, 37, 38}
	{8, 9, 10, 11}		**{17, 18, 22, 23, 29, 36, 37, 38}**
C_{14}	{14}	–	–
	{14, 15}	{`complexType`, `complexContent`}	{13, 14}
	{14, 15, 16}	{`complexType`, `complexContent`, `extension`}	{13}
	{14, 15, 16, 17}	{`complexType`, `complexContent`, `extension`, `attribute`}	{}
	{14, 15, 16}	{`complexType`, `complexContent`, `extension`}	**{13}**
	{14, 17}	{`complexType`, `attribute`}	**{27, 28}**
C_{21}	**{21}**	element	**{15, 16, 21, 24, 25}**

Table 6.2: The iterative states of variables over execution of Algorithm 1 and 2 on the input $\mathcal{FS}(\mathcal{XS})$ model specified in Table 6.1.

Block	***P***
{2, 3, 4, 5}	{4}
{8, 9, 10, 11}	{17, 18, 22, 23, 29, 36, 37, 38}
{14, 15, 16}	{13}
{14, 17}	{27, 28}
{21}	{15, 16, 21, 24, 25}

(a)

Block	***P***
{2, 3, 4, 5}	{4}
{8, 9, 10}	{17, 37}
{8, 9, 11}	{17}
{14, 15, 16}	{13}
{14, 17}	{27}
{21}	{21}

(b)

Table 6.3: The resulting matrix of (a) *PI* of Listing 6.1 by identification algorithm (Algorithm 1) (b) *PP* of given *PI* Matrix by pertinent patterns (Algorithm 3).

Table 5.3. It can be noted from Table 6.3a that for a given block, multiple patterns can be identified, e.g. for block {8, 9, 10, 11} set of {17, 18, 22, 23, 29, 36, 37, 38} patterns are identified. In such cases, XSD components of the block are further scrutinized to

Pattern	XSD Constructions
27	⟨`xs:complexType name="ct_name"`⟩
	⟨`xs:attribute name="attr_name" type="st_name"`⟩
28	⟨`xs:complexType name="ct_name"`⟩
	⟨`xs:attribute ref="attr_name"`⟩

Table 6.4: XSD constructions based different between similar patterns **set 2** referenced from Figure 5.4.

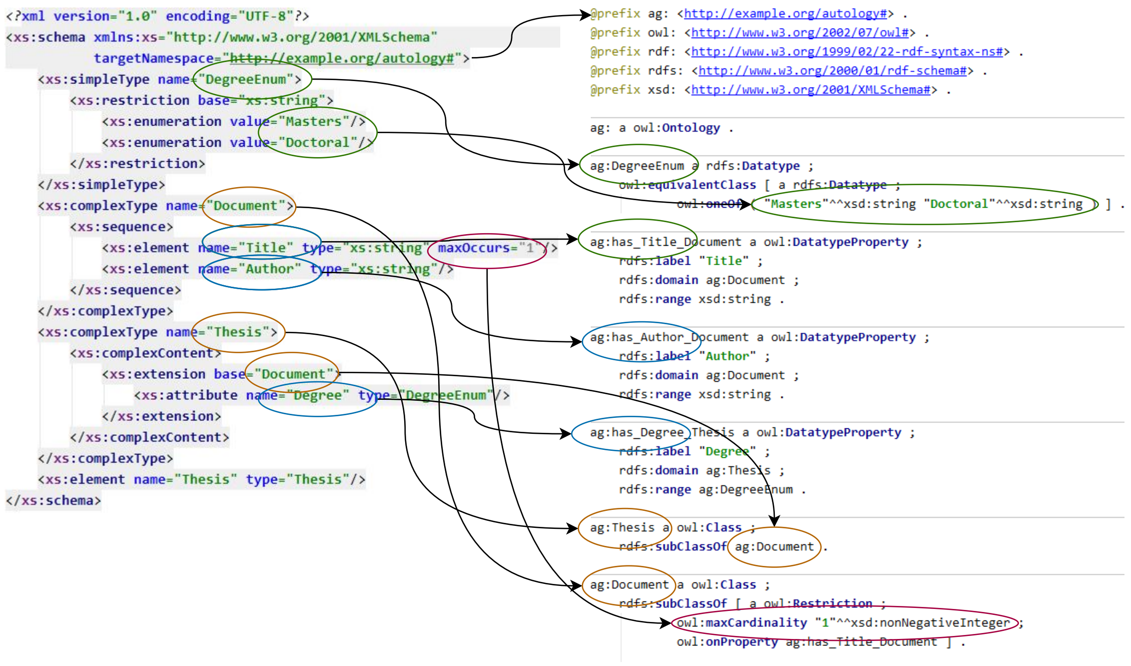
distinguish and identify pertinent pattern. While Section 5.2.2.2 provides detail on such distinguishing process to find pertinent pattern, relative example of these similar patterns is illustrated in Table 6.4 where the constructions of set of patterns {27, 28} comprises same XSD elements {`complexType`, `attribute`}. The difference occurs in the construct with `attribute` element, for which pattern 27 has attribute `name` in contrast to pattern 28 with the attribute `ref`.

In another example of identifying specific pattern, from the *PI* matrix in Table 6.3a, the block {8, 9, 10, 11} identifies P_6 or Set 6 = {17, 18, 22, 23, 29, 36, 37, 38} is sliced based on the number of `element` constructs (see Algorithm 3). Afterwards, each sub-block with `element` construction is distinguished for pattern 17, 18, 22 or 23 by either comprising `name` or `ref` XSD attribute or explicitly with the *type* of their attribute values (see Algorithm 4). After these pertinent patterns, the same `element` construct is examined for `minOccurs` or `maxOccurs` attribute and appends respective patterns as one of: 36, 37, (36 and 37) or only 38 (see last section in Algorithm 4). Henceforth, the Table 6.3b shows the pertinent pattern filtration results of the identified patterns in Table 6.3a.

In the last phase of prototype, EPIXCO implements Algorithm 5 exploiting APIs from

RDFLib[15] package with OWL-API to create consolidated OWL graph for extracted XSD information of a given XSD block to generate an OWL block by using pertinent patterns *PP* matrix in Table 6.3b. Each record in *PP* is processed by *OntologyGeneration* algorithm with tailored *writePatterns* methods where an XSD block against carefully identified patterns is mapped to an OWL syntax. Figure 6.2 illustrates the principle mapping of XSD block with OWL by established correspondence rules in Figure 5.5. For elaboration, segments of ontology model shows OWL fragment of respective XSD blocks. These OWL fragments of XSD blocks passed to sub-functions are synthesized with sub-functions such

[15]https://rdflib.readthedocs.io



```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://example.org/autology#">
    <xs:simpleType name="DegreeEnum">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Masters"/>
            <xs:enumeration value="Doctoral"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="Document">
        <xs:sequence>
            <xs:element name="Title" type="xs:string" maxOccurs="1"/>
            <xs:element name="Author" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="Thesis">
        <xs:complexContent>
            <xs:extension base="Document">
                <xs:attribute name="Degree" type="DegreeEnum"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:element name="Thesis" type="Thesis"/>
</xs:schema>
```

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

ag: a owl:Ontology .

ag:DegreeEnum a rdfs:Datatype ;
    owl:equivalentClass [ a rdfs:Datatype ;
            owl:oneOf ( "Masters"^^xsd:string "Doctoral"^^xsd:string ) ] .

ag:has_Title_Document a owl:DatatypeProperty ;
    rdfs:label "Title" ;
    rdfs:domain ag:Document ;
    rdfs:range xsd:string .

ag:has_Author_Document a owl:DatatypeProperty ;
    rdfs:label "Author" ;
    rdfs:domain ag:Document ;
    rdfs:range xsd:string .

ag:has_Degree_Thesis a owl:DatatypeProperty ;
    rdfs:label "Degree" ;
    rdfs:domain ag:Thesis ;
    rdfs:range ag:DegreeEnum .

ag:Thesis a owl:Class ;
    rdfs:subClassOf ag:Document .

