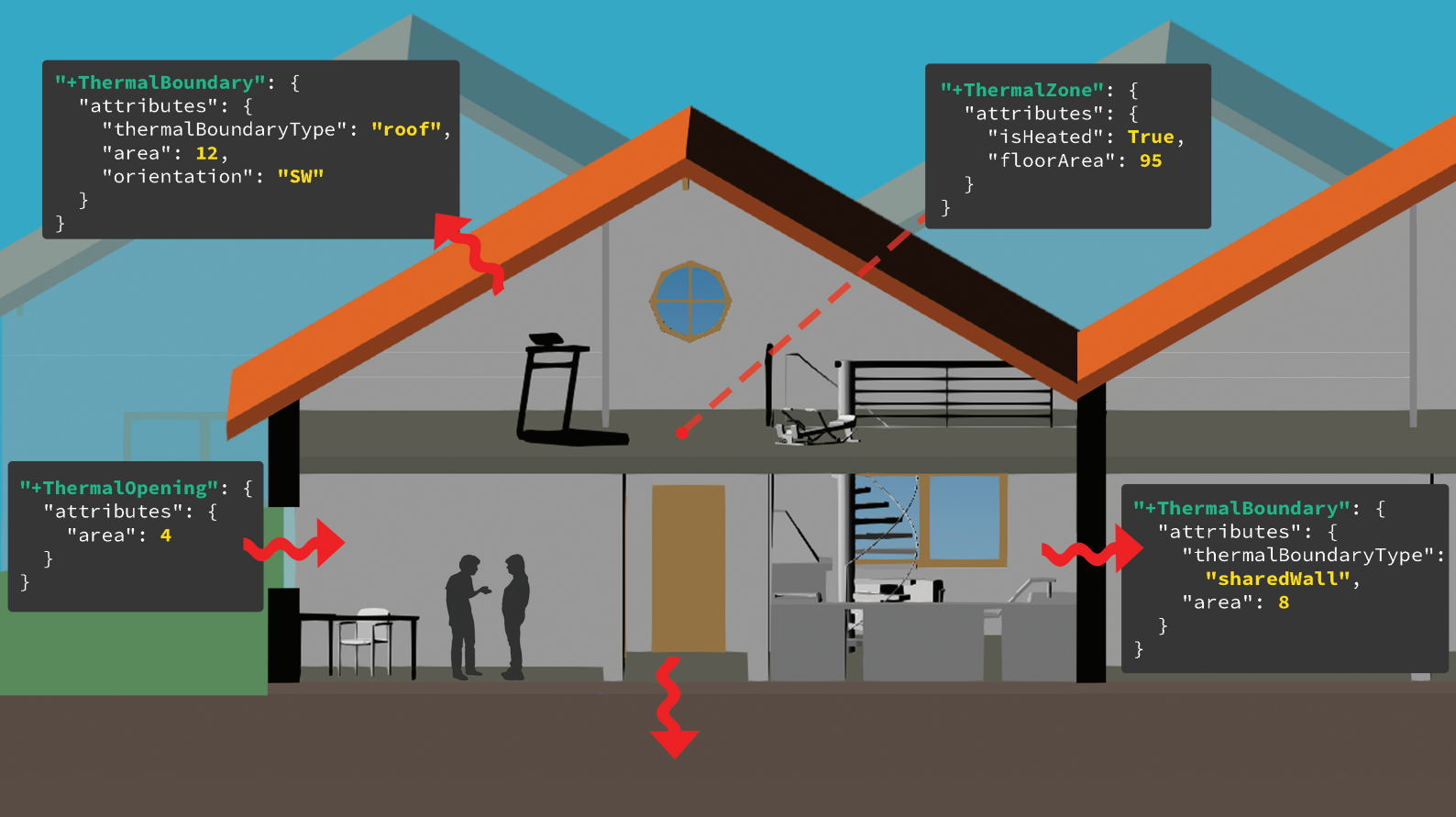


# Development and Testing of the CityJSON Energy Extension for Space Heating Demand Calculation

Özge Tufan | 2022



Cover illustration:

*IFC model*: Institute for Automation and Applied Informatics (IAI) / Karlsruhe Institute of Technology (KIT). Retrieval from: [https://www.ifcwiki.org/index.php?title=KIT\\_IFC\\_Examples](https://www.ifcwiki.org/index.php?title=KIT_IFC_Examples).

MSc thesis in Geomatics

# **Development and Testing of the CityJSON Energy Extension for Space Heating Demand Calculation**

Özge Tufan

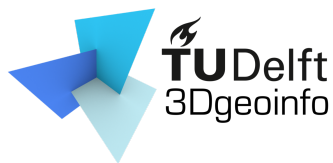
June 2022

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of Master of Science in Geomatics

Özge Tufan: *Development and Testing of the CityJSON Energy Extension for Space Heating Demand Calculation* (2022)

© This work is licensed under a Creative Commons Attribution 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by/4.0/>.

The work in this thesis was carried out in the:



3D geoinformation group  
Delft University of Technology

Supervisors: Camilo León-Sánchez  
Ken Arroyo Ohori  
Co-reader: Hugo Ledoux

# Abstract

3D city models are frequently used to acquire and store energy-related information of buildings to be used in energy applications, such as solar potential analyses and energy demand calculations. In this context, the most common data model is CityGML, which provides an application domain extension called the Energy ADE to store energy-related data in a systematic manner in XML format. On the other hand, CityJSON has been developed as a JSON-based encoding to exchange 3D city models, with the aim of eliminating the hierarchical structure and shortcomings of the XML-based CityGML. However, even though an extension mechanism exists in CityJSON, an energy-related CityJSON extension is not present in the current literature. Therefore, the aim of this thesis is to develop and test a CityJSON Energy Extension. To achieve this, the space heating demand calculation of buildings is chosen as the use case to validate and test the Extension.

In this thesis, a simplified version of the Energy ADE, called the Energy ADE KIT profile, is used as the first step to create a semi-direct translation to a CityJSON Energy Extension. After validating the Extension with the official validator of CityJSON, the space heating demand is calculated for a subset of the Rijssen-Holten in the Netherlands according to the Dutch standard *NTA 8800*. Required input data is collected from various data sources, such as the 3D city model of the area and the TABULA building physics library, and stored in the CityJSON Energy Extension to test its usability for the use case. The Extension is then improved depending on the results of the tests based on the use case.

The results show that the semi-direct translation lacked numerous objects and attributes to store certain input data for the use case, while the final version of the CityJSON Energy Extension fully supports the use case with the possibility of storing all required input data. Furthermore, while the semi-direct translation contained deep hierarchical structures, these were eliminated in the final Extension to comply with the design decisions behind CityJSON. The main differences between the Energy ADE and the CityJSON Energy Extension reflected this philosophy as well, where the former data model was built with a deep hierarchical structure, while the latter flattens this hierarchy as much as possible by using the characteristics of JSON. In addition, a comparison in file sizes showed that the input 3D city model of the study area in CityJSON format (with 3318 objects) had a file size of 40.6 MB, whereas the output CityJSON + Energy Extension file with all input and output data was 65.8 MB (with 108732 objects). It was discussed that this increase of 25.2 MB in file size is not significant, considering the high increase in the number of objects stored in the file. On the other hand, the space heating demand calculation resulted in negative values for 32 buildings in the study area, and for the majority of buildings during the solar gains computation, which was not expected. While the possible reasons were detected with numerous tests, a solution could not be developed in the given time frame of the thesis. Overall, this thesis showed that the CityJSON Energy Extension can provide an easy to use alternative to CityGML Energy ADE, where the Extension files can be simply parsed by software and easily understood by the user without reaching a large file size.



# Acknowledgements

I would like to express my sincere gratitude to my supervisors, Camilo León-Sánchez and Ken Arroyo Ohori, for their guidance and support throughout my thesis. Both Camilo and Ken have devoted a significant amount of their time to have frequent meetings and brainstorming sessions with me, and to answer my endless questions. Without their patience and support, I could not have found the daily motivation that I needed to complete my thesis. I would also like to thank my co-reader, Hugo Ledoux, who has not only given constructive feedback on my thesis, but also answered all my CityJSON-related questions. Furthermore, I would like to thank Giorgio Agugiaro, who has had numerous meetings with me and gave me his valuable feedback throughout my thesis. Without his help, I would have struggled greatly to overcome certain challenges. Special thanks to Stelios Vitalis for his suggestions and valuable contribution to help me solve certain problems in my thesis. Finally, I would like to thank my mom, my friends, and Matteo for always being there and supporting me during this somewhat lonely process.





# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Objectives and research questions . . . . .	2
1.2	Scope . . . . .	2
1.3	Thesis structure . . . . .	3
<b>2</b>	<b>Related work</b>	<b>5</b>
2.1	CityGML . . . . .	5
2.1.1	Core and Building modules . . . . .	5
2.1.2	Level of Detail . . . . .	6
2.1.3	Extending CityGML . . . . .	6
2.2	CityJSON . . . . .	7
2.2.1	Extending CityJSON . . . . .	9
2.3	Current CityJSON Extensions . . . . .	10
2.3.1	CityJSON Point Cloud Extension . . . . .	10
2.3.2	CityJSON Noise Extension . . . . .	11
2.4	Energy ADE . . . . .	12
2.4.1	Energy ADE KIT Profile . . . . .	13
2.5	Space heating demand calculation and 3D city models . . . . .	15
2.5.1	3D city models and the Energy ADE for space heating demand calculation . . . . .	15
2.5.2	Current energy simulation software with 3D city models . . . . .	17
<b>3</b>	<b>Methodology</b>	<b>20</b>
3.1	Semi-direct translation from the Energy ADE . . . . .	21
3.1.1	New City Objects . . . . .	21
3.1.2	New attributes to existing City Objects with additional data types . . . . .	21
3.1.3	New non-City Objects . . . . .	22
3.1.4	Relations among City Objects and non-City Objects . . . . .	23
3.2	Validation of the CityJSON Energy Extension . . . . .	23
3.2.1	Validation through the use case . . . . .	23
3.2.2	Validation through cjval . . . . .	23
3.2.3	Validation against the Energy ADE KIT profile . . . . .	24
3.3	Space heating demand calculation . . . . .	24
3.3.1	Calculation method . . . . .	24
3.3.2	Required input data for space heating demand calculation . . . . .	29
3.4	Improvements on the CityJSON Energy Extension . . . . .	30
<b>4</b>	<b>Study area and datasets</b>	<b>33</b>
4.1	Study area . . . . .	33
4.2	Collection of input data for space heating demand . . . . .	34
4.2.1	3D city model of Rijssen-Holten . . . . .	35
4.2.2	Basisregistratie Adressen en Gebouwen (BAG) . . . . .	37
4.2.3	Meteorological Data Portal . . . . .	38
4.2.4	TABULA Building Physics Library . . . . .	38
4.2.5	NTA 8800 . . . . .	40

<b>5</b>	<b>Implementation</b>	<b>42</b>
5.1	Mapping rules of the semi-direct translation . . . . .	42
5.1.1	Creating new City Objects . . . . .	42
5.1.2	New attributes to existing City Objects . . . . .	43
5.1.3	Creating new non-City Objects . . . . .	43
5.1.4	New data types, enumerations and code lists . . . . .	45
5.1.5	Relations among City Objects and non-City Objects . . . . .	45
5.2	Experiments on the storage of input data in the CityJSON Energy Extension . . . . .	46
5.3	Calculation of space heating demand . . . . .	48
5.3.1	Limitations on the retrieval of data from the CityJSON Energy Extension during the space heating demand calculation . . . . .	49
5.4	Improvements on the semi-direct translation . . . . .	50
5.4.1	Reconsidering the Energy ADE KIT profile attributes . . . . .	50
5.4.2	Associations among objects . . . . .	50
5.4.3	Removal of deep hierarchies . . . . .	51
5.4.4	Naming conventions . . . . .	53
<b>6</b>	<b>Results and Analysis</b>	<b>55</b>
6.1	Semi-direct translation versus the final CityJSON Energy Extension . . . . .	55
6.1.1	Change in the hierarchical structure . . . . .	55
6.1.2	Change in the storage of relations . . . . .	56
6.2	The Energy ADE KIT profile versus the CityJSON Energy Extension . . . . .	57
6.2.1	Changes in the used elements . . . . .	57
6.3	Comparison of file size . . . . .	59
6.4	Results of the space heating demand calculation . . . . .	60
6.4.1	Comparison of results with energy simulation tools . . . . .	65
<b>7</b>	<b>Conclusions and Future Work</b>	<b>70</b>
7.1	Research overview . . . . .	70
7.2	Discussion . . . . .	72
7.2.1	Contributions . . . . .	72
7.2.2	Limitations . . . . .	72
7.3	Future work . . . . .	74
7.3.1	Further development of the Energy Extension . . . . .	74
7.3.2	Additional methods for validation and comparison . . . . .	74
7.4	Self-reflection . . . . .	75
<b>A</b>	<b>Comparison of the Extension elements</b>	<b>78</b>

# List of Figures

2.1	A subset of the CityGML Building module focusing on semantic features. . . . .	6
2.2	Geometrical and semantic differentiation of the five LODs in CityGML. . . . .	7
2.3	The hierarchy of CityJSON, structured as 1st- and 2nd-level city objects. . . . .	8
2.4	UML diagrams of the Noise ADE. . . . .	11
2.5	The modules and dependencies of the Energy ADE. . . . .	13
2.6	Approaches in Urban Energy Modelling for energy demand estimations. . . . .	16
2.7	Topologically adjacent buildings and the party walls between them as stored in the 3D city model. . . . .	17
3.1	Overview of the methodology. . . . .	20
3.2	UsageZone object defined as a City Object in the Energy ADE KIT profile. . . . .	21
3.3	The “hook” mechanism of the Energy ADE KIT profile to define new attributes, and the newly defined data types, enumerations, and code lists. . . . .	22
3.4	Non-City Objects defined in the Energy ADE KIT profile. . . . .	22
3.5	Example of unidirectional and bi-directional relations. . . . .	30
4.1	The location of the Rijssen-Holten municipality in the Netherlands, and the study area with used buildings. . . . .	33
4.2	Distribution of the building functionalities, and the building typologies in the study area. . . . .	34
4.3	The 3D city model of the study area: a subset of Rijssen-Holten in the Netherlands. . . . .	35
4.4	Residential units as points in the BAG dataset. . . . .	38
4.5	TABULA building classification for the Netherlands. . . . .	39
5.1	The hierarchy of CityJSON Energy Extension objects to store the needed input data. . . . .	47
5.2	A party wall between two buildings, and the remaining unshared wall surface with a considerably small area. . . . .	49
5.3	The relation between ThermalBoundary and Construction objects in the semi-direct translation and in the final version of the Extension. . . . .	51
6.1	The links to be passed to reach weather data values stored with the semi-direct translation, and the final CityJSON Energy Extension. . . . .	56
6.2	The definition of a <i>volume</i> attribute for Building objects with its value and unit of measurement as two separate properties, and the links needed to be passed to reach the volume data with and without the unit of measurement. . . . .	56
6.3	The relationship between a UsageZone object and its occupant(s) stored in attributes, and in properties of the object. . . . .	57
6.4	Extension CityObjects before and after the parent/children relationships were reconsidered. . . . .	58
6.5	TimeSeries objects defined in the Energy ADE KIT profile with inheritance, and in the CityJSON Energy Extension without the abstract class. . . . .	58
6.6	Considered and omitted buildings during the space heating demand calculation in the center and a residential area in Rijssen. . . . .	60
6.7	Energy demand (kWh) in the month January in the center and a residential area in Rijssen. . . . .	61

*List of Figures*

6.8	Examples of corner and middle buildings with the type Terrace House. . . . .	61
6.9	Two buildings with differing sizes from the study area. . . . .	62
6.10	Average monthly energy demand (kWh) and the distribution of values on logarithmic scale depending on the construction period. . . . .	63
6.11	Average monthly energy demand (kWh) and the distribution of values on logarithmic scale depending on the building type. . . . .	64
6.12	The differences between the average energy demand values calculated with NTA 8800 and Simstadt for a 10-building subset. . . . .	65
A.1	A comparison of the Energy ADE KIT profile and CityJSON Energy Extension elements.	80

# List of Tables

3.1	Validation functionalities of <i>cjval</i> . . . . .	24
3.2	Elements of the space heating demand calculation according to <i>NTA 8800</i> , and the needed input data. . . . .	29
4.1	Input parameters of the space heating demand calculation and the data sources. . . .	34
4.2	The attributes stored in the 3D city model of Rijssen-Holten. . . . .	36
4.3	The available attributes of the selected residential unit in the BAG dataset. . . . .	38
6.1	File size and the number of objects stored in each input/output file used in the thesis.	60
6.2	Comparison of energy demand in January for corner, stand-alone and middle buildings.	62
6.3	Comparison of energy demand in January for two buildings with distinct construction periods. . . . .	62
6.4	Comparison of energy demand in January for two buildings with distinct volumes. . .	63
6.5	Average volume of the three building typologies in the study area. . . . .	65
6.6	Comparison of monthly space heating demand values for a 10-building subset calculated with <i>NTA 8800</i> and <i>SimStadt</i> . . . . .	66
6.7	Buildings with negative energy demand values. . . . .	67
6.8	Change in the energy demand values for one building after adjustments in the calculation method. . . . .	68

# List of Schemas

2.1	A basic CityJSON Extension schema with all possible properties. . . . .	9
2.2	An instance of the +SolitaryVegetationObjectPC object. . . . .	10
2.3	Definition of the +NoiseCityFurnitureSegment object in the Noise Extension. . . . .	12
2.4	Building CityObject, extended with noise-related attributes. . . . .	12
5.1	Definition of the UsageZone CityObject in the Extension schema, and the exemplary JSON object with example data. . . . .	43
5.2	Extra attributes defined for the Building CityObject, and how it is implemented for a Building object. . . . .	43
5.3	EnergyDemand, a non-CityObject, defined with the "extraCityObjects" keyword, and an exemplary JSON object with the corresponding attributes. . . . .	44
5.4	Definition of the WeatherData object, and its use together with a TimeSeries object. . . . .	44
5.5	floorArea data type defined as a subschema, and referenced as the data type of the floorArea attribute of Buildings. . . . .	45
5.6	ThermalZone object with its parents and children properties, and the use of this concept to form a relation with a UsageZone object. . . . .	46
5.7	The relation between a Building and EnergyDemand object. . . . .	46
5.8	UsageZone CityObject with its weatherData and energyDemand relations as attributes, and the occupants relation defined as a property. . . . .	52
5.9	WeatherData, defined as a subschema, and its use in a weatherData attribute for a Building object. . . . .	52
5.10	The use of the "+unitOfMeasurement" root property in a CityJSON + Energy Extension file. . . . .	53



# Acronyms

ADE	Application Domain Extension . . . . .	1
OGC	Open Geospatial Consortium . . . . .	5
LoD	Level of Detail . . . . .	6
BIM	Building Information Modeling . . . . .	5
BAG	Basisregistratie Adressen en Gebouwen . . . . .	ix





# 1 Introduction

Energy performance of buildings is a prevalent discussion all around the world, considering that more than one-third of global energy consumption and almost 40% of energy-related CO<sub>2</sub> emissions are caused by buildings and construction activities [United Nations Environment Programme, 2020]. Modelling the current energy use and predicting future scenarios require comprehensive knowledge about buildings and tools to process and simulate this information [Agugiaro et al., 2018]. As a result, *Urban Energy Modelling* has gained substantial importance over the years [Ali et al., 2021], in which semantic 3D city models are frequently used to obtain and store energy-related data in urban areas [Kaden and Kolbe, 2014; Agugiaro, 2016b].