ag:Document a owl:Class ;
    rdfs:subClassOf [ a owl:Restriction ;
            owl:maxCardinality "1"^^xsd:nonNegativeInteger ;
            owl:onProperty ag:has_Title_Document ] .
```

Figure 6.2: Principle mapping visualization of procedurally generated OWL blocks in TTL format (right) corresponding to an XSD blocks (left) of Listing 6.1 by EXPICO prototype.

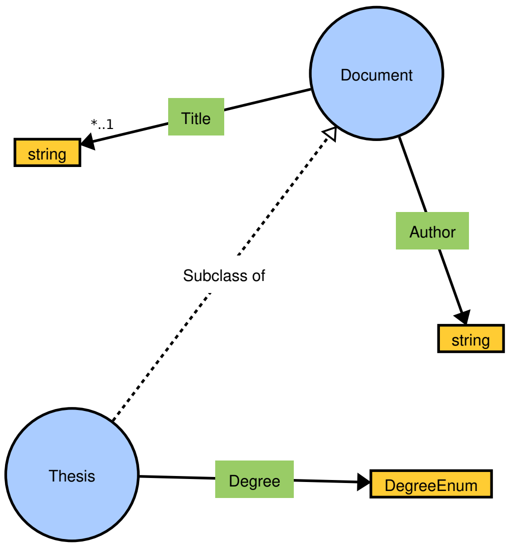


Figure 6.3: A WebVOWL visualization of ontology model generated for Listing 6.1 using EPIXCO architecture.

as Algorithm 6, 7, 8, 9, etc, and results an RDF/OWL file in RDF/XML and RDF/TTL formats containing resulting ontology axioms (`Class`, `ObjectProperty`, . . .). RDFLib also support more graph parsers and serializers[16] like *n3* and *nquads*. Figure 6.3 shows the visualization of OWL model generated from the Listing 6.1 using EPIXCO prototype.

As seen from the example presented in Figure 6.2, the EPIXCO methodology provides detailed semantic correspondence. It presents ontological vocabularies of existing schema information in OWL axioms like classes and properties. The XSD components of patterns are considered during the transformation process, and the framework requires no user-intervention to complete the transformation.

[16]https://rdflib.readthedocs.io/en/stable/plugin_serializers.html

6.2 Experiments Preparations

Before conducting experiments on the EPIXCO prototype framework, *eight* XSD schemas were established to be used as testing datasets. For this purpose, the six XSD schema of ifcXML and CityGML formats are used along with an example XSD Listing 6.1 and a schema generated for a GeoBIM benchmark *IFCGeometries* data (see Section 4.2.1). The following sections provide the process of establishing ground truth for these are schemas and extracting XSD schema from an EXPRESS-based .ifc file.

6.2.1 XSD Extraction from IFC EXPRESS File

The IFC files created in academia and industry environment are mainly based on EXPRESS schema. However, a procedure can be defined to create ifcXML from an IFC Step file and further using XML to XSD conversion tools to create the respective XSD schema. For this, the IfcConvert tool from IfcOpenShell library[17] is applied to convert the IFC Step file into ifcXML. *IfcConvert.exe* command-line tool generates an XML file for a given .ifc file. It is to note that IfcOpenShell only supports 'IFC2x3' schema. Following this, online tools like XML to XSD converter[18] and XSD/XML Schema Generator[19] can be used to generate XSD of XML document, which also provides options between XSD designs (see Section 3.3 for XSD designs). The generated XSD file can be validated with

[17]http://ifcopenshell.org/

[18]https://www.liquid-technologies.com/online-xml-to-xsd-converter

[19]https://www.freeformatter.com/xsd-generator.html

respective XML Schema Validator[20] tool.

6.2.2 Ground Truth Matrix

As mentioned earlier, one of the problems encountered in establishing the ontology generation process was the lack of tools available to evaluate the potential semantics generated for a given XML schema. In this formal ontology generation process, a detailed correspondence of defined rules is exercised over selected XSD files, and Table 6.5 presents their generated respective OWL axioms as ground truth using correspondence rules defined in Figure 5.5. These ground truth values presented are scrutinized versions of possible duplicates of axioms that may share the same `name` among the same XSD components. Since EPIXCO, at this stage, implements **30 of 43** transformation patterns, the constructs not being considered in the generated ontology are filtered from ground truth to create filtered ground truth, shown in Table 6.6. These calculated ground truth values using Equation 6.1 are further used in Section 6.3 to calculate the accuracy of the EPIXCO

[20]https://extendsclass.com/xml-schema-validator.html

Schema	`Class`	`Datatype`	`DatatypeProperty`	`ObjectProperty`
Example	2	1	3	-
cityGMLBase	17	12	13	13
building	47	20	24	47
bridge	50	21	24	45
IFCGeometries	21	6	12	20
IFC2x3 TC1	1388	279	649	643
IFC4_ADD1	1147	341	527	663
IFC4 ADD2 TC1	1161	342	529	676

Table 6.5: Ground Truth of input XML schema specifications (listed in Table 3.2) with unique OWL axioms count calculated using rules in Figure 5.5.

ontology generation process, which can be presented as:

$$Accuracy\,(a) = \frac{Correct\ Samples\,(a)}{All\ Samples\,(a)} \tag{6.1}$$

where a is an *axiom* in OWL model (i.e. `class, datatype or datatypeProperty`), *Correct Samples* are unique axioms of type a and *All Samples* provides ground truth axioms of type a.

Schema	**`Class`**	**`Datatype`**	**`DatatypeProperty`**	**`ObjectProperty`**
Example	2	1	3	-
cityGMLBase	16	12	11	13
building	47	20	24	47
bridge	50	21	24	45
IFCGeometries	21	6	12	20
IFC2x3 TC1	1388	164	558	520
IFC4_ADD1	1087	211	527	482
IFC4 ADD2 TC1	1100	212	529	493

Table 6.6: Filtered Ground Truth of input XSD schemas with unique OWL axioms count of currently implemented 30 transformation patterns.

6.3 Experimental Results

To validate this research, experiments are executed on *eight* XML schemas, that are derived by IFC schema specifications[21], CityGML 2.0[22], an IFC benchmark dataset (see Section 4.2.1) and an example schema. Details on extracted XSD components of these the documents and their schema specifications are presented in Section 3.4.

Since there are no readily tools available for XSD to OWL generation, there is a

[21]https://technical.buildingsmart.org/standards/ifc/ifc-schema-specifications/

[22]https://www.citygmlwiki.org/index.php/CityGML_XML_Schema

lack of referent ontology for these schemas; motivation towards building ground truth in this methodology to produce a correct expected result. The first schema *Example* in Listing 6.1 is a simple illustration XSD schema with limited in terms of dimensions of XSD components for a more humanly accessible execution of process and correspondence with resultant model. The remaining schema documents of IFC and CityGML are for better analysis and overall observation of the scalability process.

6.3.1 Validation of OWL Models

In this study, the ontology models generated from experiments performed on selected schemas are validated using W3C RDF validator Service[23] and OWL Validator[24]. The models are also evaluated using OOPS! (OntOlogy Pitfall Scanner!) (Poveda-Villalón et al. 2014) provides pitfalls in three levels: critical, important and minor, where only minor and important issues are identified. Furthermore, Protégé (Musen 2015) tool is also used for validation assessment and deriving ontology metrics of generated models as well as another web-based WebVOWL[25] tool for visualization of valid ontology graphs with capability of handling large ontology graphs (see Figure 6.4). These OWL models are further evaluated using quantitative and comparative analysis in our study, presented following sections.

[23]https://www.w3.org/RDF/Validator/

[24]http://visualdataweb.de/validator/

[25]http://www.visualdataweb.de/webvowl/

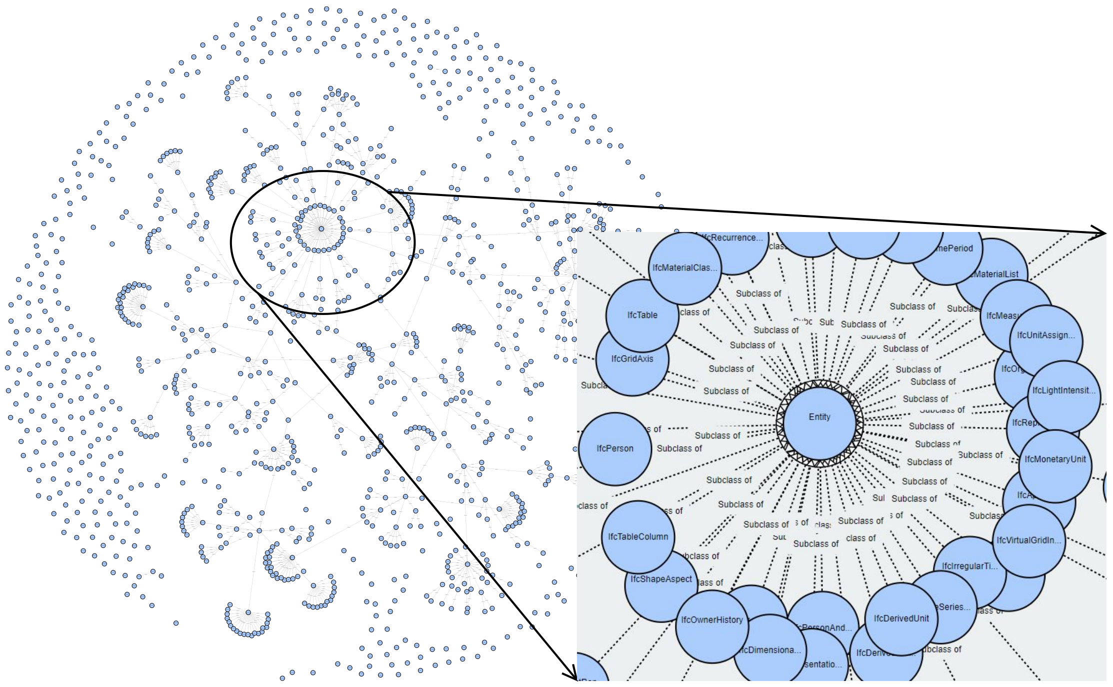


Figure 6.4: WebVOWL visualization of only `Class` axioms of ontology automatically generated for the XSD schema 'IFC4 ADD2 TC1' using the EPIXCO framework, right image shows top right cluster of classes of left image.

6.3.2 Quantitative Evaluation

To provide the statistical details of generated owl models of selected schemas, ground truth presented in Table 6.5 for their OWL axioms is evaluated with resulting axioms of generated OWL models by EPIXCO, presented in Table 6.7. It applies similar filtration applied for the duplicate occurrence of axioms to establish correct axioms count. The accuracy of selected axioms generated in OWL models by EPIXCO methodology is calculated using Equation 6.1 and presented in Table 6.8. The accuracy of IFC schemas (except *IFCGeometries*) at this point for `Datatype` and `ObjectProperty` are below 0.65 on average, with 1 being highest. The reason being not all XSD constructs participated

Schema	`Class`	`Datatype`	`DatatypeProperty`	`ObjectProperty`
Example	2	1	3	-
cityGMLBase	16	12	11	13
building	47	20	24	47
bridge	50	21	24	45
IFCGeometries	21	6	12	20
IFC2x3 TC1	1190	164	558	520
IFC4_ADD1	1087	211	526	482
IFC4 ADD2 TC1	1100	212	528	493

Table 6.7: OWL axioms generated by proposed method for the OWL models of selected XML schemas.

in determining the ground truth are transformed. Since the current implementation of the EPIXCO prototype implements 30 transformation patterns, and if the components of input XSD constructs are not in respective 30 patterns, then filtered ground truth is considered to reevaluate the accuracy of axioms, the accuracy results are highly improved. They are presented in Table 6.9 illustrating the completeness factor of the framework that the considered patterns provide enhanced transformation of axioms. Hence, the statistical results of accuracy show that OWL models generated from EPIXCO constitute the maximum number of axioms for the XSD constructions with implemented transformation patterns.

Schema	`Class`	`Datatype`	`DatatypeProperty`	`ObjectProperty`
Example	1	1	1	-
cityGMLBase	0.94	1	0.85	1
building	1	1	1	1
bridge	1	1	1	1
IFCGeometries	1	1	1	1
IFC2x3 TC1	0.86	0.59	0.85	0.76
IFC4_ADD1	0.95	0.62	0.99	0.68
IFC4 ADD2 TC1	0.95	0.62	0.99	0.69

Table 6.8: The accuracy of axioms (1 being highest) calculated using established ground truth and axioms predicted by EPIXCO for all XSD elements.

Schema	`Class`	`Datatype`	`DatatypeProperty`	`ObjectProperty`
Example	1	1	1	-
cityGMLBase	1	1	1	1
building	1	1	1	1
bridge	1	1	1	1
IFCGeometries	1	1	1	1
IFC2x3 TC1	0.86	1	1	1
IFC4_ADD1	1	1	0.99	1
IFC4 ADD2 TC1	1	1	0.99	1

Table 6.9: The revised accuracy of axioms recalculated by removing XSD components of unimplemented patterns from ground truth.

6.3.3 Comparative Analysis

This section conducts the comparison of available and currently standardized ontology models of IFC formats with generated ontology models of respective formats using EPIXCO approach. The ontologies are compared with ontology metrics in the Table 6.10 of statistics reported by Protégé (Musen 2015) tool for quantitative evaluation of the buildingSmart standard ontology for *IFC4_ADD1* and *IFC2x3* formats by Pauwels and Terkaj (2016), and procedure herein proposed. The table shows that the axioms count may be moderate in comparison with standard ontology (due to unimplemented transformation patterns), their are noticeable key features in terms of *DL expressivity*, *Data Properties* and *SubObjectPropertyOf* axioms. Pauwels and Terkaj (2016) utilizes full DL expressivity, however, the library used for ontology graph in current implementation only supports OWL 2-RL profile[26] (subset of OWL2 DL) which is more efficient for SPARQL queries. On contrary, more axioms count of *Data Properties* and *SubObjectPropertyOf* are attained with transformation patterns implemented in EPIXCO, as compared

[26]https://github.com/RDFLib/OWL-RL

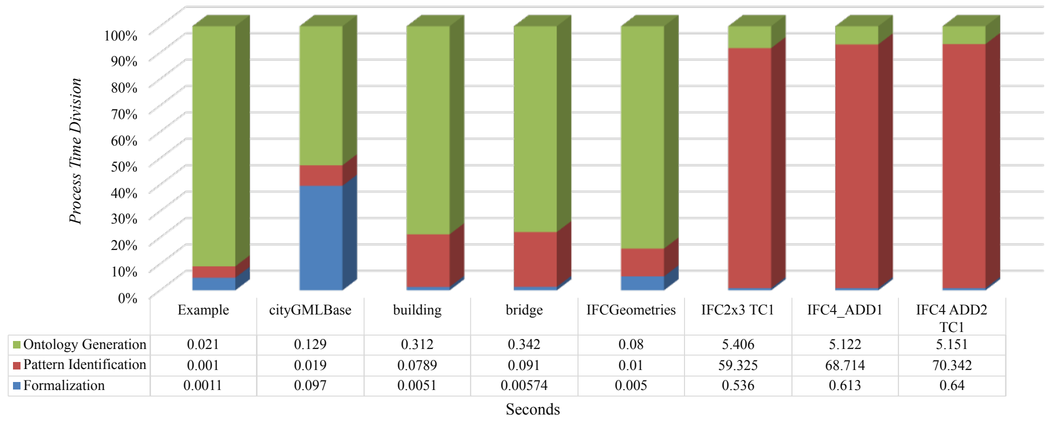
to ontology generated from EXPRESS based IFC schema. This is probably because the standard ontology considers most properties as *ObjectProperty* while Janus patterns specifies mapping of *DatatypeProperty* axioms for certain XSD constructions. Similarly, CityGML ontology generated by Métral et al. (2013) provides incorrect mappings of properties, e.g. *creationDate* is `owl:DatatypeProperty` as well as `owl:ObjectProperty`. Nonetheless, the comparison provides promising results with motive of more transformation patterns implemented and better DL expressivity to attain exhaustive ontology models with inference can be generated.

Metrics	**IFC4_ADD1**	**IFC2x3**	**IFC4_ADD1**	**IFC2x3**
Axiom	20675	17782	10252	10336
Logical axiom count	13867	11790	5301	5449
Declaration axioms count	4062	3544	3443	3559
Class count	1313	1093	1192	1200
Object property count	1580	1422	1416	1985
Data property count	5	5	1373	1181
Individual count	1158	1018	0	0
Annotation Property count	13	13	4	3
DL expressivity	SHIQ(D)	SHIQ(D)	ALN(D)	ALN(D)
SubClassOf	5099	4344	1077	1360
EquivalentClasses	2	2	0	0
DisjointClasses	2429	1888	0	0
SubObjectPropertyOf	3	3	0	0
InverseObjectProperties	94	96	0	0
FunctionalObjectProperty	1442	1292	0	0
TransitiveObjectProperty	1	1	0	0
ObjectPropertyDomain	1576	1418	1416	1985
ObjectPropertyRange	1576	1418	1185	1500
FunctionalDataProperty	5	5	0	0
DataPropertyDomain	5	5	729	220
DataPropertyRange	5	5	686	220
ClassAssertion	1630	1313	0	0
AnnotationAssertion	2739	2441	1507	1328

Table 6.10: Comparison between Pauwels and Terkaj (2016) and EPIXCO ontologies with statistics of retrieved from Protégé (Musen 2015) tool.

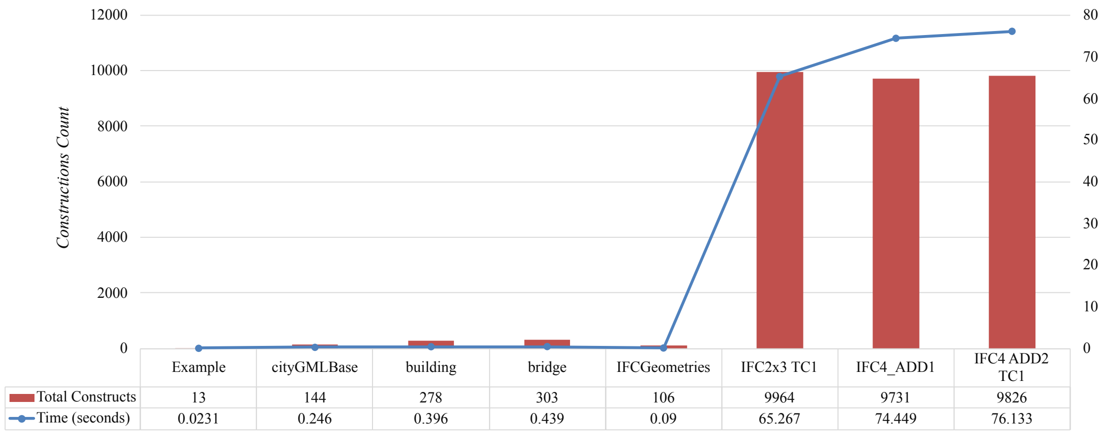
6.3.4 Performance Measures

EXPICO framework generates the ontology for a given XSD schema in a single workflow. The process is divided into three major phases, and the execution time for each phase is presented in Figure 6.5. It shows that schema with lesser XSD constructs utilized more time in ontology generation compared to other phases. For *cityGMLBase.xsd*, it is due to the excessive `documentation` XSD elements and the formalization process takes more time. The graph given in Figure 6.6 provides the performance comparison for the XSD constructs to the total execution time, which shows the time complexity exponentially correlates with a number of XSD constructions. The graph also shows that the execution time of 'IFC2x3 TC1' is lesser than other IFC schemas even though the number of XSD constructs is larger than others. The reason is 'IFC2X3 TC1' contains a large number of `enumeration` components which collectively increases the XSD constructions count



	Example	cityGMLBase	building	bridge	IFCGeometries	IFC2x3 TC1	IFC4_ADD1	IFC4 ADD2 TC1
🟩 Ontology Generation	0.021	0.129	0.312	0.342	0.08	5.406	5.122	5.151
🟥 Pattern Identification	0.001	0.019	0.0789	0.091	0.01	59.325	68.714	70.342
🟦 Formalization	0.0011	0.097	0.0051	0.00574	0.005	0.536	0.613	0.64

Figure 6.5: Time distribution among three phases of EPIXCO framework for generating ontology models of input XML schemas.



	Example	cityGMLBase	building	bridge	IFCGeometries	IFC2x3 TC1	IFC4_ADD1	IFC4 ADD2 TC1
🟥 Total Constructs	13	144	278	303	106	9964	9731	9826
● Time (seconds)	0.0231	0.246	0.396	0.439	0.09	65.267	74.449	76.133

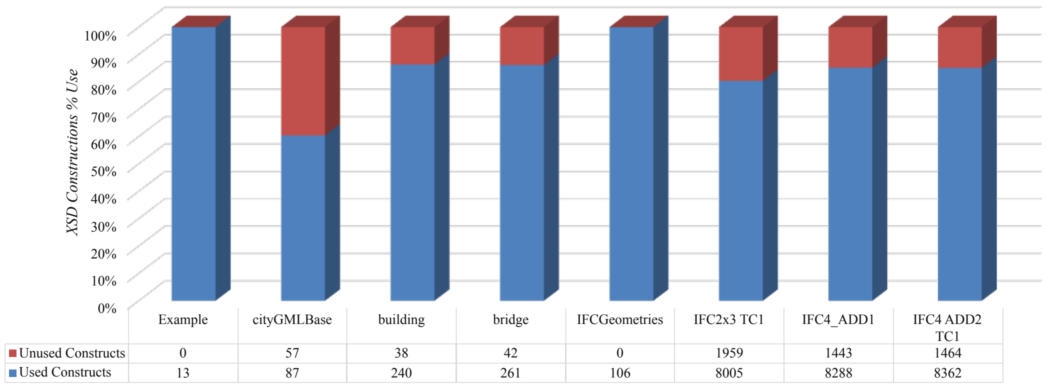
Figure 6.6: Graph comparison for the total processing time taken in ontology generation with respect to number of XSD constructions.

but not the pattern identification process that which is 59 seconds as compared to 70 seconds for 'IFC4 ADD2 TC1'. Therefore, the total execution time depends on a number of XSD constructs and the structural complexity of these XSD constructions, potentially affecting pattern identification time. The related works discussed do not have working prototypes on which such experiments for evaluation can be done. Hence, this study is unable to provide any comparative performance measures.

Regarding the complexity of the algorithms implemented in EPIXCO, it is always exponential. They are invoked in a sequence with nested sub-algorithms and depend on a number of XSD constructions. If n is the number of XSD constructs to be transformed, then the complexity of algorithms can be done in the order of $O(n^k)$ where k is the number of invoked sub-algorithm such that $1 \leq k \leq 5$.

6.4 Discussion and Considerations

In this Chapter, the proposed research framework is implemented for automatic ontology generation. However, defining an automatic process for such a dense task is a complex task, let alone defining it for complete ontology development. Scrutinizing XSD constructions in a nested hierarchy of constructs to identify correct patterns is a challenging task. Not all XSD constructions are transformed into OWL syntax as the current framework of EPIXCO implements 30 patterns. Figure 6.7 shows the percentage of constructs transformed that yields the ground truth accuracy provided in Table 6.8. In future, as more patterns are implemented, more XSD constructs are transformed into enriched and exhaustive ontology models.