In this context, the CityGML data model presents a standardised way of storing and exchanging 3D city models with an XML-based encoding also called CityGML [Gröger et al., 2012]. While CityGML is used for numerous applications ranging from energy and noise simulations to disaster management [Gröger and Plümer, 2012], the complexity and verbosity of the XML format has led to the development of two other encodings for CityGML: an SQL-based encoding called 3DCityDB [Yao et al., 2018], and a JSON-based encoding called CityJSON [Ledoux et al., 2019], both of which aim to provide a more efficient way of storing information.

While CityGML covers a wide variety of city objects with their geometries and semantics, the core data model can be extended for specific use cases with the concept of Application Domain Extension (ADE), which allows the addition of classes and attributes to the data model in a systematic way [Gröger and Plümer, 2012]. One of the most comprehensive of these is the Energy ADE, which is used to store detailed energy-related data of buildings [Benner, 2018]. This information can then be used in both steady-state and dynamic energy simulations to analyse the current state of buildings as well as to obtain future predictions. For instance, a prominent application of Energy ADE is to assess the energy performance of buildings by focusing on specific implementations such as energy demand analysis or solar irradiation of buildings [Agugiaro et al., 2018]. Furthermore, it can be used to determine future actions such as strategies on renewable energy or building refurbishment measures.

Energy ADE was originally provided with an XML-based encoding, however, due to the complexities of both the XML format and the Energy ADE data model itself, a database implementation was later created by extending the already existing 3DCityDB to store energy-specific data of buildings [Agugiaro and Holcik, 2017]. In the recent literature, this format has been used in many energy-focused studies, proving its efficiency over the XML-based encoding [Skarbal et al., 2017; Pasquinelli et al., 2019; Rossknecht and Airaksinen, 2020]. On the other hand, even though an Extension mechanism exists for CityJSON to define additional objects and attributes for the core data model, an energy-related CityJSON extension is not present in the current literature.

Compared to the CityGML ADEs, CityJSON extensions follow stricter guidelines, making them similar to the original CityJSON files in terms of structure, and thus easier to read and process by software [Ledoux et al., 2019]. Moreover, the compact structure of CityJSON without deep hierarchies may be beneficial for an energy-related extension, considering the high amount of data needed to be stored for various energy applications. In addition, such an extension would enhance the use of CityJSON for energy-related use cases while helping to close the gap with the CityGML data model. Therefore, this thesis investigates the steps necessary to design an Energy Extension for CityJSON. The Extension will be developed based on the existing extension mechanism of CityJSON, and the

space heating demand calculation is chosen as the use case to structure the Energy Extension, and for the validation, testing and improvements.

### 1.1 Objectives and research questions

The main objective of this thesis is to develop and test an Energy Extension for CityJSON that supports the calculation of space heating demand of buildings. For this, the Extension must enable the storage of all input data needed for this calculation as well as the resulting energy demand values in an efficient way. While CityJSON files already have, on average, a compression factor of 6 compared to XML-based CityGML files [Ledoux et al., 2019], the efficiency based on the use case and input data is crucial to assess as well to improve the Extension. Furthermore, the Extension must follow the philosophy and the main design decisions behind CityJSON to ensure compatibility with the core data model.

In order to achieve the outlined objective, this thesis aims to answer the following research question:

*How can a CityJSON Energy Extension be used to support the calculation of space heating demand of buildings?*

To answer this question, the following sub-questions are specified:

- How can different types of objects, other than CityObjects, be defined in a CityJSON Extension?
- How should the CityJSON Energy Extension differ from the Energy ADE?
- How can space heating demand calculation be used during the design phase to test and improve the CityJSON Energy Extension?
- To what extent is it possible to map CityGML ADEs to CityJSON Extensions? Should the CityJSON schema be extended to make this process more straightforward?

### 1.2 Scope

Considering the specified objectives and research questions, the scope of this thesis is defined as follows:

- The proposed CityJSON Energy Extension focuses on a specific use case, which is the calculation of space heating demand of buildings. Therefore, the definition of additional objects and attributes for other energy applications is out of the scope of this thesis.
- During the design and testing of the proposed Extension, limitations of the CityJSON data model, its extension mechanism, and its validator, *cjval*<sup>1</sup>, will be investigated. While certain recommendations will be provided to these projects, implementing solutions to these limitations is beyond the scope of this thesis.

---

<sup>1</sup><https://github.com/cityjson/cjval>

- Space heating demand will be calculated for a subset of the Rijssen-Holten municipality in the Netherlands with a steady-state energy balance method, as described in the Dutch standard called the *NTA 8800* [Royal Netherlands Standardization Institute, 2022]. On the other hand, the implementation of a dynamic simulation method is not a part the scope, because of the need for high-resolution input data, which is generally not possible to obtain for city-scale calculations.

### 1.3 Thesis structure

Following this section of Introduction, the rest of the thesis is structured as follows:

**Chapter 2** provides an overview of the theoretical background that is used in the rest of this thesis. The CityGML, CityJSON, and Energy ADE data models are described in detail, and various approaches in Urban Energy Modelling related to the space heating demand calculation are explained. Furthermore, the theoretical background is supported with previous studies related to the thesis topic.

**Chapter 3** introduces the methodology in detail, which is proposed to answer the research questions. In addition, the main design decisions are described in this part of the thesis.

**Chapter 4** presents the details of the study area, and the collection of required input data for the space heating demand calculation.

**Chapter 5** presents the implementation of the methodology and the predetermined design decisions, as well as the additional tests and experiments.

**Chapter 6** describes the results obtained after the implementation of the methodology, with a focus on the results of the performed tests and experiments.

**Chapter 7** concludes the thesis, in which a summary of the main results is included and the research questions are answered. Furthermore, the limitations of the thesis are presented, and recommendations are provided for future work.



## 2 Related work

This section provides an overview of the theoretical background that this thesis is built upon, together with a review of the related studies. First, the CityGML data model is introduced with a focus on the Building module and the ADE concept (Section 2.1). Second, the CityJSON encoding and its extension mechanism are explained (Section 2.2), followed by a description of two of the current CityJSON extensions (Section 2.3). Then, the Energy ADE data model is described with its thematic modules (Section 2.4). Finally, relevant studies on space heating demand calculation are discussed, and the contribution of 3D city models, the Energy ADE, and current software for energy simulations are explained (Section 2.5).

### 2.1 CityGML

3D city models represent the common objects in urban areas with their three-dimensional geometries, which can be used in a large number of domains for specific applications [Biljecki et al., 2015]. While some of these applications focus purely on geometrical properties, others require also the semantic aspects in a structured way to characterize urban objects, especially buildings [Kolbe, 2009]. In this context, the three most frequently used standards include Building Information Modeling (BIM), INSPIRE Data Specification on Buildings, and the CityGML data model. While BIM is used to store and manage geometric and semantic data of individual buildings through their life cycles [Sanhudo et al., 2018], INSPIRE Data Specification on Buildings enables the storage of data on buildings on city, province, or country level [INSPIRE Thematic Working Group Buildings, 2013]. However, it can be seen that these two standards focus only on buildings, and not on various types of objects in cities. On the other hand, CityGML is an open data model adopted by the Open Geospatial Consortium (OGC) to store and exchange 3D city models. In this standard, the spatial properties of objects are represented with a subset of the geometry model of GML3, and their topological, semantic, and visual properties are also considered [Gröger et al., 2012]. CityGML has a broader focus compared to the aforementioned standards since different types of city objects are considered, and is consequently used the most for city-scale applications. In the following paragraphs, general characteristics of CityGML that are relevant to the thesis are introduced.

#### 2.1.1 Core and Building modules

CityGML consists of a Core module and 10 thematic modules. The Core module defines abstract base classes and basic data types, where the base class of all thematic modules is the abstract class called *CityObject* [Gröger and Plümer, 2012]. The thematic modules, on the other hand, provide class definitions and properties of the most common objects in cities, namely relief, buildings, tunnels, bridges, water bodies, transportation objects, vegetation objects, city furniture, land use, and city object groups. The main class of each thematic module is defined as a subclass of *CityObject*, inheriting its attributes and relations.

The most detailed thematic module of CityGML is the Building module (Figure 2.1), in which the geometry, semantics and various attributes of buildings are modelled. The base class of the Building module, *AbstractBuilding*, includes general attributes such as the class and function of a building,

year of construction, roof type and number of storeys. Moreover, a building may be represented with a Building or BuildingPart object, as subclasses of *AbstractBuilding*, depending on whether it consists of a homogeneous part [Gröger et al., 2012]. Finally, a building may be represented as semantic objects with boundary surfaces such as WallSurface, RoofSurface, and GroundSurface.

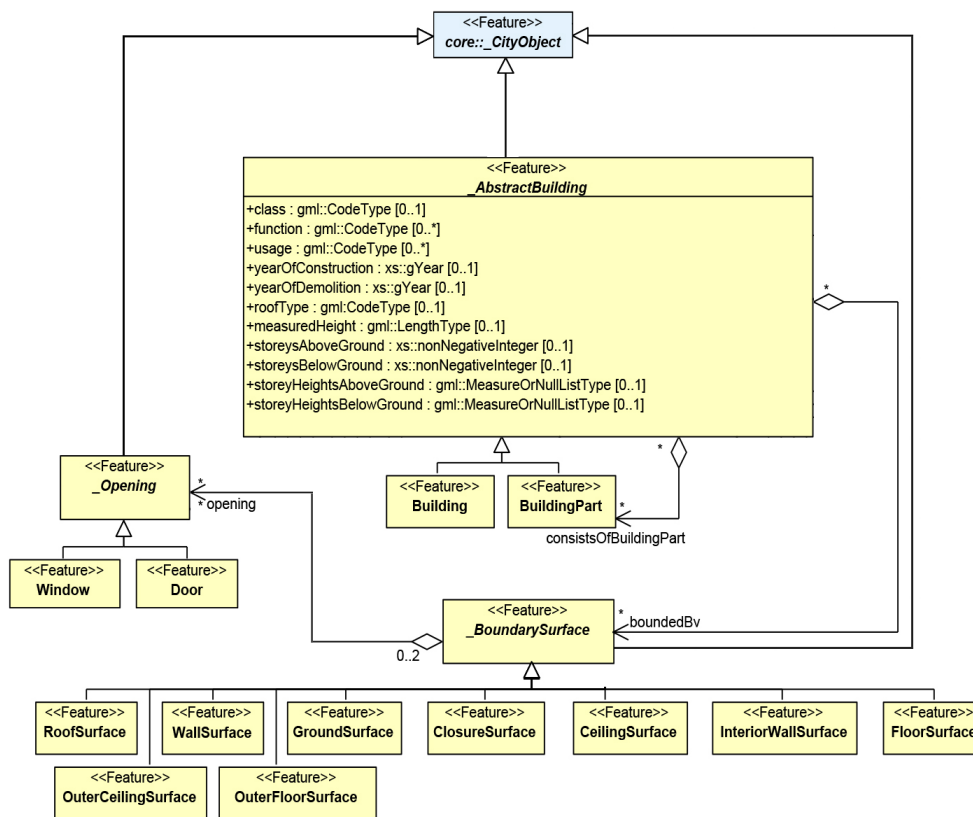


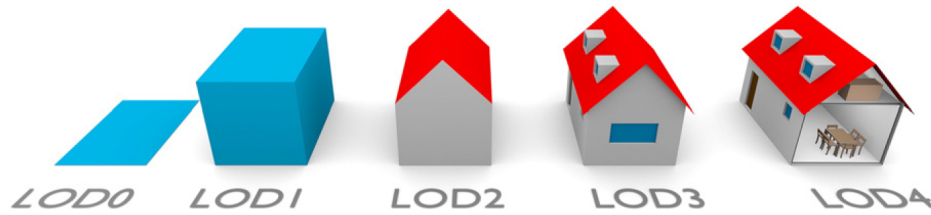
Figure 2.1: A subset of the CityGML Building module focusing on semantic features. Figure adapted from Gröger et al. [2012]

### 2.1.2 Level of Detail

A prominent aspect of CityGML is that it allows the representation of an object in different Level of Detail (LoD) simultaneously, from LoD0 to LoD4 (Figure 2.2) [Gröger and Plümer, 2012]. LoD0 constitutes the least detailed representation as a 2.5D digital terrain model, which may include building footprints, and LoD1 is a blocks model with flat roof structures, usually obtained by extruding the LoD0 geometries to a height. With LoD2, roof structures are represented more distinctively and boundary surfaces are differentiated semantically such as walls and roofs. LoD3 provides even more detail to add doors and windows to create an architectural model, and finally, LoD4 provides a complete model by including interior elements like rooms, stairs, and furniture [Gröger et al., 2012].

### 2.1.3 Extending CityGML

CityGML can be extended in two ways to define additional features that are missing in the data model for certain applications. The first option is to define new city objects or attributes using



**Figure 2.2:** Geometrical and semantic differentiation of the five LoDs in CityGML.  
Figure from Biljecki et al. [2016]

the Generics module, which includes `GenericCityObject` and `genericAttribute` classes for this purpose. While these classes provide the flexibility to extend CityGML, the fact that new objects and attributes are not defined in an official schema with their own namespaces creates a major drawback, since this may prevent validity checks and lead to interoperability problems [Agugiaro et al., 2018; Biljecki et al., 2018]. The second option for extending CityGML is called an Application Domain Extension (ADE). The use of an ADE is similar to the Generics module in the sense that it is used to define additional classes, attributes, and relations to the data model. However, different from the generic city objects and generic attributes, an ADE has its own formal schema and namespace to define additional features in a systematic manner without causing confusion with the main CityGML elements [Kolbe, 2009]. There are two main rules about how to define new elements in an ADE. If new feature types are created, these must be based on already existing CityGML classes with inheritance relationships [Gröger et al., 2012]. On the other hand, if existing CityGML classes are extended with new attributes and relations, the “hook” mechanism is used, which allows to attach additional properties to existing CityGML classes without having to use the inheritance mechanism [Gröger and Plümer, 2012].

While Biljecki et al. [2018] detected more than 40 ADEs for a large number of applications ranging from robotics to road traffic as of 2018, this number has been growing since then with new applications, such as a support mechanism for metadata in 3D city models [Labetski et al., 2018], keeping relevant information from IFC-sourced 3D city models [Biljecki et al., 2021], and the integration of BIM and environmental planning [Wilhelm et al., 2021]. However, even though ADEs provide a systematic way of extending CityGML, they have their own disadvantages as well. Firstly, ADEs increase the complexity of the data model since elements from different namespaces have to be taken into account [Gröger and Plümer, 2012]. Secondly, special parsers and validators are needed for specific ADEs, and existing software would fail to process additional information coming from an ADE [Ledoux et al., 2019].