	Example	cityGMLBase	building	bridge	IFCGeometries	IFC2x3 TC1	IFC4_ADD1	IFC4 ADD2 TC1
🟥 Unused Constructs	0	57	38	42	0	1959	1443	1464
🟦 Used Constructs	13	87	240	261	106	8005	8288	8362

Figure 6.7: Distribution of XSD constructions processed by EPIXCO framework from total constructs of input XML schema sets.

Janus and extended patterns in PIXCO define patterns on the well-established XSD block of constructions. However, XSD schema exists in multiple styles that it becomes almost impossible for every XSD constructs block to be defined, identified or implemented. For example, transformation pattern similar to #19 and #20 is required to be defined for inline `complexType` instead of only `simpleType`. Similarly, for pattern #37, if construction with `maxOccurs` attribute contains `"unbounded"` as attribute value, then occurrence needs to redefined for OWL model with possibly `owl:minCardinality="1"`, if no `minOccurs` attribute is present. On contrary, few patterns like #29 are never used, which is a similar pattern to #17 and #22. External schema entities are associated using pattern #24 and identified as `ObjectProperty`.

6.5 Overall Analysis and Conclusion

Throughout this Chapter, a detailed view of the most exciting parts of the implementation of this dissertation is presented. It includes the XSD components of eight schemas introduced in Chapter 3 are tailored in mathematical model presentation described in Chapter 5, along with transformation patterns formalization in FCA context that is used overall in pattern identification algorithms, and further generating the ontology models. It presents the experimental results to validate the thesis and to present the outcome by the EPIXCO framework. This last section of this Chapter also highlights some fundamental level issues with defining the automatic ontology development process, detailing our approach and solutions.

The implementation of this research work has been more complex than expected initially as ontology development is an evolving process. Since there is no direct validation available, manual efforts are required to validate the processing of XSD fragments to

OWL transformation, which later requires another pass for complete model generation. Overall, to define a completely automated process, laborious work is required to match the outcome as of the correspondence rules first and then proceed with the next step to iterates and repeat the process for all outcomes. Furthermore, some transformation patterns defined are not accurate and leave misconceptions as presented in Section 5.3 of Chapter 5, which further complicates the process. Despite these numerous problems in developing the procedure (with around 3000 lines of Python code), the framework could prove its initial statement.

Transforming XSD schema structures with different design styles is a complex process. Generating ontology with correct and complete semantics for large and strongly coupled XSD components becomes challenging and easy to maintain. The framework provided in-memory models of scalable, improvable information and evolved to output a more rich ontology model. The output correctness is evaluated in Section 6.3 using custom-defined ground truth, and although errors might be discovered, the results are presented with details to be observed to help with verification. Since the automation of ontology generation is an evolving process and requires great on-going effort, the last section discusses strengths about the framework, considerations made during the implementation and limitations that can guide future work. Therefore, based on these reasons, the system achieves the primary objective of this research to transform proper knowledge from XML schema into an ontology.

Chapter 7

Conclusion and Perspectives

7.1 Conclusion and Main Contributions

Over the past decade, the Semantic Web has shown the vision of ontology for integration systems with numerous potential applications. With the availability of various software tools (like Protégé (Musen 2015)) and methodologies (Hacherouf et al. 2015) for building ontologies, and different alignment techniques for mapping (merging) ontologies (like S-Match (Giunchiglia et al. 2012), GMO (Hu et al. 2005), COMA (Do and Rahm 2002)), researchers have carried out semantic web-based solutions for bringing together heterogeneous information of BIM and GIS technologies (Hor et al. 2018, Zhao et al. 2019). Nevertheless, as shown in Chapter 2 and 3, these solutions, as well as their ontology development techniques, mainly involves human intervention or are either assisted by semi-automatic approaches.

Limitations of these integration techniques in their adaption to general applications, among other reasons, mainly are: (i) the ontology building techniques are mostly laborious task, that are not adaptive to evolve and integrate other ontologies (ii) the complexity of automatically aligning ontologies, a complex task which is computationally expensive for significant information sources; (iii) the unavailability of readily tools to evaluate and validate based on background knowledge for built ontologies and their alignment.

This research contributes to the automatic transformation of XML schema into RDF/OWL with a system implemented that notably improves the complex transformations. The system follows the guidelines of Janus and PIXCO prototypes to implement transformation patterns to the best of its ability to provide maximum transformation. Furthermore, the system is extensive to augment more XSD components, enhanced correspondence rules, and improve produced OWL representations. One of the fundamental challenges addressed in this research is providing a solution that extracts information from different sources and favourably attempts to generate ontology for the respective source.

This research presented a preparatory framework in Section 5.1 which defines the broad approach incorporating the integration of BIM and GIS data with IoT sensors information by the transformation of XML-based IFC and CityGML data into an integrated RDF graph further linked with IoT. The preparatory conceptual approach presents a comprehensive framework divided into three-modules of ontology generation, ontology alignment and semantic graph generation. The first module of ontology generation implements a formal process that utilizes a patterns-based transformation approach for ifcXML and CityGML schema documents to generate their respective ontology models.

As the second module, the ontology alignment introduces a sub-framework that exploits semantic-based and structure-based alignment algorithms to find similarities among the entities of ontology models from the previous module to produce cross-domain ontology. Lastly, the cross-domain ontology is associated with data from IFC-XML and CityGML documents to construct an integrated RDF graph in the third module of semantic graph generation. Overall, the preparatory framework exploits semantic web technologies to present heterogeneous data formats into a common data model (i.e. RDF graph) and emphasizes that in order to construct a common data model, a thorough approach to generate ontology models is required.

In this dissertation, the first module of the preparatory framework is focused on a contemporary objective. It distinctively manifests a formal approach for the transformation of XML schema documents into OWL models. The research implements the EPIXCO (**E**nhanced **P**atterns **I**dentification for **X**SD **C**onversion to **O**WL) framework as an enhanced approach of PIXCO (Hacherouf et al. 2019) implementation utilizing Janus (Bedini et al. 2011) patterns generalized method for XSD schema to OWL transformation. However, in this study, the focused XSD documents are ifcXML and CityGML schema of BIM and CityGML domains, respectively. The detailed 43 transformation patterns provide a foundation to correspondence rules of XSD to OWL transformation, and 30 of these patterns are implemented in this research. The EPIXCO framework is divided into three major phases: formalization of XML schema and transformation patterns, patterns identification and filtration, and ontology generation. The complete framework is presented in Chapter 5, and implementation with its results are evaluated in Chapter 6, which shows promising contribution in the domain of building ontologies.