## 2.2 CityJSON

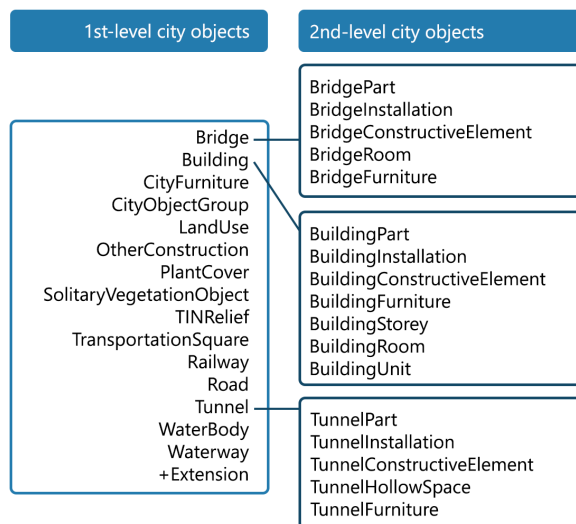
CityJSON is a JSON-based encoding for a subset of the CityGML data model [Ledoux et al., 2019], which has been developed to create an easy-to-use alternative to the official XML-based encoding of CityGML. The first motivation behind this comes from the fact that complex relationships between objects in CityGML are stored as deep hierarchies in the XML-based encoding. This makes it harder to develop software to parse and validate CityGML files, and results in large files that are not human readable [Ledoux et al., 2019]. Moreover, the second motivation concerns the storage of geometries in CityGML. The disadvantage of using the geometry model of GML3 is that the same geometry may be represented in numerous ways with GML3 objects, adding to the complexity of CityGML and making it even more difficult to parse CityGML files, since different variations for each geometry must be supported in the software [Ledoux et al., 2019].



## 2 Related work

The design decision behind CityJSON is to remove the deep hierarchical structure of CityGML and to ensure an efficient way of storing geometries and semantics without leaving any room for ambiguity [Ledoux et al., 2019]. For this, CityGML classes and objects are mapped to CityJSON in a straightforward manner as long as they do not contradict with the main goal of CityJSON, which is to create a compact and user-friendly encoding. If, however, the addition of a CityGML element leads to an increase in complexity, either that object is removed for the sake of simplicity or, and more often, an alternative solution is provided in CityJSON to translate the same concept in a more efficient way. An example to the latter is the Versioning module of CityGML (version 3.0), which is not directly supported by CityJSON arguing its shortcomings, but a Git-based alternative is being developed to provide the same functionality [Vitalis et al., 2019b].

The main differences between CityGML and CityJSON lie in the way objects are defined. First, CityJSON has a simpler structure where city objects are defined as either 1st-level or 2nd-level objects (Figure 2.3). While the former can exist by themselves, the latter must have a relation to a 1st-level object to exist [Dukai and Ledoux, 2021]. The reason of following such a structure is to remove the deep hierarchies of CityGML, while providing the relationship between 1st- and 2nd-level objects implicitly with object properties such as the parent/children relationship [Vitalis et al., 2019a]. In relation to this, some abstract classes are not contained in CityJSON to flatten the hierarchy as much as possible, unless their absence causes structural problems. Second, boundary surfaces such as WallSurface or RoofSurface are defined in a different way in CityJSON than in CityGML. While CityGML defines boundary surfaces as city objects that bound the object in question, CityJSON maps these as semantic objects that are stored separately from the geometry. The aim of this is to decrease the verbosity of CityGML, since each type of semantic object may be then declared only once, and all surfaces with that semantics can refer to it instead of defining the same type of object multiple times [Ledoux et al., 2019]. Finally, even though CityJSON provides ways to bypass the complexities of CityGML, certain aspects of the JSON schema fall behind that of XML. For instance, while CityGML depends heavily on the creation of specialised classes from a general class, this is not directly possible in CityJSON since the JSON schema does not support inheritance. Similarly, CityGML makes use of the namespace concept from the XML schema which allows to add uniquely named attributes to avoid collisions from a different file, or within the same file [Bray et al., 2009]. In CityJSON, this is not possible as well since the JSON schema does not support namespaces. Therefore, validating a CityJSON file might require extra considerations compared to CityGML.



**Figure 2.3:** The hierarchy of CityJSON, structured as 1st- and 2nd-level city objects.  
Figure from Dukai and Ledoux [2021]

### 2.2.1 Extending CityJSON

Despite the fact that it is a rather new encoding, many studies can be found in the literature that make use of CityJSON in various fields, such as automatic building generation using point clouds or deep learning [Nys et al., 2020; Kippers et al., 2021], and urban energy modelling [Prataviera et al., 2021]. Moreover, similar to the ADE concept of CityGML, CityJSON provides an Extension mechanism to add extra elements that are not already included in the core data model. The main difference between the two concepts is that CityGML ADEs are more flexible in the sense that they allow to define different types of objects or new data types so long as they comply with the basic rules behind the ADE concept (see Section 2.1.3). On the other hand, CityJSON restricts the way the core data model can be extended with three possible options [Ledoux et al., 2019]:

1. New (complex) attributes can be added to existing CityObjects.
2. New CityObjects can be created (or existing ones can be extended), and complex geometries can be defined.
3. New properties can be added at the root of the document.

A CityJSON Extension file containing all the elements listed above is demonstrated in Schema 2.1, where the Extension is defined as a separate JSON file with its own URI, and the three possible options to extend the core data model are included as three properties of this file [Ledoux et al., 2019]. It can then be referred to in a CityJSON file when extra elements defined in the Extension are used. The aim of creating such a structure is to eliminate the extra work that needs to be done to process the Extension elements in a CityJSON file. To ensure this, it is recommended for all Extension object names to start with a "+" sign, and the object type must be included.

```

1 {
2   "type": "CityJSONExtension",
3   "name": "Traffic",
4   "description": "Extension to model the traffic",
5   "uri": "https://extensionurl.org/traffic.ext.json",
6   "version": "1.0",
7   "versionCityJSON": "1.1",
8   "extraAttributes": {},
9   "extraCityObjects": {},
10  "extraRootProperties": {}
11 }
```

**Schema 2.1:** A basic CityJSON Extension schema with all possible properties.

It can be discussed that these restrictions and rules help with parsing a CityJSON file that includes extra elements, while putting limitations on the Extension mechanism compared to CityGML ADEs. The fact that the extra elements can only be of three type (extraAttributes, extraCityObjects, extraRootProperties) is a big limitation since any other type of object, such as extra object properties, does automatically not qualify to be included in the Extension. This may restrict the use of CityJSON Extensions for various applications while resulting in the use of CityGML ADEs instead. Moreover, the shortcomings of JSON schema, such as the lack of namespaces and inheritance, also have a negative effect on CityJSON Extensions. For instance, the only way of extending a City Object in the Extension is to create a new one since CityJSON does not support inheritance [Dukai and Ledoux, 2021]. Furthermore, the lack of namespaces may result in confusion about different elements and attributes, especially if a CityJSON file refers to more than one Extension. In this case, different Extensions may contain elements with exactly the same name, which may lead to complications in the parsing process, or general confusion about the data. As a result, it is recommended to prepend all object names with the corresponding extension name to prevent this confusion.

## 2.3 Current CityJSON Extensions

Despite the aforementioned disadvantages of CityJSON Extensions, their benefits over CityGML ADEs can be easily detected as well, such as the fact that software can generally handle CityJSON Extensions without having to change the parsing code [Ledoux et al., 2019]. As a result, numerous CityJSON Extensions can be found in the literature that successfully extend the core data model. These range from the support for 3D point clouds [Nys et al., 2021] to the addition of topological information for city objects [Vitalis et al., 2019a], from storing information on building permits [Wu, 2021] to the representation of data quality in city models [Ilizirov and Dalyot, 2022]. In addition, there are also extensions developed by the CityJSON team to convert the CityGML Noise ADE, and to provide a way to handle GenericCityObjects<sup>1</sup>. In the following paragraphs, the work of Nys et al. [2021] and the CityJSON team to create a Noise Extension<sup>2</sup> will be described.

### 2.3.1 CityJSON Point Cloud Extension

The Point Cloud Extension developed by Nys et al. [2021] supports the storage of 3D point clouds in two ways: either as a link to an external source such as an LAS file, or inline with a MultiPoint geometry. The former is realised by creating a new object called "AbstractCityObjectPC", which has an extra property called "pointcloud-file" to specify the MIME type, URI, and the SRS name of the point cloud file. As a result, all City Objects in CityJSON are defined again as new objects since they must refer to the "AbstractCityObjectPC" element to be able to use the "pointcloud-file" property. An example to this case is given in Schema 2.2<sup>3</sup>, where a tree is represented as a point cloud in LAS format.

```

1 "CityObjects": {
2   "Tree_example": {
3     "type": "+SolitaryVegetationObjectPC",
4     "geometry": [],
5     "pointcloud-file": {
6       "mimeType": "application/vnd.las",
7       "pointFile": "https://github.com/GANys/cityjson-pointcloud/raw/dev/
      example/Tree.las",
8       "referenceSystem": "https://www.opengis.net/def/crs/4326"
9     }
10  }
11 }

```

**Schema 2.2:** An instance of the +SolitaryVegetationObjectPC object, which uses the "pointcloud-file" property.

It can be noticed that the already existing AbstractCityObject element could not be used here since the extension mechanism of CityJSON only allows the addition of extra attributes, but not extra properties. The main difference between these two features is that CityJSON attributes are generally used to specify object-specific characteristics, while properties are utilised to define spatial and relational features, such as the geometry and the parent/children relationships between objects. Furthermore, attributes are stored one level deeper in the hierarchy of a CityJSON object. Then, the second option to store 3D point clouds in the Extension is realised by adding MultiPoint as one of the allowed geometry types of the newly formed City Objects.

<sup>1</sup><https://www.cityjson.org/extensions/>

<sup>2</sup><https://www.cityjson.org/tutorials/extension/>

<sup>3</sup><https://github.com/GANys/cityjson-pointcloud/blob/dev/example/example.city.json>

This extension demonstrates both the strengths and weaknesses of the Extension mechanism of CityJSON. For instance, it is clear that the addition of an extra City Object is highly straightforward since it is one of the allowed operations in the Extension mechanism. On the other hand, extending a City Object with a new property requires finding a way around since this is not directly allowed.

### 2.3.2 CityJSON Noise Extension

The CityJSON Noise Extension<sup>4</sup> is a direct mapping from the Noise ADE, which is defined as an example in the CityGML 2.0 standard to be used in simulations and analyses related to noise pollution [Biljecki et al., 2018]. It extends the core of CityGML by defining new City Objects (Figure 2.4a) and adding new attributes for buildings and building parts (Figure 2.4b).

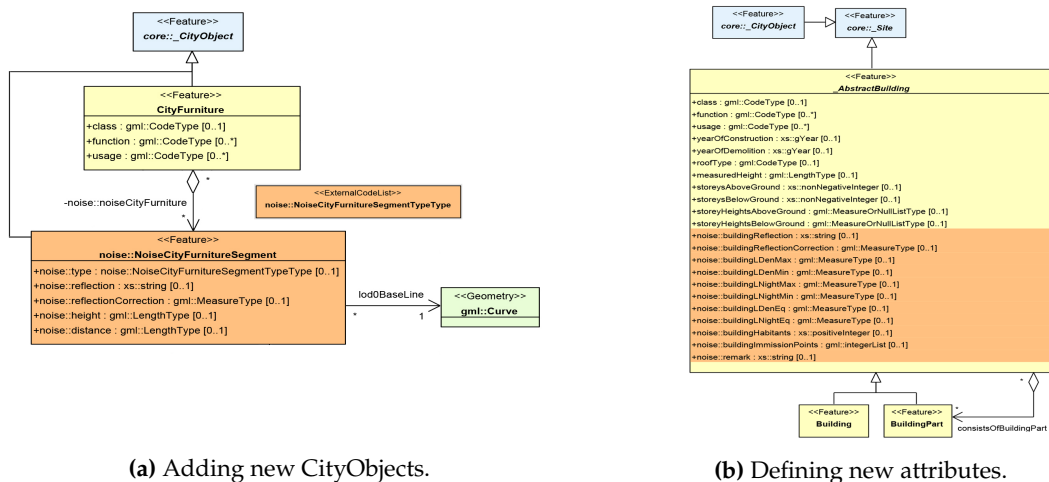


Figure 2.4: UML diagrams of the Noise ADE. Figures adapted from Gröger et al. [2012]

Both of these operations are easily translated to CityJSON by using its Extension mechanism. First, a new City Object called *NoiseCityFurnitureSegment* is defined with its attributes and geometry (Schema 2.3), and the parent/children relationship is used to ensure the connection with the *CityFurniture* object, where each *NoiseCityFurnitureSegment* object has a *CityFurniture* object as a parent. Moreover, it is important to note that the *noise::type* attribute is not directly mapped to the Extension since CityJSON does not support code lists, while the other attributes are mapped with equivalent data types.

<sup>4</sup><https://www.cityjson.org/tutorials/extension/>

```

1 "+NoiseCityFurnitureSegment": {
2   "allOf": [
3     {"$ref": "cityobjects.schema.json#/_AbstractCityObject"},
4     {"properties": {
5       "type": {"enum": ["+NoiseCityFurnitureSegment"] },
6       "attributes": {
7         "properties": {
8           "reflection": {"type": "string"},
9           "reflectionCorrection": {"$ref": "#/definitions/measure"},
10          "height": {"$ref": "#/definitions/measure"}
11        }
12      },
13      "parent": {"type": "string"},
14      "geometry": { ... }
15    }
16    ...
17  }

```

**Schema 2.3:** Definition of the +NoiseCityFurnitureSegment object in the Noise Extension.<sup>5</sup>

Second, new attributes are defined separately for Building and BuildingPart objects since CityJSON does not support inheritance from the AbstractBuilding object. Here, the extraAttributes property is used, and the Extension schema is given in [Schema 2.4](#), where new attributes with several data types are exemplified.

```

1 "extraAttributes": {
2   "Building": {
3     "+noise-buildingReflection": { "type": "string" },
4     "+noise-buildingLNightMax": { "$ref": "#/definitions/measure" },
5     "+noise-buildingHabitants": { "type": "integer" },
6     "+noise-buildingImmissionPoints": {
7       "type": "array",
8       "items": { "type": "integer" } },
9     ...
10  },
11  "BuildingPart": { ... }
12 }

```

**Schema 2.4:** Building CityObject, extended with noise-related attributes.<sup>5</sup>

It can be concluded that the mapping from the Noise ADE to a CityJSON Noise Extension is mostly straightforward since the two operations, adding new CityObjects and attributes, are supported in the Extension mechanism of CityJSON. However, certain design decisions must still be made when it is not possible to map elements due to the disadvantages of the JSON format, or when certain concepts are not supported in CityJSON by choice.

## 2.4 Energy ADE

The Energy ADE (currently on version 1.0) is a data model that extends the CityGML version 2.0 to store and manage energy-related information about buildings [[Agugiaro, 2016a](#)]. Even though the

<sup>5</sup><https://www.cityjson.org/tutorials/extension/>

Building module of CityGML provides certain attributes that can be used in energy applications, such as the year of construction and building class, the fewness and insufficiency of these attributes are two of the main motivations behind the development of such an extension [Benner, 2018]. Therefore, the Energy ADE extends the Core and Building modules of CityGML, and is designed to be used in numerous energy applications, such as the analysis of building elements, energy demand calculations, solar potential studies, and future refurbishment scenarios. The scope of the Energy ADE extends from single-buildings to city-scale assessments, which allows for dynamic simulations with detailed input data, as well as for simplified models [Agugiaro et al., 2018]. As a result, the Energy ADE is designed in 6 thematic modules with different functionalities, each one focusing on a distinct aspect of buildings related to energy applications [Benner, 2018] (Figure 2.5).

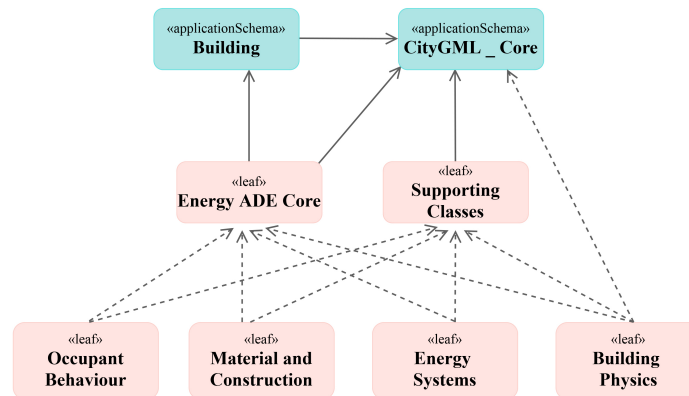


Figure 2.5: The modules and dependencies of the Energy ADE. Figure adapted from Benner [2018]

### 2.4.1 Energy ADE KIT Profile

While the broad scope of the Energy ADE and its detailed modularisation contribute to its use in a wide variety of applications, this comes with the consequence that the data model may be unnecessarily complex for certain simplified applications and simulations. Thus, a subset of the Energy ADE has been created by the Karlsruhe Institute of Technology, called the Energy ADE KIT profile, that excludes the classes and attributes needed only for dynamic simulations, which is consequently easier to use [León-Sánchez et al., 2021]. The main differences between the original Energy ADE data model and the KIT profile are that the Energy Systems module is completely removed and the Supporting Classes module is highly simplified in the KIT profile, while numerous classes and attributes are also discarded in the remaining modules. The following paragraphs explain in more detail each of the KIT Profile modules, since this thesis considers the KIT profile instead of the full Energy ADE.

#### Core module

The Core module extends the AbstractBuilding and CityObject classes of CityGML. The “hook” mechanism is used for this purpose (see Section 2.1.3), where new attributes and relations are attached to the existing classes. Some of the new attributes include the building type, floor area, and volume for buildings, while the new relations are formed to provide a way to store additional information about the weather and energy demand of all city objects. However, it can be discussed that these relations are sometimes irrelevant for certain city objects, since not all types of objects are considered in energy demand or meteorological calculations. For instance, the Relief, WaterBody, and LandUse city objects of CityGML are also extended in the KIT profile with energy demand information, even though these objects are not usually taken into account in energy demand calculations.

## 2 Related work

In addition to the additional attributes and relations, the Core module also defines the abstract base classes for other thematic modules, namely *AbstractThermalZone*, *AbstractUsageZone*, and *AbstractConstruction* [Agugiario et al., 2018]. Finally, numerous data types, enumerations and code lists are provided in the Core module.

### Building Physics module

The Building Physics module includes three classes to define the physical and thermal properties of buildings to be used in energy applications. The first of these is the *ThermalZone* class, which is used to define the smallest unit inside a building with the same thermal behaviour [Benner, 2018]. Moreover, each *ThermalZone* is bounded by a number of *ThermalBoundary* objects that separate different *ThermalZones*, or a *ThermalZone* from the outside environment. Outer walls of a building or the party walls between two adjacent buildings can be given as examples of a *ThermalBoundary*. Finally, each *ThermalBoundary* object may have *ThermalOpenings* such as windows or doors [Benner, 2018]. Each of these objects includes a number of attributes to define their physical properties such as the area, azimuth, inclination, and geometry. In addition, *ThermalBoundary* and *ThermalOpening* objects contain a link to the *AbstractConstruction* object to define more detailed thermal properties such as the U-value, glazing ratio, or reflectance.

### Material and Construction module

The Material and Construction module is used to define the thermal and optical properties of the construction parts of buildings. While a *Construction* can be used by itself to define these properties in a generalised way, each *Construction* object may also be decomposed into *Layers*, where each of the *Layer* objects represents a separate part of the *Construction* with different properties. Furthermore, a *Layer* object consists of one or more *LayerComponent* objects to specify a homogenous part with a specific material [Benner, 2018]. These objects are usually utilised when a high amount of input data is present for more detailed simulations [Agugiario and Holcik, 2017].

### Occupant Behaviour module

The Occupant Behaviour module stores detailed information on different *UsageZones*, which are defined as “the zones of a building with homogeneous usage conditions and indoor climate control settings” [Benner, 2018]. In addition, *Occupants* of a building and the *Facilities* in it are modelled, such as electrical appliances and lighting facilities. Numerous attributes are provided for these objects, such as the current use type, heating and cooling schedules of a *UsageZone*, occupancy rate of *Occupants*, and operation schedule of *Facilities*. These information can then be used in energy demand calculations, since the behaviour of a building’s occupants greatly affects the heat management of a building [Benner et al., 2016].

### Supporting Classes module

The Supporting Classes module focuses on the representation of physical values as time series and schedules. First, the KIT profile contains the *RegularTimeSeries* and *RegularTimeSeriesFile* objects. While the former allows to store a list of measurement values during a specific period with a constant time interval, the latter makes it possible to store the same data on an external file. Second, the KIT profile supports the most frequently used schedule type, *DailyPatternSchedule*, which can be utilised to define specific daily schedules within specific periods in a year [Agugiario et al., 2018].

## 2.5 Space heating demand calculation and 3D city models

Two main approaches can be detected in Urban Energy Modelling for energy demand estimations: *top-down* and *bottom-up* (Figure 2.6). The top-down approaches use aggregated national or regional energy consumption data instead of individual buildings [Ghiassi and Mahdavi, 2017]. Urban scale hypotheses and trends are then built about socio-economic, technical, or physical drivers to disaggregate this data for energy demand calculations [Prataviera et al., 2021]. On the other hand, the bottom-up approaches take input data from individual or groups of houses to calculate energy demand, which are then aggregated for estimations on larger scales [Swan and Ugursal, 2009]. The superiority of one approach over the other depends highly on the application and availability of data, since top-down approaches mostly work with total sum values that lead to lower accuracy, while bottom-up approaches provide more detailed results at the cost of requiring a large amount of data that may not be publicly available in all cases [Ferrando and Causone, 2019].

Within bottom-up approaches, statistical and building simulation (physical) methods are highly used for energy demand estimations. Statistical methods use historical consumption values to determine energy properties of distinct types of buildings. These values are then used to estimate the energy demand of other buildings with similar properties [Kaden and Kolbe, 2014]. Conversely, building simulation (physical) methods use simulation techniques, energy characteristics of buildings, and external data such as climate data to calculate energy demand instead of using already measured consumption data. However, due to the high number of input data requirements, buildings are usually categorised by their properties, such as the typology and year of construction, and the energy demand calculation is performed for each of these categories [Ali et al., 2021].

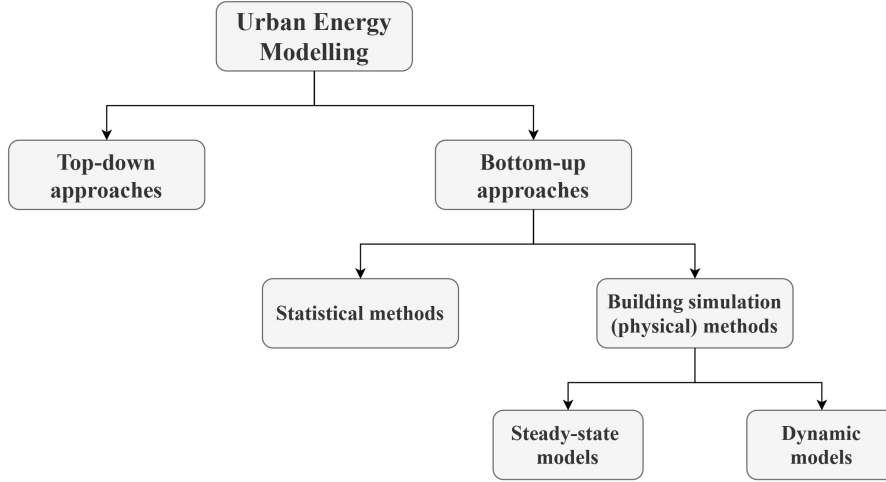
Building simulation methods can be classified as steady-state or dynamic models, depending on the level of detail of the used parameters. In steady-state models, temporal resolution of input data is generally lower for simplification purposes. For instance, climate data such as solar irradiation or outdoor air temperature is usually taken in seasonal or monthly averages [Dalla Mora et al., 2021]. Similarly, fixed values can be used for set-point temperatures since detailed indoor temperature data is mostly not available. Therefore, energy demand values calculated with steady-state models are also presented as monthly or annual values. Contrarily, dynamic models present more detailed results since dynamic effects in buildings, such as the composition of households and the behaviour of dwellers, are taken into account [Verwiebe et al., 2021]. Furthermore, climate data is taken in daily or hourly values, and the resulting energy demand is presented as hourly or daily values as well [Agugiaro et al., 2015]. However, these models are usually preferred when detailed input parameters are available.

### 2.5.1 3D city models and the Energy ADE for space heating demand calculation

The literature suggests that the steady-state models are more frequently used for city-scale space heating demand estimations, since less input parameters are needed and simplifications can be made for the missing information [Agugiaro, 2016b; Skarbal et al., 2017; Pasquinelli et al., 2019; Rossknecht and Airaksinen, 2020]. For instance, Van den Brom [2020] explains the calculation method for space heating demand of buildings in the Netherlands, where the indoor air temperature is taken as a fixed value of 18 °C for simplification purposes. It is important to note that this fixed value is considered for this thesis as well, since the current version of the calculation method does not include a predetermined value for indoor air temperature.

To calculate space heating demand, 3D city models are frequently used for obtaining the input parameters about physical characteristics of buildings, as well as for storing the resulting space heating demand values. Agugiaro [2016b] uses an energy balance method that follows the Italian





**Figure 2.6:** Various approaches in Urban Energy Modelling for energy demand estimations. Figure adapted from Swan and Ugursal [2009]; Kaden and Kolbe [2014]

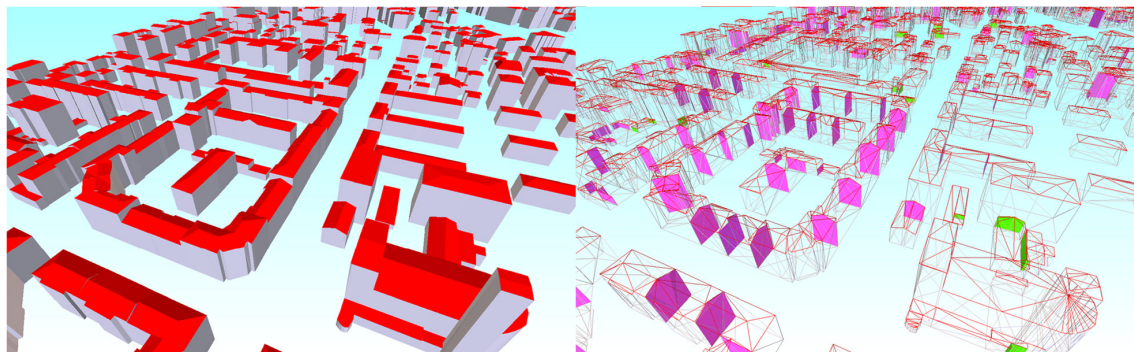
standards to calculate monthly space heating demand values for the city Trento, where many physical parameters such as the area, orientation, and pitch angles of building surfaces are obtained through the 3D city model of the city. Moreover, party walls between topologically adjacent buildings are calculated and stored in the 3D city model as separate geometries (Figure 2.7). On the other hand, detailed parameters such as heat transfer coefficients and window-to-wall ratios are either obtained from existing building physics libraries or fixed values are assumed. This study provides certain input parameters that will be utilised in this thesis. First, the dimensionless reduction factor ( $b_u$ ) for non-adiabatic surfaces, which is given as a fixed value of 0.5, will be used in the calculation of the heat transfer between a heated and unheated space. Second, the formula to compute the air volume flow for the heat losses through ventilation will be adopted in the thesis, which is given in Equation 2.1, where the heated volume is assumed to be the 80% of the whole volume of a building.

$$airvolume\ flow = V_{heated} \cdot n_{ve} \quad (2.1)$$

where

$V_{heated}$  : heated volume of the building, in  $m^3$ ;  
 $n_{ve}$  : air exchange rate, in  $h^{-1}$ .

Kaden and Kolbe [2014] perform a city-wide energy demand estimation for Berlin according to the German standards based on an energy balance method. The 3D city model of the city in CityGML is used to calculate physical and thermal properties of LoD2 buildings, such as the heated volume and energy reference area of surfaces. Furthermore, the final energy demand values are stored in the 3D city model as CityGML attributes. These values are then used for validation by comparing with the already existing energy demand estimations, which have been calculated after visual inspections and manual measurements to obtain physical parameters of buildings. Their results show that the estimated energy demand values using the 3D city model of Berlin varies only 5% from the estimation values obtained with manual calculations [Kaden and Kolbe, 2014], which shows the potential of 3D city models for energy demand calculations.



**Figure 2.7:** Topologically adjacent buildings (left) and the party walls between them (right) as stored in the 3D city model. Figure from Agugiaro [2016b]

Later studies make use of the CityGML Energy ADE to calculate and store the input parameters, which are then used in energy demand simulations. Rossknecht and Airaksinen [2020] use the Energy ADE on both the input and output sides of the calculation. The energy-related input parameters (type of usage zones, number of occupants, heating status, etc.) are first stored in the already existing 3D city model in CityGML with the Energy ADE. Then, these parameters are used for energy demand simulations, and the resulting monthly space heating demand values are stored with the *EnergyDemand* class of the Energy ADE for each building.

Similarly, Skarbal et al. [2017] use the Energy ADE to store additional data about buildings to be used in the energy demand calculation. While some of this data is calculated directly from the 3D city model, such as the footprint area and gross volume, the data coming from external sources is integrated as well, such as the solar irradiance values and building typologies. Therefore, it can be concluded that the Energy ADE is utilised successfully to store heterogeneous energy-related data from various sources in a single 3D city model.

### 2.5.2 Current energy simulation software with 3D city models

There are numerous energy simulation tools available with distinct focuses and considerations. For instance, certain tools such as EnergyPlus [Crawley et al., 2001], TRNSYS [Beckman et al., 1994] and SIM-VICUS<sup>6</sup> are generally used for performing detailed simulations on building level. These tools, therefore, require highly detailed and precise input data about the building, such as the geometry, building physics parameters, and HVAC equipments, to be able to run the simulation. On the other hand, certain other software, such as SimStadt and CitySim, can be used for energy simulations on city scale instead of focusing on single buildings. Since a city scale energy demand calculation is within the scope of this thesis, detailed information about SimStadt and CitySim is provided in the following paragraphs.

Firstly, SimStadt was developed at HFT Stuttgart to help authorities make energy-related decisions by performing energy simulations, such as solar and PV potential analysis and energy demand simulations [Scartezzini et al., 2015]. The software accepts CityGML files as input data, while the missing additional information, such as building physics parameters and weather data, can be retrieved from SimStadt’s pre-built libraries [León-Sánchez et al., 2021]. Alternatively, SimStadt accepts additional data provided by the user, as long as the data is created in a compatible format for the software to read and process. Furthermore, SimStadt uses a steady-state method based on the German standard DIN V 18599 for automatically calculating monthly energy demand of buildings

<sup>6</sup><https://www.sim-vicus.de/>

## 2 Related work

[Monien et al., 2017]. The data in the pre-built libraries of SimStadt is therefore based on the German regulations that specify the desired building characteristics.

Secondly, CitySim uses a dynamic method based on an RC model to simulate the energy flows of buildings [Robinson et al., 2009]. In contrast to SimStadt, CitySim requires the input data to be in a specific format (CitySim XML file format), which can be produced using the provided GUI with CityGML files [León-Sánchez et al., 2021]. In addition, CitySim requires the user to provide additional information such as weather and building physics data, instead of automatically using pre-built libraries. Finally, CitySim differs from SimStadt in terms of the temporal resolution of output values. While SimStadt provides monthly simulation values, CitySim produces values in a higher temporal resolution, at hourly intervals, since it uses a dynamic method, which is known to provide more detailed results [León-Sánchez et al., 2021]. The hourly simulation values can then be aggregated into monthly or yearly values depending on the needs of the user.



# 3 Methodology

This section presents the details of the methodology used in this thesis. As shown in Figure 3.1, the methodology consists of two interrelated parts: the development of the CityJSON Energy Extension, and the space heating demand calculation. In the rest of this section, the main parts of the methodology are explained in detail. Firstly, Section 3.1 explains the main considerations in creating a semi-direct translation from the Energy ADE to a CityJSON Energy Extension. Secondly, the validation process and the used tools are described in Section 3.2. Then, the calculation method for the use case, space heating demand of buildings, is described, and the needed input parameters are determined in Section 3.3. Finally, Section 3.4 explains the methods used to improve the initial CityJSON Energy Extension to make it more efficient and fully compatible with the use case.

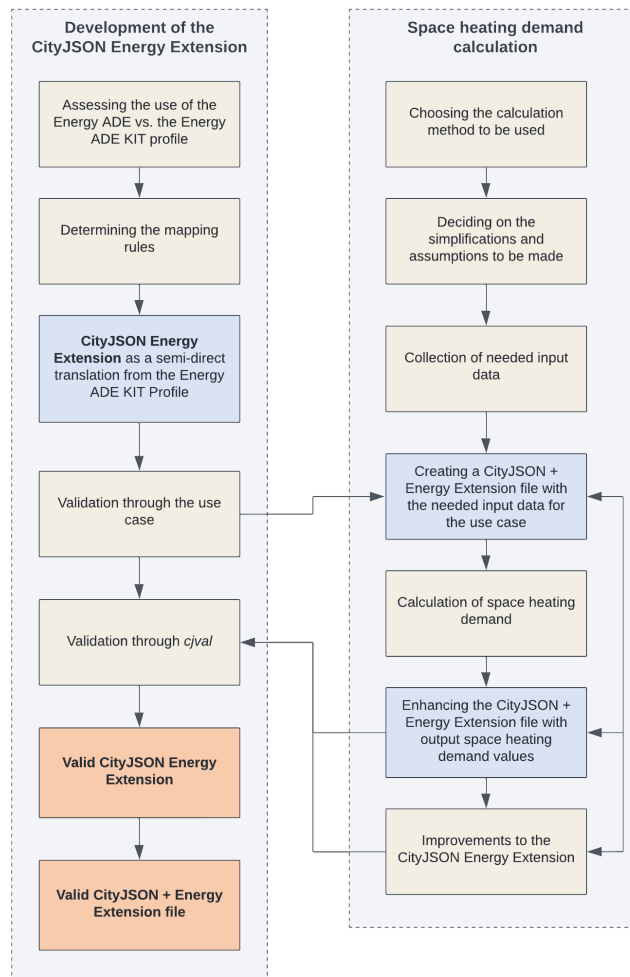


Figure 3.1: Overview of the methodology. Grey: main steps, blue: intermediate outputs, orange: final output of the thesis.

### 3.1 Semi-direct translation from the Energy ADE

The first step of the methodology involved creating a semi-direct translation from the Energy ADE to a CityJSON Energy Extension to explore the possibilities, and to grasp the similarities and differences between the CityGML and CityJSON data models. For this, an assessment was made to decide whether to use the full Energy ADE or the Energy ADE KIT profile considering the use case. Based on the previous studies where the Energy ADE is used for steady-state space heating demand calculations [Agugiario, 2016b; Rossknecht and Airaksinen, 2020], it was decided that the full Energy ADE provides a much more complex option than needed for the use case, therefore, the Energy ADE KIT profile was decided to be used throughout this thesis.