The contemporary framework of this dissertation has also published a preliminary study Usmani et al. (2020), and here in research implemented its enhanced version. The contribution of this dissertation also investigates the interoperability of BIM and GIS by participating in GeoBIM Benchmark (Noardo et al. 2020a) which depicts that off-the-shelf tools and preparatory software still lacks full support towards complicated BIM and GIS formats and their bidirectional conversions.

7.2 Research Challenges and Limitations

Using semantic web technology stack in ontology building techniques and using alignment techniques to achieve integrated solutions is one of the most exciting research areas. However, since semantic web technologies are still developing and maturing, it is often time-consuming to find efficient ways of using these technologies. The implemented framework has limitations in the evaluation and complete expression of semantics, as the approaches discussed in the related works do not have prototypes with readily available tools on which such experiments for evaluation could be done. For the available techniques, it is a time-consuming process to be deployed and further requires manual adjustments according to use case or applications. Therefore, this study cannot give any standard comparative analysis and performance measure between our approach and others.

A number of open issues can be listed in the implementation of this framework. Generally, the expressive power of OWL is more compared to XML semantics. However, the OWL has no direct representation for XSD components like `xs:list, xs:minLength` and `xs:maxLength`, as for some constructs like `list`. Instead, a workaround for using *LIST* ontologies is available for mapping. The ontology models generated for anonymous declared XSD elements or attributes are not fully connected because of missing domain or range restriction axioms for object and datatype properties. Currently, EPIXCO implements a subset of total Janus and PIXCO patterns, whereas all 43 patterns implementation will create more semantic rich ontology models. Inverse associations of OWL are also not dealt with in this version of EPIXCO, which provides more inference to the data semantics. Nonetheless, the ontology design is a creative and evolving process, and there is no single correct ontology for a domain Noy and McGuinness (2001). However, the quality of designed ontology can be assessed using it in applications for which it is designed.

Furthermore, GML is an XML grammar, and XML has become a de-facto standard for information exchange and is incredibly inefficient for network, processor and storage performance. Because RDF graphs contain more semantics of information represented in the source, converting XML-based information to RDF produces enormous output, making a single output file that is not best for query and data retrieval. Therefore, the semantic-based system requires enhancements in dealing with significant information sources. Lastly, there are very few globally agreed ontologies for construction domain Karan and Irizarry (2015); therefore, multi-disciplinary researchers develop their ontologies, limiting the effective transfer of information.

7.3 Perspectives and Future Works

The integration of BIM and GIS has come a long way, where innovative methods have been adapted to bridge the gap between two fundamentally distinct domains. The IoT information integration with BIM and GIS data has gain leading interest for Smarty City applications like Digital Twin and evolving towards Smart Data. Accordingly, the second and third module of preparatory framework proposes semantic web technologies as a promising approach on open-standard data-formats to achieve interoperability among BIM, GIS and IoT. In this study, the extension of this research is also presented with established framework modules highlight the comprehensive process that can utilize exhaustive ontology models generated from the EXPICO framework. The second module of preparatory framework presents how to obtain cross-domain ontology models and the third module reflects semantics graphs with integrated information. Henceforth, in the future, with the complete implementation of the proposed primary framework, a densely integrated data system as Smart Data can be provided that provides machine-readable information.

Another time, it is to emphasize that this research work targeted as much automation as possible, the reason why the focus of the most generic and relevant way to extract knowledge from different XSD documents was designed to generate ontology models that can be extended and integrated with future frameworks. The findings of this research are shared, and essential optimization strategies on the proposed method can be applied for efficient and effective systems. The notion to represent data from BIM, GIS and IoT in a semantic web technology stack with minimal human intervention elevates information integration and exchange. Generating ontology models of geospatial data and automatically interlinking their cross-domain entities with defined specifications will create the semantic process efficient, extensible, and flexible towards achieving interoperability. As more research is devoted to this area, the future of BIM and GIS integration by semantic web technologies is promising. Integrated information of BIM and GIS will nourish further research in knowledge-discovery and smart city applications. Incorporating IoT with integrated geospatial information will provide a digital environment, a space for extensive analysis, planning and better decision making, leading the urban innovation in lifestyle, environment and mobility with smart data.

Bibliography

Giovanni Acampora, Vincenzo Loia, and Autilia Vitiello. Enhancing ontology alignment through a memetic aggregation of similarity measures. *Information Sciences*, 250:1–20, 2013. ISSN 00200255. doi: 10.1016/j.ins.2013.06.052.

M. Hadi Amini, Hamidreza Arasteh, and Pierluigi Siano. Sustainable smart cities through the lens of complex interdependent infrastructures: Panorama and state-of-the-art. *Studies in Systems, Decision and Control*, 186:45–68, 2019. ISSN 21984190. doi: 10.1007/978-3-319-98923-5_3.

Sam Amirebrahimi, Abbas Rajabifard, Priyan Mendis, and Tuan Ngo. A BIM-GIS integration method in support of the assessment and 3D visualisation of flood damage to a building. *Journal of Spatial Science*, 61(2):317–350, 2016. ISSN 14498596. doi: 10.1080/14498596.2016.1189365.

Junghyen An and Young B. Park. Methodology for Automatic Ontology Generation Using Database Schema Information. *Mobile Information Systems*, 2018, 2018. ISSN 1875905X. doi: 10.1155/2018/1359174.

Salman Azhar. Building Information Modeling (BIM): Trends, Benefits, Risks, and Challenges for the AEC Industry. *Leadership and Management in Engineering*, 11(3):241–252, 2011. doi: 10.1061/(ASCE)LM.1943-5630.0000127.

Thomas Becker, Claus Nagel, and Thomas H. Kolbe. *A Multilayered Space-Event Model for Navigation in Indoor Spaces*, pages 61–77. Springer Berlin Heidelberg, 2009. ISBN 978-3-540-87395-2. doi: 10.1007/978-3-540-87395-2_5.

Ivan Bedini and Benjamin Nguyen. Automatic Ontology Generation : State of the Art. *Evaluation*, pages 1–15, 2007.

Ivan Bedini, Georges Gardarin, and Benjamin Nguyen. Deriving Ontologies from XML Schema. 2010a.

Ivan Bedini, Benjamin Nguyen, and Georges Gardarin. Janus: Automatic Ontology Builder from XSD Files. (May 2014):1–3, 2010b.

Ivan Bedini, Christopher Matheus, Peter F. Patel-Schneider, Aidan Boran, and Benjamin Nguyen. Transforming XML Schema to OWL using patterns. *Proceedings - 5th IEEE International Conference on Semantic Computing, ICSC 2011*, pages 102–109, 2011. doi: 10.1109/ICSC.2011.77.

Radim Bělohlávek. Introduction to Formal Concept Analysis. *Department of Computer Science, Faculty of Science, Palacký University*, 2008.

Nikos Bikakis, Chrisa Tsinaraki, Nektarios Gioldasis, Ioannis Stavrakantonakis, and Stavros Christodoulakis. The XML and semantic web worlds: Technologies, interoperability and integration: A survey of the state of the art. *Studies in Computational Intelligence*, 418(c):319–360, 2013. ISSN 1860949X. doi: 10.1007/978-3-642-28977-4_12.

Filip Biljecki, Jantien Stoter, Hugo Ledoux, Sisi Zlatanova, and Arzu Çöltekin. Applications of 3D city models: State of the art review. *ISPRS International Journal of Geo-Information*, 4(4):2842–2889, 2015. ISSN 22209964. doi: 10.3390/ijgi4042842.

Filip Biljecki, H. Ledoux, and Jantien Stoter. Generation of multi-LOD 3D city models in CityGML with the procedural modelling engine Random3Dcity. volume IV-4/W1, pages 51–59, 09 2016a. doi: 10.5194/isprs-annals-IV-4-W1-51-2016.

Filip Biljecki, Hugo Ledoux, and Jantien Stoter. An improved LOD specification for 3D building models. *Computers, Environment and Urban Systems*, 59:25–37, 2016b. ISSN 01989715. doi: 10.1016/j.compenvurbsys.2016.04.005.

Hannes Bohring and Sören Auer. Mapping XML to OWL ontologies. In Klaus P. Jantke, Klaus-Peter Fähnrich, and Wolfgang S. Wittig, editors, *Marktplatz Internet: Von e-Learning bis e-Payment, 13. Leipziger Informatik-Tage (LIT 2005)*, pages 147–156, Bonn, 2005. Gesellschaft für Informatik e. V.

Zouhaier Brahmia, Fabio Grandi, and Rafik Bouaziz. Conversion of XML schema design styles with StyleVolution. *International Journal of Web Information Systems*, 16(1): 23–64, 2019. ISSN 17440092. doi: 10.1108/IJWIS-05-2019-0022.

Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, and François Yergeau. Extensible Markup Language (XML) 1.0 (Fifth Edition) – W3C Recommendation 26 November 2008. https://www.w3.org/TR/2008/REC-xml-20081126/, 2012. (Last Accessed 1 February 2021).

Martin Breunig, Patrick Erik Bradley, Markus Jahn, Paul Kuper, Nima Mazroob, Norbert Rösch, Mulhim Al-Doori, Emmanuel Stefanakis, and Mojgan Jadidi. Geospatial data management research: Progress and future directions. *ISPRS International Journal of Geo-Information*, 9(2), 2020. ISSN 22209964. doi: 10.3390/ijgi9020095.

Joel Carneiro, Rosaldo J.F. Rossetti, Daniel C. Silva, and Eugenio C. Oliveira. BIM, GIS, IoT, and AR/VR Integration for Smart Maintenance and Management of Road

Networks: A Review. *2018 IEEE International Smart Cities Conference, ISC2 2018*, 2019. doi: 10.1109/ISC2.2018.8656978.

Yiqun Chen, Soheil Sabri, Abbas Rajabifard, Muyiwa Elijah Agunbiade, Mohsen Kalantari, and Sam Amirebrahimi. The design and practice of a semantic-enabled urban analytics data infrastructure. *Computers, Environment and Urban Systems*, 81(June 2019):101484, 2020. ISSN 01989715. doi: 10.1016/j.compenvurbsys.2020.101484.

Christophe Cruz and Christophe Nicolle. Ontology Enrichment and Automatic Population From XML Data. *Odbis*, pages 17–20, 2008.

Isabel F. Cruz, William Sunna, and Anjli Chaudhry. Semi-automatic Ontology Alignment for Geospatial Data Integration. In *Geographic Information Science*, pages 51–66, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-540-30231-5.

Richard Cyganiak, David Wood, and Markus Lanthaler. RDF 1.1 Concepts and Abstract Syntax – W3C Recommendation 25 February 2014. https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/, 2014. (Last Accessed 1 February 2021).

Tim Darr, John Hamilton, and Ronald Fernandes. Design considerations for xml-based te standards. http://hdl.handle.net/10150/595666, 2011. ISSN 0884-5123.

Yichuan Deng, Jack C.P. Cheng, and Chimay Anumba. Mapping between BIM and 3D GIS in different levels of detail using schema mediation and instance comparison. *Automation in Construction*, 67:1–21, 2016. ISSN 09265805. doi: 10.1016/j.autcon.2016.03.006.

Hong-Hai Do and Erhard Rahm. COMA — A system for flexible combination of schema matching approaches. In *VLDB '02: Proceedings of the 28th International Conference on Very Large Databases*, pages 610–621. Morgan Kaufmann, San Francisco, 2002. ISBN 978-1-55860-869-6. doi: https://doi.org/10.1016/B978-155860869-6/50060-3.

Charles M Eastman, Chuck Eastman, Paul Teicholz, Rafael Sacks, and Kathleen Liston. *BIM handbook: A guide to building information modeling for owners, managers, designers, engineers and contractors.* John Wiley & Sons, 2011.

O. EL Hajjamy, L. Alaoui, and Mohamed Bahaj. XSD2OWL2: Automatic mapping from XML schema into OWL2 ontology. *Journal of Theoretical and Applied Information Technology*, 95:1781–1796, 2017. ISSN 1992-8645.

M El-Mekawy and Anders Östman. Semantic Mapping: an Ontology Engineering Method for Integrating Building Models in IFC and CITYGML. *Proceedings of the 3rd ISDE Digital Earth Summit*, (January):1–11, 2010. ISSN 0041-1337. doi: 10.1021/bm049735c.

Mohamed El-Mekawy, Anders Östman b, and Ihab Hijazi c. An Evaluation of IFC-CityGML Unidirectional Conversion. *International Journal of Advanced Computer Science and Applications*, 3(5), 2012. doi: 10.14569/IJACSA.2012.030525.

Mohamed Farah, Hafedh Nefzi, and Imed Riadh Farah. A similarity-based framework for the alignment of an ontology for remote sensing. *Computers and Geosciences*, 96: 202–207, 2016. ISSN 00983004. doi: 10.1016/j.cageo.2016.08.018.

Karim Farghaly, Henry Fonbeyin Abanda, Christos Vidalakis, and Graham Wood. Semantic and syntactic interoperability of BIM and asset management data. *Proceedings of the 2019 European Conference on Computing in Construction*, 1(2017):406–413, 2019. doi: 10.35490/ec3.2019.154.

Matthias Ferdinand, Christian Zirpins, and David Trastour. Lifting XML Schema to OWL. In Nora Koch, Piero Fraternali, and Martin Wirsing, editors, *Web Engineering*, pages 354–358, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-540-27834-4.

Richelle Fosu, Kamal Suprabhas, Zenith Rathore, and Clark Cory. Integration of Building Information Modeling (BIM) and Geographic Information Systems (GIS) – a literature review and future needs. *Proc. of the 32nd CIB W78 Conference 2015, 27th-29th October 2015, Eindhoven, The Netherlands*, pages 196–204, 2015.

Shudi (Sandy) Gao, C. M. Sperberg-McQueen, and Henry S. Thompson. W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures – W3C Recommendation 5 April 2012. https://www.w3.org/TR/2012/REC-xmlschema11-1-20120405/, 2012. (Last Accessed 1 February 2021).

Qian Geng, Siyu Deng, Danping Jia, and Jian Jin. Cross-domain ontology construction and alignment from online customer product reviews. *Information Sciences*, 531:47–67, 2020. ISSN 00200255. doi: 10.1016/j.ins.2020.03.058.

Raji Ghawi and Nadine Cullot. Building Ontologies from XML Data Sources. pages 480–484, 01 2009. doi: 10.1109/DEXA.2009.68.

Fausto Giunchiglia, Aliaksandr Autayeu, and Juan Pane. S-Match: An open source framework for matching lightweight ontologies. *Semantic Web*, 3(3):307–317, 2012. ISSN 15700844. doi: 10.3233/SW-2011-0036.

Gerhard Gröger and Lutz Plümer. CityGML - Interoperable semantic 3D city models. *ISPRS Journal of Photogrammetry and Remote Sensing*, 71:12–33, 2012. ISSN 09242716. doi: 10.1016/j.isprsjprs.2012.04.004.

W3C OWL Working Group. OWL 2 Web Ontology Language Document Overview – W3C Working Draft 27 March 2009. https://www.w3.org/TR/2009/WD-owl2-overview-20090327/, 2009. (Last Accessed 1 February 2021).

W3C OWL Working Group. OWL 2 Web Ontology Language Document Overview (Second Edition) – W3C Recommendation 11 December 2012. https://www.w3.org/TR/2012/REC-owl2-overview-20121211/, 2012. (Last Accessed 1 February 2021).

Aditya Grover and Jure Leskovec. Node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 855—864. Association for Computing Machinery, 2016. ISBN 9781450342322. doi: 10.1145/2939672.2939754.

Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1992. ISSN 1042-8143. doi: https://doi.org/10.1006/knac.1993.1008.