While the initial idea was to develop a fully-direct translation in this step, the literature review presented in Chapter 2 showed the restricted extension mechanism of CityJSON, which made it clear that a fully-direct translation is not possible in the case of the Energy ADE in general. Therefore, all types of objects and attributes defined in the Energy ADE KIT profile were considered to determine the mapping rules, and an overview of each of these elements and the main considerations are described in the following paragraphs.

#### 3.1.1 New City Objects

In the Energy ADE KIT profile, new city objects are defined as subclasses of the abstract *\_CityObject* class of CityGML. Therefore, all attributes and relations defined for *\_CityObject* are inherited by the new Energy ADE KIT profile city objects as well. An example is given in Figure 3.2 for the UsageZone class, where an *AbstractUsageZone* is first created as a subclass of *\_CityObject*, inheriting its four attributes: *creationDate* and *terminationDate* to keep track of the feature’s history, and *relativeToTerrain* and *relativeToWater* to determine the position of the feature [Gröger et al., 2012]. Then, the UsageZone class itself is defined with its own attributes while inheriting additional ones from its superclass. As a result, while determining the mapping rules for the new city objects in the Energy ADE KIT profile, their own attributes as well as the inherited ones from their abstract classes or the *\_CityObject* class itself were considered.

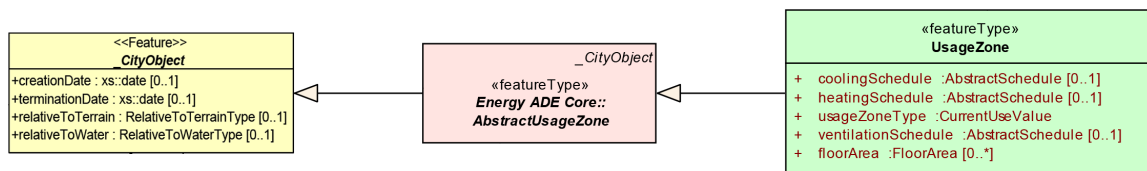


Figure 3.2: UsageZone object defined as a City Object in the Energy ADE KIT profile. Figure adapted from Gröger et al. [2012]; Benner [2018]

#### 3.1.2 New attributes to existing City Objects with additional data types

The Energy ADE KIT profile extends the existing *AbstractBuilding* class of CityGML with additional energy-related attributes using the “hook” mechanism, as described in Section 2.4.1. For the semi-direct translation, additional mapping rules were determined to discover the possibilities to use such a mechanism in the CityJSON Energy Extension.

Furthermore, as shown in Figure 3.3, new data types, enumerations, and codelists are also defined besides the simple types such as *CharacterString* or *Decimal*. While the data types and enumerations are created directly in the schema, codelists are defined by referring to well-known standard

codelists with their URLs. Since these additional data types are used extensively throughout the Energy ADE KIT profile for all types of classes, they were considered carefully in the mapping rules for the CityJSON Energy Extension.

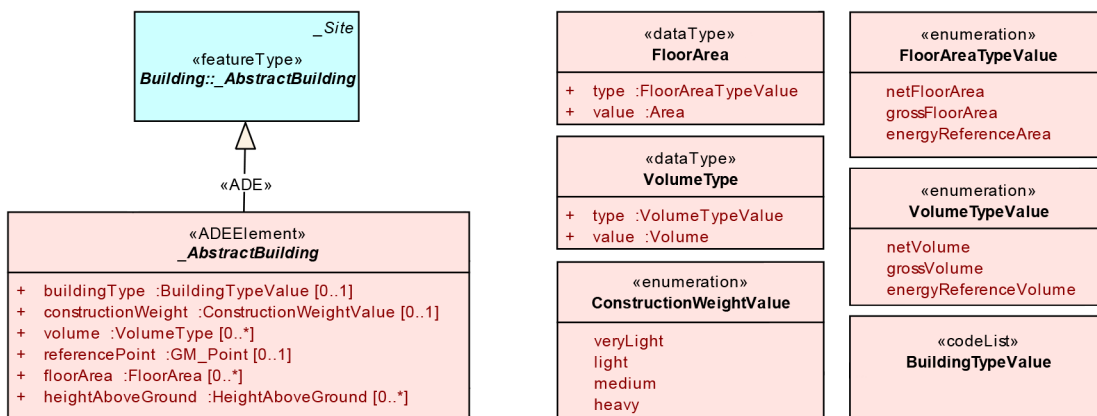


Figure 3.3: The “hook” mechanism of the Energy ADE KIT profile to define new attributes (left), and the newly defined data types, enumerations, and code lists (right). Figure adapted from Benner [2018]

### 3.1.3 New non-City Objects

In addition to the new City Objects, new non-City Objects are created in the Energy ADE KIT profile, which do not inherit the attributes and relations that *AbstractCityObject* contains. In the UML model (Figure 3.4), these objects are defined with either `«featureType»` or `«type»` stereotypes, which correspond to the subtypes of *gml:AbstractFeatureType* or *gml:AbstractGMLType* respectively [Portele, 2007]. Therefore, the attributes of these types are inherited by the non-City Objects in the KIT profile, such as *gml:name*, *gml:identifier*, and *gml:description*. While City Objects are themselves derived from *gml:AbstractFeatureType*, and therefore inherit the same attributes, non-City Objects do not inherit from *\_CityObject* class. As a result, extra attention was paid to these types of objects when determining the mapping rules.

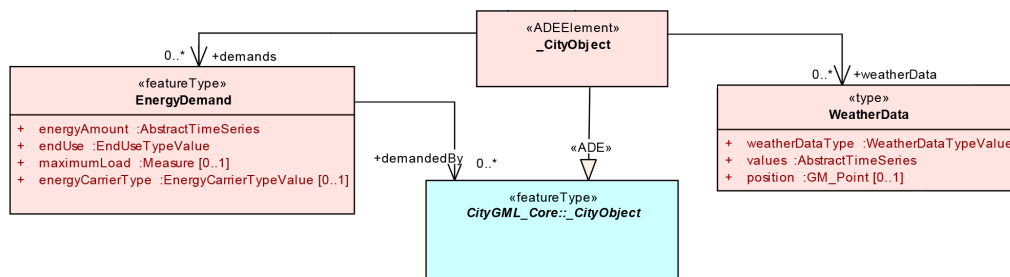


Figure 3.4: Non-City Objects defined in the Energy ADE KIT profile. Figure adapted from Benner [2018]

### 3.1.4 Relations among City Objects and non-City Objects

The associations between City Objects and non-City Objects are formed with the same mechanism in the Energy ADE KIT profile, where additional attributes are defined to store the *gml:identifier* of the objects that they have a relation with (Figure 3.4). Moreover, this mechanism considers multiplicity since some objects may be related to only one object, while others may be associated with numerous objects. These aspects of the relations among City Objects and non-City Objects were taken into consideration in the determination of mapping rules, while also taking a critical side to create a CityJSON Energy Extension that conforms to the design decisions behind the CityJSON data model itself.

## 3.2 Validation of the CityJSON Energy Extension

After creating a semi-direct translation from the Energy ADE KIT profile to a CityJSON Energy Extension, the second step of the methodology involved the validation of the Extension. While three possible validation methods were detected, only the first two were used in this thesis. The following paragraphs give an overview of each of the three methods, including the reasons behind using them or not.

### 3.2.1 Validation through the use case

The first validation method considered the use case, space heating demand calculation, since the CityJSON Energy Extension was developed directly for this application. To validate the semi-direct translation, a CityJSON + Energy Extension file was created, where the input data needed for the use case was stored with the energy-related Extension objects and attributes. During this step, it was evaluated whether all input data could be stored in the CityJSON Energy Extension in a straightforward way or not. This was done by comparing all required input data against the objects and attributes defined in the CityJSON + Energy Extension file, and any data that could not be stored with the predefined objects or attributes was marked as a shortcoming of the semi-direct translation to be improved later. Moreover, special attention was paid to distinct data types and whether these are compatible with the input data. If, however, incompatibilities were detected, these were marked to be improved later in the process as well.

### 3.2.2 Validation through *cjval*

During, and after, the validation of the Extension with the use case, *cjval*<sup>1</sup>, the official validator of CityJSON, was used as the second validation method. This validator provides various functionalities for CityJSON files with or without an Extension by checking their validity against the official JSON syntax, CityJSON schemas, Extension schemas (if provided), and other geometric checks (Table 3.1). In the case of the CityJSON Energy Extension, the CityJSON + Energy Extension file created in the previous step was validated to check for any inconsistencies. Furthermore, the additional functionality called *cjvalext* was used to validate the schema of the CityJSON Energy Extension against the official extension schema of CityJSON.

---

<sup>1</sup><https://github.com/cityjson/cjval>



<i>Cjval</i> function	What it does
JSON syntax	Checks the file against the JSON schema
CityJSON schemas	Checks the file against CityJSON schemas ( v1.1)
Extension schemas	If an Extension is present in the input file, it is validated against the Extension schema of CityJSON
Parent/children consistency	Checks whether the referenced parents/children actually exist
Wrong vertex index	Vertices list and their indices are checked
Semantics array	Checks if semantics of a geometry are added correctly
Extra root properties	Gives a warning if an extra root property is not defined in an Extension
Duplicate vertices	Gives a warning if there are duplicate vertices
Unused vertices	Gives a warning if there are unreferenced vertices

Table 3.1: Validation functionalities of *cjval* <sup>1</sup>

### 3.2.3 Validation against the Energy ADE KIT profile

The third, and final, validation method that could be used for the CityJSON Energy Extension was to validate it against the Energy ADE KIT profile. This could be done by developing a program to convert the CityJSON + Energy Extension file back to a CityGML + Energy ADE KIT profile file. This way, it could be checked whether this is a lossless conversion, or whether certain data is lost in the translation due to distinct data types or different ways of storing information. While this is an effective validation method as well, it was decided not to be used for this thesis. The main reason of this was that the validation step through the use case showed that the semi-direct translation was not enough to fully support the use case, since certain classes and attributes were missing to store particular input data, and these would have to be included in the final version of the CityJSON Energy Extension. Considering that the newly added classes and attributes could not be translated back to a CityGML + Energy ADE KIT profile file in a straightforward way, the validation against the Energy ADE KIT profile was not considered in the thesis.

## 3.3 Space heating demand calculation

Space heating demand calculation of buildings was chosen as the use case to be benefited from in every step of the development of the CityJSON Energy Extension, such as the validation, testing, and improvements. In the following paragraphs, the details of the calculation method are given, and the required input data is explained.

### 3.3.1 Calculation method

Space heating demand calculation was performed based on the determination method specified by the Dutch standard called *NTA 8800*. The aim of this standard is to provide a determination method to calculate the energy performance of buildings with a focus on the energy requirement, primary fossil energy use, renewable energy, and the net heat demand of a building [Royal Netherlands Standardization Institute, 2022].

### 3 Methodology

The calculation is based on a steady-state energy balance method, which considers heat losses and heat gains with monthly average values of input parameters, while the dynamic effects are included with utilisation factors [Radwan, 2021]. Accordingly, the net monthly energy demand of a calculation zone for space heating ( $Q_{H;nd;zi;mi}$ ) is calculated with Equation 3.1, where the left side of the formula represents the heat losses through transmission ( $Q_{H;tr;zi;mi}$ ) and ventilation ( $Q_{H;ve;zi;mi}$ ), and the right side illustrates the heat gains through internal ( $Q_{H;int;zi;mi}$ ) and solar ( $Q_{H;sol;zi;mi}$ ) gains multiplied by a dimensionless utilisation factor ( $n_{H;gn;zi;mi}$ ) to consider dynamic effects.

$$Q_{H;nd;zi;mi} = (Q_{H;tr;zi;mi} + Q_{H;ve;zi;mi}) - n_{H;gn;zi;mi} (Q_{H;int;zi;mi} + Q_{H;sol;zi;mi}) \quad (3.1)$$

#### Heat losses through transmission

The heat losses through transmission represent the total heat transfer through the building envelope elements due to the difference in temperature. In *NTA 8800*, this is calculated with Equation 3.2, in kWh,

$$Q_{H;tr;zi;mi} = \left( H_{H;tr(excl.gf;m);zi;mi} (\theta_{int;calc;H;zi;mi} - \theta_{e,avg;mi}) + H_{g;an;zi;mi} (\theta_{int;calc;H;zi;mi} - \theta_{e,avg;an}) \right) \cdot 0.001 \cdot t_{mi} \quad (3.2)$$

where

- $H_{H;tr(excl.gf;m);zi;mi}$  : total heat transfer coefficient through transmission, except for the ground floor, in  $W/K$ , calculated in Equation 3.3;
- $\theta_{int;calc;H;zi;mi}$  : temperature of the calculation zone, in  $^{\circ}C$ ;
- $\theta_{e,avg;mi}$  : average outside temperature in month  $mi$ , in  $^{\circ}C$ ;
- $H_{g;an;zi;mi}$  : heat transfer coefficient for building elements in contact with the ground, such as basements, in  $W/K$ , calculated in Equation 3.7;
- $\theta_{e,avg;an}$  : average outdoor temperature for the entire year, in  $^{\circ}C$ ;
- $t_{mi}$  : length of the considered month, in  $h$ .

The first element of this calculation, the total heat transfer coefficient through transmission, except for the ground floor, is calculated with Equation 3.3,

$$H_{H;tr(excl.gf;m);zi;mi} = H_{H;D;zi;mi} + H_{H;U;zi;mi} + H_{H;A;zi;mi} + H_{H;p;zi} \quad (3.3)$$

where

- $H_{H;D;zi;mi}$  : direct heat transfer coefficient between the heated space and the outside air, in  $W/K$ ;
- $H_{H;U;zi;mi}$  : heat transfer coefficient through adjacent unheated spaces, in  $W/K$ ;
- $H_{H;A;zi;mi}$  : heat transfer coefficient through adjacent heated spaces, in  $W/K$ ;
- $H_{H;p;zi}$  : heat transfer coefficient through vertical pipes, in  $W/K$ .

Firstly, the direct heat transfer coefficient between the heated space and the outside air is calculated with Equation 3.4, which considers the area ( $A_{T;i}$ ) and U-value ( $U_{C;i}$ ) of the building envelope elements that are in direct contact with the outside air:

$$H_{H;D;zi;mi} = \sum_i (A_{T;i} \times (U_{C;i} + \Delta U_{for})) \quad (3.4)$$

### 3 Methodology

while taking into account linear thermal bridges with Equation 3.5:

$$\Delta U_{for} = \max \left[ 0; 0.1 - 0.25 \times \left( \frac{\sum_i (A_{T;i} \times U_{C;i})}{\sum_i A_{T;i}} - 0.4 \right) \right] \quad (3.5)$$

Secondly, the heat transfer coefficient through adjacent unheated spaces is calculated for the building elements that are in contact with an adjacent unheated space. Since these surfaces are not directly in contact with the outside air, the heat transfer coefficient is first calculated with Equation 3.4, then, this value is multiplied with a dimensionless reduction factor ( $b_u$ ). Thirdly, the heat transfer coefficient through adjacent heated spaces is considered. However, according to NTA 8800, the heat transfer from a heated space to another heated space is neglected [Royal Netherlands Standardization Institute, 2022]. Therefore, this value is considered 0 in the rest of the calculation. Finally, the heat transfer coefficient through vertical pipes is calculated with Equation 3.6,

$$H_{H;p;zi} = \sum_j N_{bouwlaag;j} \cdot H_{H;p;spec;j} \quad (3.6)$$

where

- $j$  : number of vertical pipes in the calculation zone;
- $N_{bouwlaag;j}$  : number of storeys of the calculation zone in which vertical pipe  $j$  is located;
- $H_{H;p;spec;j}$  : heat transfer coefficient for vertical pipe  $j$ , in  $W/K$ , fixed at 1.8.

Next, the second element in the calculation of heat losses through transmission is computed with Equation 3.7, which is the heat transfer coefficient for building elements in contact with the ground:

$$H_{g;an;zi;mi} = A_{T;fl} \times U_{fl} \times 0.5 \times P \quad (3.7)$$

where

- $A_{T;fl}$  : area of the floor, in  $m^2$ ;
- $U_{fl}$  : heat transfer coefficient of the floor, in  $W/(m^2 \cdot K)$ ;
- $P$  : perimeter of the floor, in  $m$ .

#### Heat losses through ventilation

The heat losses through ventilation consider various types of air flows entering the calculation zone such as through infiltration and natural/mechanical ventilation. According to NTA 8800, for each calculation zone  $zi$  and month  $mi$ , the total heat loss through ventilation is computed with Equation 3.8, in  $kWh$ ,

$$Q_{H;ve;zi;mi} = \left( p_a \cdot c_a \cdot \sum_k (q_{v;k;H;zi;mi} \cdot b_{v;k;H;zi;mi} \cdot f_{v;dyn;k;zi;mi}) / 3600 \right) \cdot (\theta_{int;calc;H;zi} - \theta_{e;avg;mi}) \cdot 0.001 \cdot t_{mi} \quad (3.8)$$

where

### 3 Methodology

- $p_a$  : air density, fixed at  $1.205 \text{ kg/m}^3$ ;  
 $c_a$  : specific heat capacity of air, fixed at  $1005 \text{ J/kgK}$ ;  
 $q_{v;k;H;zi;mi}$  : air volume flow  $k$ , in  $\text{m}^3/\text{h}$ ;  
 $b_{v;k;H;zi;mi}$  : supply temperature correction factor for air volume flow  $k$ , fixed at 1;  
 $f_{v;dyn;k;zi;mi}$  : dynamic correction factor for air volume  $k$ , fixed at 1;  
 $\theta_{int;calc;H;zi}$  : temperature of the calculation zone, in  $^\circ\text{C}$ ;  
 $\theta_{e;avg;mi}$  : average outside temperature in month  $mi$ , in  $^\circ\text{C}$ ;  
 $t_{mi}$  : length of the considered month, in  $h$ .