Florian Haag, Steffen Lohmann, Stefan Negru, and Thomas Ertl. OntoViBe 2: Advancing the Ontology Visualization Benchmark. In *Proceedings of EKAW 2014 Satellite Events*, volume 8982 of *LNAI*, pages 83–98. Springer, 2015.

Mokhtaria Hacherouf, Safia Nait Bahloul, and Christophe Cruz. Transforming XML documents to OWL ontologies: A survey. *Journal of Information Science*, 41(2):242–259, 2015. ISSN 17416485. doi: 10.1177/0165551514565972.

Mokhtaria Hacherouf, Safia Nait-Bahloul, and Christophe Cruz. Transforming XML schemas into OWL ontologies using formal concept analysis. *Software and Systems Modeling*, 18(3):2093–2110, 2019. ISSN 16191374. doi: 10.1007/s10270-017-0651-4.

Carol L. Hanchette. *Geographic Information Systems*, pages 431–466. Springer New York, 2003. ISBN 978-0-387-22745-0. doi: 10.1007/0-387-22745-8_21.

Elio Hbeich and Ana Roxin. Linking BIM and GIS Standard Ontologies with Linked Data. *Researchgate-online*, (July):146–159, 2020.

Stefan Herle, Ralf Becker, Raymond Wollenberg, and Jörg Blankenbach. GIM and BIM: How to Obtain Interoperability Between Geospatial and Building Information Modelling? *PFG - Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, 88(1):33–42, 2020. ISSN 25122819. doi: 10.1007/s41064-020-00090-4.

A. E.Hadi Hor, G. Sohn, P. Claudio, M. Jadidi, and A. Afnan. A semantic graph database for BIM-GIS integrated information model for an intelligent urban mobility web application. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 4(4):89–96, 2018. ISSN 21949050. doi: 10.5194/isprs-annals-IV-4-89-2018.

A. H. Hor, A. Jadidi, and G. Sohn. BIM-GIS INTEGRATED GEOSPATIAL INFORMATION MODEL USING SEMANTIC WEB and RDF GRAPHS. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 3(July):73–79, 2016. ISSN 21949050. doi: 10.5194/isprs-annals-III-4-73-2016.

Ian Horrocks. Description Logic : A Formal Foundation for Ontology Languages and Tools. *Methods In Cell Biology*, 78(0091-679X (Print) LA - eng PT - Journal Article RN - 0 (Intermediate Filament Proteins) RN - 0 (Nerve Tissue Proteins) RN - 0 (Sulfur Radioisotopes) RN - 63-68-3 (Methionine) SB - IM):765–775, 2007.

Shaun Howell, Yacine Rezgui, and Thomas Beach. Integrating building and urban semantics to empower smart water solutions. *Automation in Construction*, 81:434–448, 2017. ISSN 0926-5805. doi: https://doi.org/10.1016/j.autcon.2017.02.004. URL `https://www.sciencedirect.com/science/article/pii/S0926580517301759`.

Wei Hu, Ningsheng Jian, Yuzhong Qu, and Yanbing Wang. GMO: A Graph Matching for Ontologies. volume 156, pages 43–50, 2005.

Javier Irizarry and Ebrahim Karan. Optimizing location of tower cranes on construction sites through GIS and BIM integration. *Electronic Journal of Information Technology in Construction*, 17:351–366, 09 2012a.

Javier Irizarry and Ebrahim P Karan. Optimizing location of tower cranes on construction sites through GIS and BIM integration. *Journal of information technology in construction (ITcon)*, 17(23):351–366, 2012b.

Javier Irizarry, Ebrahim P. Karan, and Farzad Jalaei. Integrating bim and gis to improve the visual monitoring of construction supply chain management. *Automation in Construction*, 31:241–254, 2013. ISSN 0926-5805. doi: https://doi.org/10.1016/j.autcon.2012.12.005.

Umit Isikdag, Jason Underwood, and Ghassan Aouad. An investigation into the applicability of building information models in geospatial environment in support of site selection and fire response management processes. *Advanced Engineering Informatics*, 22(4):504–519, 2008. ISSN 14740346. doi: 10.1016/j.aei.2008.06.001.

Elmira Jamei, Michael Mortimer, Mehdi Seyedmahmoudian, Ben Horan, and Alex Stojcevski. Investigating the role of virtual reality in planning for sustainable smart cities, 2017. ISSN 20711050.

Zhou Jiehan, Juha Pekka Koivisto, and Eila Niemelä. A survey on semantic web services and a case study. *Proceedings - 2006 10th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2006*, pages 763–769, 2006. doi: 10.1109/CSCWD.2006.253254.

Tae Wook Kang and Chang Hee Hong. A study on software architecture for effective BIM/GIS-based facility management data integration. *Automation in Construction*, 54:25–38, 2015. ISSN 09265805. doi: 10.1016/j.autcon.2015.03.019.

Ebrahim Karan and Javier Irizarry. Developing a Spatial Data Framework for Facility Management Supply Chains. pages 2355–2364, 05 2014. ISBN 978-0-7844-1351-7. doi: 10.1061/9780784413517.239.

Ebrahim P. Karan and Javier Irizarry. Extending BIM interoperability to preconstruction operations using geospatial analyses and semantic web services. *Automation in Construction*, 53:1–12, 2015. ISSN 09265805. doi: 10.1016/j.autcon.2015.02.012.

Ebrahim P. Karan, Javier Irizarry, and John Haymaker. BIM and GIS Integration and Interoperability Based on Semantic Web Technology. *Journal of Computing in Civil Engineering*, 30(3):4015043, 2016. ISSN 08873801. doi: 10.1061/(ASCE)CP.1943-5487.0000519.

John Keeney, Aidan Boran, Ivan Bedini, Christopher J. Matheus, and Peter F. Patel-Schneider. Approaches to relating and integrating semantic data from heterogeneous sources. *Proceedings - 2011 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2011*, 1(August):170–177, 2011. doi: 10.1109/WI-IAT.2011.129.

Karam Kim, Hyunjoo Kim, Wooyoung Kim, Changduk Kim, Jaeyo Kim, and Jungho Yu. Integration of ifc objects and facility management work information using Semantic Web. *Automation in Construction*, 87(January):173–187, 2018. ISSN 09265805. doi: 10.1016/j.autcon.2017.12.019.

Damien Lacoste, Kiran Prakash Sawant, and Suman Roy. An efficient XML to OWL converter. *Proceedings of the 4th India Software Engineering Conference 2011, ISEC'11*, pages 145–154, 2011. doi: 10.1145/1953355.1953376.

Xin Liu, Rui Liu, Graeme Wright, Jack Cheng, Xiangyu Wang, and Xiao Li. A State-of-the-Art Review on the Integration of Building Information Modeling (BIM) and Geographic Information System (GIS). *ISPRS International Journal of Geo-Information*, 6(2):53, 2017. ISSN 2220-9964. doi: 10.3390/ijgi6020053.

Nuno Lopes, Stefan Bischof, Orri Erling, Axel Polleres, Alexandre Passant, Diego Berrueta, Antonio Campos, Jérôme Euzenat, Kingsley Idehen, Stefan Decker, Stéphane Corlosquet, Jacek Kopecký, Janne Saarela, Thomas Krennwallner, Davide Palmisano, and Michal Zaremba. RDF and XML: Towards a Unified Query Layer. *W3C Workshop — RDF Next Steps*, pages 1–5, 2010.

Yuqian Lu and Muhammad Rizwan Asghar. Semantic communications between distributed cyber-physical systems towards collaborative automation for smart manufacturing. *Journal of Manufacturing Systems*, 55:348–359, 2020. ISSN 0278-6125. doi: https://doi.org/10.1016/j.jmsy.2020.05.001.

Zhiliang Ma and Yuan Ren. Integrated Application of BIM and GIS: An Overview. *Procedia Engineering*, 196(June):1072–1079, 2017. ISSN 18777058. doi: 10.1016/j.proeng.2017.08.064.

Eva Savina Malinverni, Berardo Naticchia, Francesco Di Stefano, and Joaquin Lopez Uriarte. Interoperability between GIS and BIM : From 3D GIS data to CityGML and graph database. pages 1–16, 2019.

J. McBeath, D. Farrell and S. Hinkelman. XML schema design guidelines – version 1.3. https://medbiq.org/std_specs/ techguidelines/xmldesignguidelines.pdf, 2004.

Claudine Métral, Roland Billen, Af Cutting-Decelle, and Muriel Ruymbeke. Ontology-based approaches for improving the interoperability between 3d urban models. *Electronic Journal of Information Technology in Construction*, 15:169–184, 02 2010.

Claudine Métral, Nizar Ghoula, Vitor Silva, and Gilles Falquet. A Repository of Information Visualization Techniques to Support the Design of 3D Virtual City Models. volume II-2/W1, 11 2013. doi: 10.5194/isprsannals-II-2-W1-247-2013.

Clement Mignard and Christophe Nicolle. Merging BIM and GIS using ontologies application to Urban facility management in ACTIVe3D. *Computers in Industry*, 65(9): 1276–1290, 2014. ISSN 01663615. doi: 10.1016/j.compind.2014.07.008.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and Their Compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, NIPS'13, pages 3111–3119. Curran Associates Inc., 2013.