#### Internal heat gains

Internal heat gains concerns the contribution of internal sources, such as the activity of occupants and the utilisation of electrical devices, to the heat management of a building [Cerný and Kocí, 2015], which is calculated with Equation 3.9, in  $\text{kWh}$ ,

$$Q_{H;int;zi;mi} = 180 \cdot N_{woon;zi} \cdot N_{P;woon;zi} \cdot 0.001 \cdot t_{mi} \quad (3.9)$$

where

- 180 : average heat production per person, in  $W$ ;  
 $N_{woon;zi}$  : number of residential units in the calculation zone  $zi$ ;  
 $N_{P;woon;zi}$  : average number of residents per calculation zone per residential unit, calculated in Equation 3.10;  
 $t_{mi}$  : length of the considered month, in  $h$ .

Accordingly, the average number of residents per calculation zone per residential unit ( $N_{P;woon;zi}$ ) is computed based on the average usable area of the calculation zone ( $A_{g;zi}$ ) as follows:

$$\begin{aligned}
 A_{g;zi}/N_{woon;zi} \leq 30\text{m}^2 : N_{P;woon;zi} &= 1 \\
 30\text{m}^2 < A_{g;zi}/N_{woon;zi} \leq 100\text{m}^2 : N_{P;woon;zi} &= 2.28 - 1.28/70 \times \left(100 - \frac{A_{g;zi}}{N_{woon;zi}}\right) \\
 A_{g;zi}/N_{woon;zi} > 100\text{m}^2 : N_{P;woon;zi} &= 1.28 + 0.01 \times \frac{A_{g;zi}}{N_{woon;zi}}
 \end{aligned} \quad (3.10)$$

#### Solar gains

Solar gains of a building involves the heat gain from incident solar radiation through both transparent and non-transparent elements of the building [Royal Netherlands Standardization Institute, 2022], which is calculated with Equation 3.11, in  $\text{kWh}$ ,

$$Q_{H;sol;zi;mi} = \sum_k Q_{H;sol;wi;k,mi} + \sum_k Q_{H;sol;op;k,mi} \quad (3.11)$$

where

- $Q_{H;sol;wi;k,mi}$  : solar heat gain through transparent element  $wi, k$  for month  $mi$ , in  $\text{kWh}$ ;  
 $Q_{H;sol;op;k,mi}$  : solar heat gain through non-transparent element  $op, k$  for month  $mi$ , in  $\text{kWh}$ .

### 3 Methodology

First, the solar heat gain through transparent elements ( $Q_{H;sol;wi,k,mi}$ ) is calculated with Equation 3.12, in kWh,

$$Q_{H;sol;wi,k,mi} = g_{gl;wi,k;H;mi} \cdot A_{wi,k} \cdot (1 - F_{fr;wi,k}) \cdot F_{sh;obst;wi,k;mi} \cdot I_{sol;wi,k;mi} \cdot 0.001 \cdot t_{mi} - Q_{sky;wi,k;mi} \quad (3.12)$$

where

- $g_{gl;wi,k;H;mi}$  : average effective total solar gain factor of window  $wi, k$  for month  $mi$ ;
- $A_{wi,k}$  : area of window  $wi, k$ , in  $m^2$ ;
- $F_{fr;wi,k}$  : frame fraction of window  $wi, k$ , fixed at 0.25;
- $F_{sh;obst;wi,k;mi}$  : shading reduction factor for external obstacles of window  $wi, k$  in month  $mi$ ;
- $I_{sol;wi,k;mi}$  : average total incident solar radiation per  $m^2$  of window  $wi, k$  in month  $mi$ , in  $W/m^2$ ;
- $t_{mi}$  : length of the considered month, in  $h$ ;
- $Q_{sky;wi,k;mi}$  : extra heat flow due to heat radiation to the sky from window  $wi, k$  in month  $mi$ , in kWh, calculated in Equation 3.13.

Accordingly, the extra heat flow due to heat radiation to the sky ( $Q_{sky;wi,k;mi}$ ) is calculated with Equation 3.13,

$$Q_{sky;wi,k;mi} = 0.001 \cdot F_{sky;k} \cdot R_{se;k} \cdot U_{c;k} \cdot A_{c;k} \cdot h_{lr,e;k} \cdot \Delta\theta_{sky;mi} \cdot t_{mi} \quad (3.13)$$

where

- $F_{sky;k}$  : visibility factor between the building envelope element  $k$  and the sky;
- $R_{se;k}$  : heat transfer resistance on the outside of element  $k$ , in  $(m^2K)/W$ ;
- $U_{c;k}$  : heat transfer coefficient of element  $k$ , in  $W/(m^2K)$ ;
- $A_{c;k}$  : area of the element  $k$ , in  $m^2$ ;
- $h_{lr,e;k}$  : heat transfer coefficient for long wave radiation on the outside of the construction, fixed at  $4.14 W/(m^2K)$ ;
- $\Delta\theta_{sky;mi}$  : average difference between the apparent sky temperature and the outdoor temperature, fixed at  $11 K$ ;
- $t_{mi}$  : length of the considered month, in  $h$ .

Second, the solar heat gain through non-transparent elements ( $Q_{H;sol;op,k,mi}$ ) is calculated with Equation 3.14, in kWh,

$$Q_{H;sol;op,k,mi} = \alpha_{sol} \cdot R_{se} \cdot U_{c;op,k} \cdot A_{c;op,k} \cdot F_{sh;obst;op,k;mi} \cdot I_{sol;op,k;mi} \cdot 0.001 \cdot t_{mi} - Q_{sky;op,k;mi} \quad (3.14)$$

where

### 3 Methodology

$\alpha_{sol}$	: dimensionless absorption coefficient for solar radiation, fixed at 0.6;
$R_{se}$	: heat transfer resistance on the outside, in $(m^2K)/W$ ;
$U_{c;op,k}$	: heat transfer coefficient of non-transparent element $op, k$ , in $W/(m^2K)$ ;
$A_{c;op,k}$	: area of non-transparent element $op, k$ , in $m^2$ ;
$F_{sh;obst;op,k;mi}$	: shading reduction factor for external obstacles of non-transparent element $op, k$ , fixed at 1;
$I_{sol;op,k;mi}$	: average total incident solar radiation per $m^2$ of non-transparent element $op, k$ in month $mi$ , in $W/m^2$ ;
$t_{mi}$	: length of the considered month, in $h$ ;
$Q_{sky;op,k;mi}$	: extra heat flow due to heat radiation to the sky from non-transparent element $op, k$ in month $mi$ , in $kWh$ , calculated in Equation 3.13, where the variable $wi$ is replaced with variable $op$ .

#### 3.3.2 Required input data for space heating demand calculation

As the complexity of the calculation method from the *NTA 8800* standard suggests, the space heating demand calculation of buildings requires a large amount of input data, which must be obtained from various types of data sources. An overview of the main elements of this calculation, as well as the needed input data, is presented in Table 3.2. However, since this calculation was done on city-scale and not for single buildings, it was not possible to obtain all needed data for each individual building. Therefore, simplifications and assumptions were made to modify the calculation method and to be able to perform the calculation for the whole study area. Moreover, the methods for collecting the data and the used datasets will be explained in detail in Chapter 4.

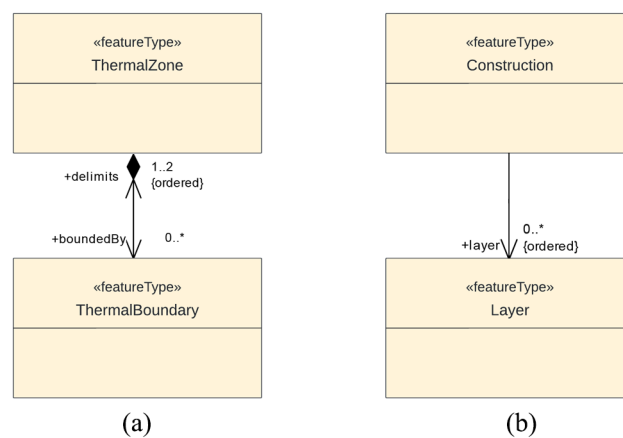
Element	Explanation	Needed input data
<b>Heat losses through transmission</b> ( $Q_{H;tr;zi;mi}$ )	Considers the heat transfer (1) between the heated space and outside air, (2) via adjacent unheated spaces, (3) via adjacent heated spaces, (4) through vertical pipes in direct contact with outside air	- Area and U-value of surfaces - Number of storeys of the building - Perimeter of the calculation zone - Indoor/outdoor temperature - Length of the month
<b>Heat losses through ventilation</b> ( $Q_{H;ve;zi;mi}$ )	Considers the ventilation air that enters the calculation zone and causes heat losses	- Volume of effective air flow - Indoor/outdoor temperature - Length of the month
<b>Internal heat gains</b> ( $Q_{H;int;zi;mi}$ )	Considers the internal heat produced by occupants and the facilities located in the calculation zone	- Number of residential units in the calculation zone - Average usable area of the calculation zone - Length of the month
<b>Solar gains</b> ( $Q_{H;sol;zi;mi}$ )	Considers the on-site solar radiation, orientation, sun absorption and heat transfer properties of the receiving transparent and opaque surfaces	- Area of the building element - Frame fraction of windows - g-values of windows - U-value of the building element - Monthly average solar radiation - Heat transfer resistance - Length of the month

**Table 3.2:** Elements of the space heating demand calculation according to *NTA 8800*, and the needed input data.

### 3.4 Improvements on the CityJSON Energy Extension

The final step of the methodology focused on improving the semi-direct translation that was created in the first step. These improvements were mainly designed based on the results obtained from the validation process, which considered the validity of the Extension schema as well as the testing through the use case. As a result, the improvements focused on four different mechanisms/aspects of the Extension, which are summarised below:

- **Providing full support for the use case:** As the main objective of this thesis is to develop a CityJSON Energy Extension for the calculation of space heating demand of buildings as the use case, the main consideration for improvements was to ensure that the Extension fully supports the use case. The results of the validation process were taken into account for this, and new classes and attributes were added when certain data could not be stored in the semi-direct translation in an efficient way, which was assessed by comparing against the design decisions behind CityJSON, such as avoiding a hierarchical structure as much as possible.
- **Relations between objects:** In the Energy ADE KIT profile, the relations between different classes are ensured with additional properties to store object IDs. In CityJSON, on the other hand, the parent/children mechanism is used to relate objects to one another. To improve the semi-direct translation, the possibilities to use the two concepts were examined, and the decision of using additional properties or the parent/children structure was based on the type of relation between objects and the resulting hierarchy. In other words, the main consideration in this step was to ensure that the final relation between two objects do not result in a deep hierarchy, in which the data can only be reached after going down several steps in the hierarchy. In addition, the direction of relations between objects was considered, such as unidirectional and bi-directional relations (Figure 3.5). Since unidirectional relations might result in duplicate information stored in both objects on the two ends of a relation, these were modified to avoid unnecessarily complex associations between objects.



**Figure 3.5:** Example of (a) unidirectional and (b) bi-directional relations. Figure adapted from [Benner, 2018]

- **Ease of retrieving data from the Extension file:** To ensure that the CityJSON Energy Extension provides an efficient way of both storing and retrieving (the stored) data, the next step in improvements focused on the number of links/hierarchy to pass to reach a specific data during the space heating demand calculation. If this number was higher than a given threshold, modifications were made in classes, attributes, or data types to store the data with less hierarchies.

### *3 Methodology*

- **Naming conventions in the Extension schema:** The schema of the semi-direct translation was modified to establish consistency in the names of classes, attributes, and data types of the CityJSON Energy Extension. The structure of names as well as the use of uppercase and/or lowercase letters were checked, and a uniform framework was created. Furthermore, a prefix was determined for the CityJSON Energy Extension to avoid confusion with objects from other extensions or that are added outside of schema.





## 4 Study area and datasets

This section presents the study area and the used datasets for the space heating demand calculation. First, the characteristics of the study area are introduced in [Section 4.1](#). Then, the collection of input data for the calculations is explained in [Section 4.2](#). In addition, necessary pre-processing steps and the used software are described in this section.

### 4.1 Study area

To calculate the space heating demand of buildings, a study area was determined from the city of Rijssen, within the Rijssen-Holten municipality of the Netherlands ([Figure 4.1](#)). This area was preferred for the use case because of the availability of input data coming from a previous study and ongoing research, which will be described in [Section 4.2](#).

The study area consists of 3318 buildings with different functionalities ([Figure 4.2a](#)). While 1918 buildings in the area have a residential function, 180 buildings have a mixed-use, and 1220 buildings are used for non-residential purposes. Moreover, it is possible to find various typologies in the area for residential buildings, such as single-family houses, multi-family houses, apartment blocks, and terrace houses ([Figure 4.2b](#)). Since the space heating demand of a building is highly related to the type of construction, this variety in typologies was considered an advantage to analyse and compare the final energy demand values in the area.



**Figure 4.1:** The location of the Rijssen-Holten municipality in the Netherlands (a), and the study area with used buildings (b), both shown in red.

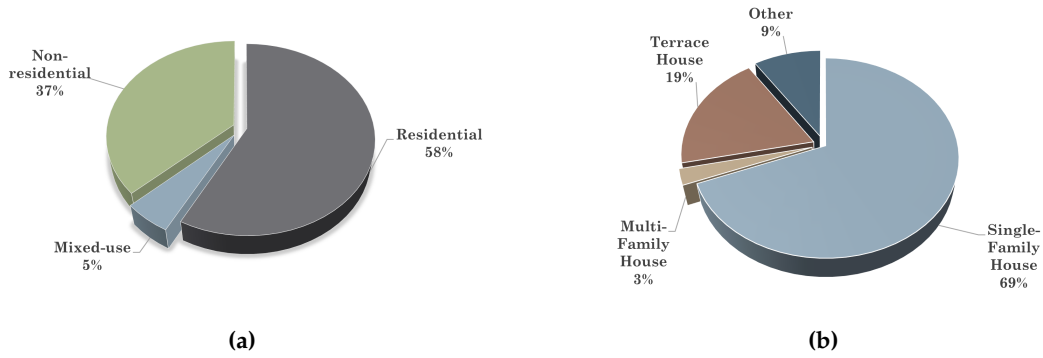


Figure 4.2: Distribution of (a) the building functionalities, and (b) the building typologies in the study area.

## 4.2 Collection of input data for space heating demand

Due to the complexity of space heating demand calculation, required input data was obtained from various datasets and previous studies for the study area of Rijssen-Holten in the Netherlands. In this section, first, an overview of the input parameters needed for each part of the calculation, as well as the corresponding data sources, are presented in Table 4.1. Then, each of the data sources is explained, together with the necessary pre-processing steps and the used software.

Element	Input parameters	Data source
<b>Heat losses through transmission</b>	Area of surfaces	3D city model
	U-value of surfaces	TABULA library
	Number of storeys	3D city model
	Perimeter of the building	3D city model
	Indoor air temperature	Van den Brom [2020]
	Outdoor air temperature	Meteorological data portal
	Length of the month	NTA 8800 standard
	Dimensionless reduction factor	Aguiaro [2016b]; Kaden and Kolbe [2014]
<b>Heat losses through ventilation</b>	Building volume	3D city model
	Air exchange rate	TABULA library
<b>Internal heat gains</b>	Number of residential units	BAG dataset
	Usable area in the building	BAG dataset
<b>Solar gains</b>	g-value of windows	TABULA library
	Area of windows	TABULA library (window ratio)
	Shading reduction factor	NTA 8800 standard
	Solar radiation	NTA 8800 standard
	Inclination of surfaces	3D city model
	Slope of surfaces	3D city model
	Visibility factor	NTA 8800 standard
Heat transfer resistance	NTA 8800 standard	
<b>Additional needed parameters</b>	Utilisation factor	TABULA library
	Construction year of buildings	3D city model
	Building typology	3D city model

Table 4.1: Input parameters of the space heating demand calculation and their data sources.

### 4.2.1 3D city model of Rijssen-Holten

In this thesis, the 3D city model of Rijssen-Holten is used as the main data source to acquire the required input data for space heating demand calculation. The 3D city model is a testbed for energy applications, currently under development at the 3D Geoinformation Group at TU Delft. In this model, geometric, semantic, and topological properties of 3318 buildings in the study area are stored in XML-based CityGML format (Figure 4.3). While 3290 of these are stored as single-part buildings as CityGML *Building* objects, 9 multi-part buildings are present as well in the dataset, resulting in extra 19 buildings modelled as CityGML *BuildingParts*.

The buildings are available with two types of geometries: an LoD0 geometry to represent the building footprints, and an LoD2 geometry which includes a more detailed representation of the building with its semantic surfaces. In the latter, each building's surfaces are modelled as *BoundarySurfaces* of CityGML, namely *WallSurface*, *RoofSurface*, and *GroundSurface*. In addition to this classification, the *WallSurfaces* between topologically adjacent buildings are instead modelled as *ClosureSurface* objects to easily differentiate the party walls from the rest. Due to the complex nature of the calculation and the dependency on highly detailed parameters, the LoD2 geometries are used in this thesis for obtaining data from, and storing energy-related information in the 3D city model.