A. Minutolo, A. Esposito, M. Ciampi, M. Esposito, and G. Cassetti. An Automatic Method for Deriving OWL Ontologies from XML Documents. In *2014 Ninth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, pages 426–431, 2014. doi: 10.1109/3PGCIC.2014.88.

Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz. OWL 2 Web Ontology Language Profiles (Second Edition) – W3C Recommendation 11 December 2012. https://www.w3.org/TR/2012/REC-owl2-profiles-20121211/, 2012. (Last Accessed 1 February 2021).

Mark A. Musen. The Protégé project: a look back and a look forward. *AI Matters*, 1(4): 4–12, 2015. doi: 10.1145/2757001.2757003.

F. Noardo, K. Arroyo Ohori, F. Biljecki, T. Krijnen, C. Ellul, L. Harrie, and J. Stoter. Geobim benchmark 2019: Design and initial results. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, 42(2/W13):1339–1346, 2019a. ISSN 16821750. doi: 10.5194/isprs-archives-XLII-2-W13-1339-2019.

F. Noardo, F. Biljecki, G. Agugiaro, K. Arroyo Ohori, C. Ellul, L. Harrie, and J. Stoter. GeoBIM Benchmark 2019: Intermediate results. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, 42(4/W15):47–52, 2019b. ISSN 16821750. doi: 10.5194/isprs-archives-XLII-4-W15-47-2019.

F. Noardo, K. Arroyo Ohori, F. Biljecki, C. Ellul, L. Harrie, T. Krijnen, M. Kokla, and J. Stoter. The ISPRS-EuroSDR GeoBIM Benchmark 2019. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, 43(B5):227–233, 2020a. ISSN 16821750. doi: 10.5194/isprs-archives-XLIII-B5-2020-227-2020.

Francesca Noardo, Lars Harrie, Ken Arroyo Ohori, Filip Biljecki, Claire Ellul, Helen Eriksson, Dogus Guler, Dean Hintz, Mojgan A Jadidi, and Maria Pla. Tools for BIM-GIS integration (IFC georeferencing and conversions): results from the GeoBIM benchmark 2019. *Preprints*, (July):1–35, 2020b. doi: 10.20944/preprints202007.0243.v1.

Natalya F. Noy and Deborah L. McGuinness. Ontology Development 101: A Guide to Creating Your First Ontology. *Stanford Knowledge Systems Laboratory*, page 25, 2001. ISSN 09333657. doi: 10.1016/j.artmed.2004.01.014.

Gozde Basak Ozturk. Interoperability in building information modeling for AECO/FM industry. *Automation in Construction*, 113(December 2019):103122, 2020. ISSN 09265805. doi: 10.1016/j.autcon.2020.103122.

Sangmin Park, Soung Hoan Park, Lee Won Park, Sanguk Park, Sanghoon Lee, Tacklim Lee, Sang Hyeon Lee, Hyeonwoo Jang, Seung Min Kim, Hangbae Chang, and Sehyun Park. Design and implementation of a Smart IoT based building and town disaster management system in Smart City Infrastructure. *Applied Sciences (Switzerland)*, 8 (11):1–28, 2018. ISSN 20763417. doi: 10.3390/app8112239.

Pieter Pauwels and Walter Terkaj. EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology. *Automation in Construction*, 63:100–133, 2016. ISSN 09265805. doi: 10.1016/j.autcon.2015.12.003.

Pieter Pauwels, Thomas Krijnen, Walter Terkaj, and Jakob Beetz. Enhancing the ifcOWL ontology with an alternative representation for geometric data. *Automation in Construction*, 80:77–94, 2017a. ISSN 09265805. doi: 10.1016/j.autcon.2017.03.001.

Pieter Pauwels, Sijie Zhang, and Yong Cheol Lee. Semantic web technologies in AEC industry: A literature overview. *Automation in Construction*, 73:145–165, 2017b. ISSN 09265805. doi: 10.1016/j.autcon.2016.10.003.

Ratchata Peachavanish, Hassan A. Karimi, Burcu Akinci, and Frank Boukamp. An ontological engineering approach for integrating CAD and GIS in support of infrastructure management. *Advanced Engineering Informatics*, 20(1):71–88, 2006. ISSN 1474-0346. doi: https://doi.org/10.1016/j.aei.2005.06.001.

María Poveda-Villalón, Asunción Gómez-Pérez, and Mari Carmen Suárez-Figueroa. OOPS! (OntOlogy Pitfall Scanner!): An On-line Tool for Ontology Evaluation. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 10(2):7–34, 2014.

Elie Raad and Joerg Evermann. The role of analogy in ontology alignment: A study on LISA. *Cognitive Systems Research*, 33:1–16, 2015. ISSN 13890417. doi: 10.1016/j.cogsys.2014.09.001.

Mohammed Jawaluddeen Sani and Alias Abdul Rahman. GIS and BIM integration at data level: A review. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, 42(4/W9):299–306, 2018. ISSN 16821750. doi: 10.5194/isprs-archives-XLII-4-W9-299-2018.

G. Saygi and F. Remondino. Management of Architectural Heritage Information in BIM and GIS: State-of-the-Art and Future Perspectives. *International Journal of Heritage in the Digital Era*, 2(4):695–713, 2013. ISSN 2047-4970. doi: 10.1260/2047-4970.2.4.695.

Guus Schreiber and Yves Raimond. RDF 1.1 Primer – W3C Working Group Note 24 June 2014. https://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/, 2014. (Last Accessed 1 February 2021).

Yongze Song, Xiangyu Wang, Yi Tan, Peng Wu, Monty Sutrisna, Jack C.P. Cheng, and Keith Hampson. Trends and opportunities of BIM-GIS integration in the architecture, engineering and construction industry: A review from a spatio-temporal statistical perspective, 2017. ISSN 22209964.

Rudi Stouffs, Helga Tauscher, and Filip Biljecki. Achieving complete and near-lossless conversion from IFC to CityGML. *ISPRS International Journal of Geo-Information*, 7(9), 2018. ISSN 22209964. doi: 10.3390/ijgi7090355.

Shu Tang, Dennis R. Shelden, Charles M. Eastman, Pardis Pishdad-Bozorgi, and Xinghua Gao. A review of building information modeling (BIM) and the internet of things (IoT) devices integration: Present status and future trends. *Automation in Construction*, 101 (January):127–139, 2019. ISSN 09265805. doi: 10.1016/j.autcon.2019.01.020.

John E. Taylor and Phillip G. Bernstein. Paradigm Trajectories of Building Information Modeling Practice in Project Networks. *Journal of Management in Engineering*, 25(2): 69–76, 2009. doi: https://10.1061/(ASCE)0742-597X(2009)25:2(69).

Chrisa Tsinaraki and Stavros Christodoulakis. Interoperability of XML Schema Applications with OWL Domain Knowledge and Semantic Web Tools. In Robert Meersman and Zahir Tari, editors, *On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS*, pages 850–869, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-76848-7.

A U Usmani, M Jadidi, G Sohn, Semantic Web, and Transformation Patterns. Automatic Ontology Generation of BIM and GIS Data. XLIII:77–80, 2020.

Hao Wang, Yisha Pan, and Xiaochun Luo. Integration of BIM and GIS in sustainable built environment: A review and bibliometric analysis. *Automation in Construction*, 103(March):41–52, 2019. ISSN 09265805. doi: 10.1016/j.autcon.2019.03.005.

Ning Wang and Raja R. A. Issa. Ontology-based integration of bim and gis for indoor routing. pages 1010–1019, 2020. doi: 10.1061/9780784482865.107.

Nora Yahia, Sahar A Mokhtar, and Abdelwahab Ahmed. Automatic Generation of OWL Ontology from XML Data Source. *International Journal of Computer Science Issues*, 9(2):77–83, 2012. ISSN 1694-0784.

Zhihang Yao, Claus Nagel, Felix Kunde, György Hudra, Philipp Willkomm, Andreas Donaubauer, Thomas Adolphi, and Thomas H Kolbe. 3DCityDB - a 3D geodatabase solution for the management , analysis , and visualization of semantic 3D city models based on CityGML. 2018.

Olga Zalamea, Jos Orshoven, and Steenberghen Thérèse. From a citygml to an ontology-based approach to support preventive conservation of built cultural heritage. 2013. ISBN 9783319337821.

Linlin Zhao, Zhansheng Liu, and Jasper Mbachu. Highway alignment optimization: An integrated BIM and GIS approach. *ISPRS International Journal of Geo-Information*, 8(4), 2019. ISSN 22209964. doi: 10.3390/ijgi8040172.

Junxiang Zhu, Graeme Wright, Jun Wang, and Xiangyu Wang. A critical review of the integration of geographic information system and building information modelling at the data level. *ISPRS International Journal of Geo-Information*, 7(2):1–16, 2018. ISSN 22209964. doi: 10.3390/ijgi7020066.

Junxiang Zhu, Yi Tan, Xiangyu Wang, and Peng Wu. BIM/GIS integration for web GIS-based bridge management. *Annals of GIS*, 00(00):1–11, 2020. ISSN 19475691. doi: 10.1080/19475683.2020.1743355.