Each building contains a number of attributes to provide information about its specific properties. While some of these are added with existing CityGML attributes from the Building module (e.g. year of construction, class, function), additional generic attributes are used to include the type of properties that can be used in energy applications (e.g. number of adjacent buildings, volume). Similarly, each *BoundarySurface* is enriched with energy-related generic attributes, such as the area, azimuth, and orientation of the surface. An overview of all *Building* and *BoundarySurface* attributes are presented in Table 4.2.



Figure 4.3: The 3D city model of the study area: a subset of Rijssen-Holten in the Netherlands.

#### Pre-processing of the 3D city model

A number of pre-processing steps were taken to prepare the dataset to be used in space heating demand calculation. First, the 3D city model was used to calculate two required parameters, namely the perimeter of buildings and the slope of all surfaces, to be used in Equation 3.7 and Equation 3.11, respectively. This was done on a Safe Software FME Workbench, where, for each building, the

	Attribute	Description
Building	Net internal area	Excludes internal structural elements
	Class	Type of use of the building, e.g. residential, mixed-use
	Function*	Further description of the class, e.g., health, business
	Usage	Whether the building is still in use
	Measured height	Height of the building, in <i>m</i>
	Relative to terrain	Whether the building is (entirely) above or below the terrain
	Roof type	E.g. slanted, single/multiple horizontal
	Year of construction	Construction year of the building
	Footprint area*	Footprint area, calculated from the LoD0 geometry, in <i>m</i> <sup>2</sup>
	Storeys above ground*	Number of storeys situated above ground level
	Storeys below ground*	Number of storeys situated below ground level
	Building name*	Unique name of the building
	Is single part	Boolean value to show whether the building has <i>BuildingParts</i>
	# of adjacent buildings	Number of topologically adjacent buildings
	LoD2 volume*	Building volume, calculated from the LoD2 geometry, in <i>m</i> <sup>3</sup>
	LoD max	Maximum LoD present for the building
Building (pand) ID	Unique ID of the building	
List adjacent buildings	Building (pand) ID of topologically adjacent buildings	
BoundarySurface	Surface ID	Unique ID of the <i>BoundarySurface</i>
	Parent building ID	Building (pand) ID of the building that the surface belongs to
	Surface name	Unique name of the <i>BoundarySurface</i>
	Azimuth	Azimuth of the surface, in <i>degrees</i>
	Inclination	Inclination of the surface, in <i>degrees</i>
	Direction	Direction of the surface
	LoD2 area	Surface area, calculated from the LoD2 geometry, in <i>m</i> <sup>2</sup>
	Surface normal	Normal vector of the surface

**Table 4.2:** The attributes stored in the 3D city model of Rijssen-Holten. If the object is a *BuildingPart*, the marked attributes (\*) are stored in the *BuildingPart*, while the rest is stored in the parent *Building* object.

necessary calculations were done and the resulting data was integrated back to the 3D city model as generic attributes.

Second, the 3D city model was converted from CityGML to CityJSON using *citygml-tools*<sup>1</sup>, a command line utility for processing CityGML files. Since *citygml-tools* currently supports CityJSON up to version 1.0.3, the file was then upgraded to the latest version, 1.1.0, with *cjio*<sup>2</sup>, which is a Python command-line interface to process and manipulate CityJSON files. In this step, an information loss was detected in the conversion between the two formats due to a limitation of CityJSON. While each building may have a number of functions in CityGML format, only the first of these

<sup>1</sup><https://github.com/citygml4j/citygml-tools>

<sup>2</sup><https://github.com/cityjson/cjio>

functions was stored in the *function* attribute of Building objects in CityJSON, since the core data model does not support complex attributes, such as array types to store multiple values. To solve this problem, an additional function attribute was created in the CityJSON Energy Extension to be able to store multiple functions in an array.

Finally, the resulting CityJSON file was examined to detect any inconsistencies in geometries or attributes. It was discovered that 57 buildings out of 3318 did not have an LoD2 geometry, but were modelled only by LoD0 geometries. Since energy-related generic attributes were not available for these buildings, they were marked to be omitted in the calculation. In addition, it was found that the parent Building objects of *BuildingParts* did not have any geometry at all. This was due to a design choice, according to which the parent Building object is used to store only general attributes that correspond to all children *BuildingParts*, while the *BuildingPart* objects themselves include the geometry and specific attributes. Therefore, the parent Building objects were marked as well to be neglected in the calculation. Furthermore, when the areas of *BoundarySurfaces* were examined, it was seen that some buildings include considerably small *WallSurfaces*, such as an area of 0.2 - 2 m<sup>2</sup>, which might be caused by errors during the construction of the 3D city model. Since these values might result in wrong assumptions or overestimations in solar gains calculation, these *WallSurfaces* were marked to be handled differently during the calculation.

### 4.2.2 Basisregistratie Adressen en Gebouwen (BAG)

The BAG is a key register for all addresses and buildings in the Netherlands, and the dataset can be freely accessed from the website of Kadaster [Kadaster, 2022], which is the land registry agency of the Netherlands. The BAG dataset contains detailed information about buildings, the most important of which for this thesis are the usable area and the number of residential units in a building. Therefore, this information was extracted from the dataset for each building in the study area, and used in the calculation of internal gains in a building (Equation 3.9).

#### Pre-processing of the BAG dataset

In the BAG dataset, each residential unit in a building is modelled with a point geometry, which refers to the Building (pand) ID that it belongs to. A number of additional attributes are available for these objects, such as the construction year, address, usable area, and its function (Figure 4.4 & Table 4.3). Since the total number of residential units and usable area per building are needed in the calculations, the residential unit objects were aggregated per building ID and the sum of usable areas was calculated. In this step, some inconsistencies were detected in the BAG dataset. First, it was discovered that 199 from the total 3318 buildings in the study area were not present in the BAG dataset. Since the usable area and number of residential units data were not available for these buildings, they were omitted during the calculations. Moreover, while the required data was aggregated per building, *BuildingPart* objects were not considered since these are not modelled in the BAG. Therefore, an assumption was made to distribute the data among the *BuildingPart* objects with a residential function, such that the usable area and number of residential units were assigned to the *BuildingPart* with the largest volume, while the others were not considered for the calculation.



**Figure 4.4:** Residential units as points.  
Figure from the PDOK viewer

Attribute	Value
Year of construction	1957
Function	Residential
House number	18
Object ID	174200000013387
Street name	Van den Broekestraat
Usable area	134
Building (pand) ID	1742100000005824
Building status	Building in use
Postcode	7462VR
Status	Residential unit in use
Place	Rijssen

**Table 4.3:** The available attributes of the selected residential unit (in blue).

### 4.2.3 Meteorological Data Portal

The outside air temperature data of Rijssen-Holten was obtained from the Meteorological Data Portal, which was created by the Photovoltaic Materials and Devices (PVMD) group at TU Delft<sup>3</sup>. This portal uses the measurements from the Royal Netherlands Meteorological Institute (KNMI), and weather data of a Dutch province as well as of a specific location in the Netherlands can be obtained. Furthermore, the data is presented in two resolutions: weather data for the current date, and the climate data for 1 year. The former includes measurements every 10 minutes for a day, and the latter contains measurements of an average year with a 1 hour resolution.

#### Pre-processing of the outside weather data

Since the space heating demand calculation was done in a monthly resolution for the whole year, the climate data for 1 year was obtained from the Meteorological Data Portal. The closest weather station to Rijssen-Holten was chosen, called *Heino*, which is located approximately 30 kilometres north-west of Rijssen-Holten. Then, the hourly data was averaged for each month, as well as for the whole year, to be used in the heat losses calculations (Equation 3.2 & Equation 3.8).

### 4.2.4 TABULA Building Physics Library

For the space heating demand calculation, detailed data on the building physics properties of each building must be known, such as the thermal transmittance of all surfaces and the g-value of windows. While this data may be easily obtained for a single-building, it is not possible to collect detailed information about all the surfaces of all building on a city-scale calculation. Therefore, in this thesis, the building physics library developed as a result of the European projects TABULA and EPISCOPE was used. The aim of these projects was to categorise buildings in each European country by their year of construction and building typology. Then, detailed building physics properties were presented for each category instead of single buildings [TABULA, 2012].

For the Netherlands, buildings are categorised in 4 main building typologies (single-family house, multi-family house, terraced house, apartment block). In addition, a more detailed classification is provided on top of the 4 main typologies, such as detached, semi-detached, middle row,

<sup>3</sup><https://www.tudelft.nl/?id=59090&L=1>

## 4 Study area and datasets

and end house. However, only the main 4 building typologies are used in this thesis because of the availability of data only for this classification. Furthermore, buildings are further categorised in 6 construction periods (till 1964, 1965 – 1974, 1975 – 1991, 1992 – 2005, 2006 – 2014, 2015 – today). Example building types for the 4 main typologies and the first five construction periods can be found in Figure 4.5.

The TABULA library provides data on three main elements of a building, which can be summarised as follows:

- **Windows:** Eight distinct window types are defined, and their U-value, g-value, and frame ratio are stored in the library.
- **Construction elements:** Different construction materials, such as brick, stone, and concrete, are defined, and their physical properties, such as density, heat capacity, conductivity, and thickness, are stored.
- **Building category:** This part of the library provides detailed building physics properties for each type of surface (e.g. outer walls, ground and roof surfaces, party walls) of each building category (based on the typology and construction period). These properties include the U-value, short-wave reflectance, window ratio, as well as the window and construction IDs to refer to the above-mentioned parts of the library for more detailed parameters. Moreover, building-specific properties are included as well, such as the average storey height, infiltration rate, and thermal bridge U-values. Finally, for each building category, the same properties are included for a medium and advanced refurbishment scenario.


























Country	Region	Construction Year Class	Additional Classification	SFH Single Family House	TH Terraced House	MFH Multi Family House	AB Apartment Block
	national (nationaal)	... 1964	generic (generiek)	 NL.N.SFH.01.Gen	 NL.N.TH.01.Gen	 NL.N.MFH.01.Gen	 NL.N.AB.01.Gen
	national (nationaal)	1965 ... 1974	generic (generiek)	 NL.N.SFH.02.Gen	 NL.N.TH.02.Gen	 NL.N.MFH.02.Gen	 NL.N.AB.02.Gen
	national (nationaal)	1975 ... 1991	generic (generiek)	 NL.N.SFH.03.Gen	 NL.N.TH.03.Gen	 NL.N.MFH.03.Gen	 NL.N.AB.03.Gen
	national (nationaal)	1992 ... 2005	generic (generiek)	 NL.N.SFH.04.Gen	 NL.N.TH.04.Gen	 NL.N.MFH.04.Gen	 NL.N.AB.04.Gen
	national (nationaal)	2006 ... 2014	generic (generiek)	 NL.N.SFH.05.Gen	 NL.N.TH.05.Gen	 NL.N.MFH.05.Gen	 NL.N.AB.05.Gen

Figure 4.5: TABULA building classification for the Netherlands. Figure from TABULA [2012]

### Pre-processing of the TABULA library

The raw data coming from the TABULA library for the Netherlands was stored in XML format, which, because of its hierarchical structure, made it challenging to retrieve the needed information for each building category. Therefore, we created a database to store all the data coming from the TABULA library as a joint effort with Yuzhen Jin, who is currently writing his thesis and also makes use of the TABULA library, and Camilo León-Sánchez, who is the first supervisor of this thesis. The aim of this was to be able to retrieve data more easily and store information in a more structured way, which may also be beneficial for future studies in this field. For space heating demand calculation,



#### 4 Study area and datasets

U-values, window ratios, and window g-values were obtained to be used in the heat transmission losses and solar gains parts of the calculation (Equation 3.2 & Equation 3.11).

##### 4.2.5 NTA 8800

The solar radiation, shading reduction and visibility factors, and the length of the month parameters are already provided in the *NTA 8800* standard, therefore, these values were used in the calculation. First, the solar radiation and visibility factor values are provided for different orientations and inclinations. Second, the shading reduction factor is provided for different orientations and slopes of surfaces. Finally, the length of the month parameter is provided in the *NTA 8800* standard for each month. These input data were then stored in the database that was created before for the TABULA library to ensure easy access to the data.



## 5 Implementation

This section presents the implementation details of the methodology. First, the mapping rules and preliminary design decisions are described in [Section 5.1](#) to create a semi-direct translation from the Energy ADE KIT profile to a CityJSON Energy Extension. Second, experiments on the storage of input data in the CityJSON Energy Extension are explained in [Section 5.2](#), followed by a detailed description of the space heating demand calculation with pre-determined assumptions and simplifications ([Section 5.3](#)). Finally, [Section 5.4](#) presents the improvements on the semi-direct translation to create the final CityJSON Energy Extension. The schema of the CityJSON Energy Extension as well as the Python script to calculate space heating demand is available at: <https://github.com/ozgetufan/cjenergy>

### 5.1 Mapping rules of the semi-direct translation

Since a fully-direct translation is not possible for the Energy ADE KIT profile, as presented in [Chapter 2](#), a semi-direct translation was created, which follows the structure of the Energy ADE KIT profile as much as possible, and includes new mapping rules and preliminary design decisions when this is not possible. The main mapping rules are explained in the following paragraphs, and examples from the Extension schema, as well as the corresponding exemplary JSON objects are given from the Core module of the KIT Profile [[Benner, 2018](#)]. However, not only the Core module elements but the whole Energy ADE KIT profile was translated with the specified mapping rules.

#### 5.1.1 Creating new City Objects

CityJSON's Extension mechanism allows the addition of new CityObjects with their attributes by using the "extraCityObjects" member of the Extension ([Schema 5.1](#)). Therefore, this functionality was used to map the new CityObjects in the Energy ADE KIT profile in a straightforward way. However, a preliminary design decision was made for the inheritance relationship between all CityObjects and the *AbstractCityObject* class of CityGML, since inheritance is not supported in CityJSON. To ensure a similar mechanism, the "allOf" keyword of the JSON schema was used, which ensures that a schema is also valid against all of the given subschemas <sup>1</sup>. For instance, the *AbstractCityObject* defined in CityJSON includes a number of properties. By using the "allOf" keyword in the schema of new CityObjects defined in the Extension, it is made sure that these properties can be directly defined for new CityObjects without facing validity problems.

In addition, it was decided not to map the new abstract CityObjects defined in the Energy ADE KIT Profile. The reason of this is that the main philosophy of CityJSON is to define objects without deep hierarchies [[Ledoux et al., 2019](#)]. To ensure this, the core data model of CityJSON itself eliminates certain abstract classes as long as their absence does not cause systematic problems. A similar check was performed, and it was seen that in the Energy ADE KIT profile, the abstract base classes, such as *AbstractUsageZone* and *AbstractThermalZone*, do not include any properties to pass on to their subclasses. Therefore, the mapping of the abstract CityObjects were deemed to be unnecessary, and they were excluded from the semi-direct translation.

<sup>1</sup><https://json-schema.org/understanding-json-schema/reference/combining.html>

```

1  "extraCityObjects": {
2    "+UsageZone": {
3      "allOf": [
4        {"$ref": "cityobjects.schema.json#/_AbstractCityObject"},
5        {
6          "properties": {
7            "attributes": {
8              "type": "object",
9              "properties": {
10               "usageZoneType": {
11                 "type": "string"
12               },
13               "floorArea": {...}
14             }
15           },
16           ...
17         }
18       ]
19     }
20   }
21 }

```

```

"Usage1": {
  "type": "+UsageZone",
  "attributes": {
    "usageZoneType": "residential",
    "floorArea": {
      "type": "netFloorArea",
      "value": {
        "value": 100,
        "uom": "m2"
      }
    }
  }
}

```

**Schema 5.1:** Definition of the UsageZone CityObject in the Extension schema (left), and the exemplary JSON object with example data (right).

### 5.1.2 New attributes to existing City Objects

CityJSON's Extension mechanism allows the addition of new attributes to existing CityObjects with the "extraAttributes" member, which supports both simple and complex attributes with varied data types. Therefore, all new attributes defined in the Energy ADE KIT profile for the existing CityObjects (*\_AbstractBuilding* and *\_CityObject*) were defined with this method. An example is given in [Schema 5.2](#), where the existing *Building* CityObject is extended with new attributes as defined in the Energy ADE KIT Profile.

```

1  "extraAttributes": {
2    "Building": {
3      "+buildingType": {...},
4      "+constructionWeight": {...},
5      "+volume": {...},
6      "+floorArea": {...},
7      "+heightAboveGround": {...}
8    }
9  }

```

```

"Build1": {
  "type": "Building",
  "geometry": [...],
  "attributes": {
    "+buildingType": "singleFamily",
    "+constructionWeight": "heavy",
  }
}

```

**Schema 5.2:** Extra attributes defined for the Building CityObject (left), and how it is implemented for a Building object called *Build1* (right).

### 5.1.3 Creating new non-City Objects

The Energy ADE KIT profile contains not only new CityObjects, but also new classes that are not derived from the *\_CityObject* class of the core data model, but from various other stereotypes, such as «featureType». However, a direct mapping of these classes to the CityJSON Energy Extension was not possible, since the Extension mechanism of CityJSON only supports the addition of new objects in the form of CityObjects. Therefore, a preliminary design decision had to be made in this stage, and it was decided to create non-CityObjects under the "extraCityObjects" member, together with the

## 5 Implementation

newly defined CityObjects. Since non-CityObjects do not inherit attributes and relationships from the `_CityObject` class, these objects were defined without using the "allOf" keyword to reference `AbstractCityObject` from the core data model of CityJSON. An example to this is given in [Schema 5.3](#), where the `EnergyDemand` class that is derived from `gml:AbstractFeatureType` is defined with the "extraCityObjects" keyword.

```
1 "extraCityObjects": {
2   "+EnergyDemand": {
3     "type": "object",
4     "properties": {
5       "type": {...},
6       "attributes": {
7         "type": "object",
8         "properties": {
9           "energyAmount": {...},
10          "endUse": {...},
11          "maximumLoad": {...},
12          "energyCarrierType": {...}
13        }
14      }
15    }
16  }
```

```
"Demand1": {
  "type": "+EnergyDemand",
  "attributes": {
    "endUse": "spaceHeating",
    "energyCarrierType": "naturalGas",
    "energyAmount": "...", //ID of
                          TimeSeries object
    "maximumLoad": {
      "value": 250,
      "uom": "kWh/m2"
    }
  }
}
```

**Schema 5.3:** `EnergyDemand`, a non-CityObject, defined with the "extraCityObjects" keyword (left), and an exemplary JSON object with the corresponding attributes (right).

Similarly, all objects with the `<<type>>` stereotype are derived from `gml:AbstractGMLType`, such as the classes of Time Series, Schedule and Weather Data in the Energy ADE KIT Profile, and were therefore mapped as "extraCityObjects" as well ([Schema 5.4](#)). Even from this early stage, it could be argued that defining CityObjects and non-CityObjects together with one keyword was not the most ideal solution, since this could result in confusion about the defined objects and their properties. However, since this restriction is related to the Extension mechanism of CityJSON, an alternative solution was not possible.

```
1 "extraCityObjects": {
2   "+WeatherData": {
3     "type": "object",
4     "properties": {
5       "type": {...},
6       "attributes": {
7         "type": "object",
8         "properties": {
9           "weatherDataType": {...},
10          "values": {...},
11          "position": {...}
12        }
13      }
14    }
15  }
```

```
"OutdoorTemperature": {
  "type": "+WeatherData",
  "attributes": {
    "weatherDataType": "airTemperature",
    "values": "RegularTimeSeries1",
              //ID of TimeSeries object
  }
},
"RegularTimeSeries1": {
  "type": "+RegularTimeSeries",
  "attributes": {
    "values": [2.61, 4.82, 5.91, 9.32,
              14.73, 16.12],
    ...
  }
}
```

**Schema 5.4:** `WeatherData` object, defined with the "extraCityObject" keyword in the Extension schema, and an exemplary JSON object, as well as the referenced Time Series object (right).

### 5.1.4 New data types, enumerations and code lists

The Energy ADE KIT Profile defines specific data types, enumerations, and code lists to be used as the allowed values of certain attributes. These additional types are defined individually and separate from their corresponding classes, since a data type, enumeration, or code list may be used in multiple classes. A similar approach was used for the semi-direct translation, and data types were defined as subschemas under the “definitions” keyword of the JSON schema. This way, defined data types could be referenced multiple times with a JSON pointer (Schema 5.5) instead of having duplicate information in the Extension schema.

After defining data types, enumerations were defined with the “enum” keyword of the JSON schema, which is used to restrict the value of an attribute to a fixed set of options<sup>2</sup>. However, unlike data types defined as subschemas, enumerations could only be defined inside a JSON object. Therefore, if an enumeration was used for multiple attributes in the Energy ADE KIT profile, these were created for each attribute individually, resulting in duplicate information in the Extension schema.

```

1  "definitions": {
2    "floorArea": {
3      "type": "object",
4      "properties": {
5        "type": {
6          "enum": ["netFloorArea",
7                  "grossFloorArea",
8                  "energyReferenceArea"]
9        },
10     "value": {...}
11   }
12 }

```

```

"extraAttributes": {
  "Building": {
    "+floorArea": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/floorArea"
      }
    },
    ...
  }
}

```

**Schema 5.5:** floorArea data type defined as a subschema (left), and referenced as the data type of the floorArea attribute of Buildings (right) in the schema.

Unlike data types and enumerations, code lists were not mapped as part of the semi-direct translation, since these values may not be fixed in advance, but may be determined by the responsible authorities. Therefore, code lists in the Energy ADE KIT profile were simply mapped as *string* types.

### 5.1.5 Relations among City Objects and non-City Objects

The associations among CityObjects and non-CityObjects were handled as two separate cases in the semi-direct translation. First, the relations between two CityObjects were implemented with the “parents/children” concept of JSON. For instance, in the Energy ADE KIT profile, a ThermalZone may contain 0 or more UsageZones [Bernner, 2018]. To ensure this relationship in the semi-direct translation, parents and children properties were defined for the two objects to specify the ID(s) of the corresponding parent or children object (Schema 5.6).

Second, the relations between a CityObject and non-CityObject, or between two non-CityObjects, were designed with additional attributes instead of the “parent/children” concept. In these cases, the additional attributes were added to the corresponding objects, and were used to store the IDs of related objects. For instance, in the Energy ADE KIT profile, all CityObjects may have energy

<sup>2</sup><https://json-schema.org/understanding-json-schema/reference/generic.html>

demand, specified with the "+demands" relation between the CityObject and the EnergyDemand object [Benner, 2018]. To ensure this relation in the semi-direct translation, an additional attribute called "energyDemand" was added to all CityObjects to store the IDs of EnergyDemand objects that they have a relation with (Schema 5.7).

```

1 "+ThermalZone": {
2   "type": {...},
3   "properties": {
4     "parents": {
5       "type": "array",
6       "items": {"type": "string"}
7     },
8     "children": {
9       "type": "array",
10      "items": {"type": "string"}
11    }
12  }
13 }

"Zone1": {
  "type": "+ThermalZone",
  ...,
  "children": ["Usage1"]
},
"Usage1": {
  "type": "+UsageZone",
  "attributes": {
    "usageZoneType": "residential"
  },
  "parents": ["Zone1"],
  ...
}

```

**Schema 5.6:** ThermalZone object with its parents and children properties (left), and the use of this concept to form a relation with a UsageZone object (right).

```

1 "extraAttributes": {
2   "Building": {
3     "+buildingType": {...},
4     "+energyDemand": {
5       "type": "array",
6       "items": {"type": "string"}
7       //ID of EnergyDemand objects
8     }
9   }
10 }

"Build1": {
  "type": "Building",
  "geometry": [...],
  "attributes": {
    "+buildingType": "singleFamily",
    ...
    "+energyDemand": ["Demand1"]
  }
}

```

**Schema 5.7:** The relation between a Building and EnergyDemand object, formed with an additional attribute "+energyDemand".

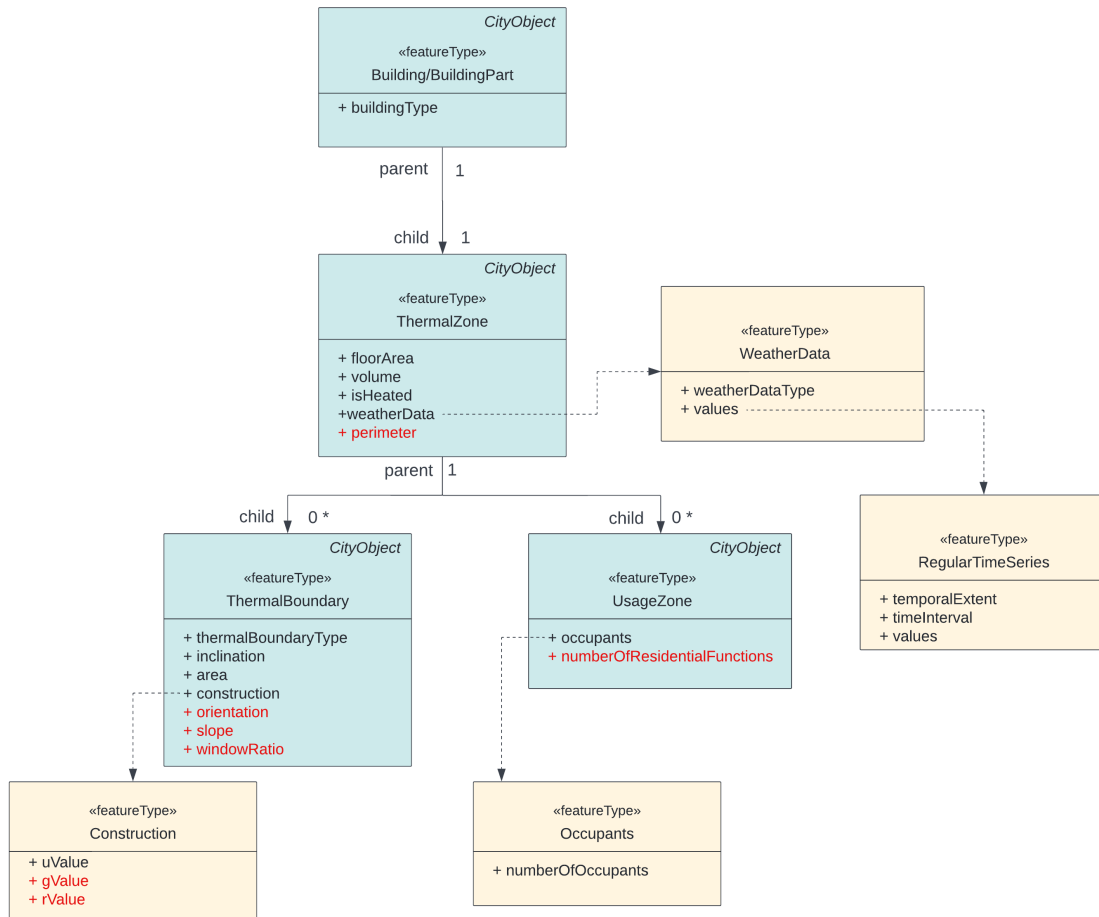
The purpose of using separate mechanisms for CityObjects and non-CityObjects was to differentiate the two types of objects by defining different functionalities regarding associations. This way, the use and understanding of the Extension schema, as well as the objects used in a CityJSON + Energy Extension file, could be made more straightforward.

## 5.2 Experiments on the storage of input data in the CityJSON Energy Extension

After all the required input data was collected from numerous data sources, this data was then stored in a CityJSON + Energy Extension file, which was created in the first step as a semi-direct translation. The aim of this step was to make experiments on how to store the input data with the CityJSON Energy Extension, and whether this process is straightforward or not. If it was discovered during this process that certain data could not be stored in the Extension because of a lack of certain attributes, these attributes were created without being defined in the Extension schema to make it

## 5 Implementation

possible to continue with the space heating demand calculation in the next step. However, these out-of-schema attributes were noted to be properly defined in the final version of the CityJSON Energy Extension. In the following paragraphs, it is described how each input data was stored in the Extension, and Figure 5.1 demonstrates the hierarchy of CityJSON Energy Extension objects after the input data was stored.



**Figure 5.1:** The hierarchy of CityJSON Energy Extension objects to store the needed input data. The unbroken lines represent relationships with parent/children mechanism, and the dashed lines show relationships with additional attributes. The red attributes represent the input data that was stored with out-of-schema attributes in the semi-direct translation.

As the first step, the typology of each Building and BuildingPart was stored in a straightforward way with the extra buildingType attribute, defined for these objects in the Extension. Then, for each Building/BuildingPart, a ThermalZone object was created, and the relationship between them was ensured with the parent/children mechanism, where each Building/BuildingPart has a ThermalZone child. The floorArea, volume, and isHeated attributes of ThermalZone objects were used to store the corresponding data. However, the perimeter of the calculation zone could not be directly stored, since the Energy ADE KIT profile does not include this as an attribute of ThermalZones, or in any other module/class. Furthermore, while a geometry can be defined for ThermalZones in the semi-direct translation, this was not considered since the ThermalZone corresponds to the whole building volume, and its geometry is already stored in the Building/BuildingPart object.

After ThermalZones, for each surface of the Building/BuildingPart, a ThermalBoundary object



was created as a child of the corresponding ThermalZone object, with one of the following types: outerWall, groundSlab, roof, or sharedWall. The inclination and area of each surface were stored in the corresponding attributes of ThermalBoundary objects, while the orientation and slope of surfaces were added as additional attributes since these were not supported in the semi-direct translation. It is important to note that ThermalOpening objects were not used in this application, since the actual number of openings were not available, except the window ratio of outer walls and roofs. However, this data was added as additional attributes in the ThermalBoundary objects as well. Finally, for each ThermalBoundary object, a Construction object was created to store the U-value of the surface in a straightforward way, while a heat transfer resistance (R-value) attribute, and a g-value attribute for windows were not supported in the semi-direct translation.

Once the building physics properties were stored, information about the building's use was stored in two steps. First, a UsageZone object was created as a child of each ThermalZone. While the intention was to store the number of residential functions in each UsageZone, this was not directly possible since the semi-direct translation did not include an attribute for this data. Therefore, the number of residential functions was added as an out-of-schema attribute. Then, an Occupant object was added for each UsageZone, and the total number of residents in the building was stored with the numberOfOccupants attribute.

The last step of creating a CityJSON + Energy Extension file was to store weather data. It was decided to store only the monthly indoor and outdoor temperature values in the Extension, while the solar irradiation and shading reduction factor values were directly obtained from the database. As a result, for each ThermalZone, two WeatherData objects were created with the type airTemperature. Then, to store the values, RegularTimeSeries objects were created for each WeatherData object, in which the monthly indoor and outdoor temperature values were stored.

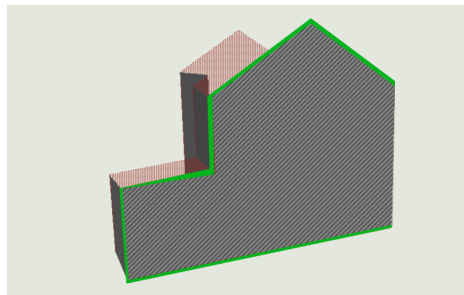
### 5.3 Calculation of space heating demand

After all the required input data was stored in a CityJSON + Energy Extension file either directly with the extension elements or as out-of-schema attributes, the space heating demand was computed based on the calculation method in the Dutch standard *NTA 8800*. However, certain simplifications and assumptions had to be made beforehand to be able to perform the calculation in city-scale, which are summarised as follows:

1. Only residential and mixed-use buildings in the study are considered in the calculation. The reason of this is that it was mostly not possible to acquire all needed input parameters for non-residential buildings, such as internal gains through the people using the building in specified periods.
2. For space heating demand calculation, a heating period of October to March (inclusive) was assumed.
3. A constant monthly value was assumed for indoor air temperature during the heating period.
4. Since the 3D city model in LoD2 does not provide information about the internal structure of buildings, each building was modelled as a single thermal zone. Accordingly, both residential and mixed-use buildings were assumed to have a single thermal zone for the whole volume.
5. Because of the lack of detailed information about the windows in buildings, a window ratio was used to obtain an estimate of the area covered by windows. Therefore, unlike the *NTA 8800* standard where the solar gains are computed for each window of each surface, one hypothetical window was assumed for each surface depending on the window ratio.

## 5 Implementation

6. In the 3D city model, it was detected that some buildings include wall surfaces with very small areas (Figure 5.2). Since assigning a window ratio even for these walls could result in an overestimation in the calculation of solar gains, only the wall surfaces with an area bigger than  $4 \text{ m}^2$  were considered for solar gains through windows, while the smaller ones were still added for solar gains through opaque parts of the building.
7. During the calculation, heat losses through party walls were only considered if the party wall is adjacent to a non-residential building. Otherwise, the heat losses between residential and/or mixed-use buildings were neglected.
8. According to the *NTA 8800* standard, if the number of vertical pipes that are in contact with outside air is not known to calculate the heat loss coefficient through vertical pipes, it can be assumed 1 vertical pipe per residential function. Therefore, during the calculations, it was assumed that the number of residential units in a building equals to the number of vertical pipes.



**Figure 5.2:** A party wall between two buildings (grey with stripe pattern), and the remaining unshared wall surface (green) with a considerably small area.

Once the space heating demand was calculated for the specified type of buildings in the study area, the CityJSON + Energy Extension file was enhanced with the resulting values using the EnergyDemand object. For each ThermalZone (therefore, for each building), an EnergyDemand object was created to store the energy amount needed for space heating in each month of the heating period.

### 5.3.1 Limitations on the retrieval of data from the CityJSON Energy Extension during the space heating demand calculation

During the space heating demand calculation, all needed input data was retrieved from the CityJSON + Energy Extension file that was created in earlier steps. During this process, some limitations and improvement possibilities were identified in the way CityJSON Energy Extension was designed. Firstly, some extension elements were found to include deep hierarchies even after the efforts to avoid them. An example to this is the WeatherData object, which itself points to a RegularTimeSeries object where the actual weather data values are stored. Therefore, retrieving the actual weather data values for a building requires passing numerous links in the CityJSON file, which is against the philosophy of CityJSON that aims to prevent a deep hierarchical structure. As a result, Extension elements with deep hierarchies were added as one of the main considerations to improve the CityJSON Energy Extension. Furthermore, an additional drawback of WeatherData object was identified for the use case. While the semi-direct translation allows to specify the type of weather data (e.g. air temperature, wind speed, solar irradiation), a more detailed specification is not possible, such as the location of weather data (e.g. indoor, outdoor). This resulted in confusion about the data during the calculation, since each ThermalZone included a reference to two WeatherData

## 5 Implementation

objects, one for indoor and one for outdoor temperature, without specifying the location for each object. Therefore, this drawback was considered as well while improving the Extension.

The second possibility for improvement of the CityJSON Energy Extension was detected in the way relations between objects were constructed. While, in the semi-direct translation, relations between all CityObjects were designed with the parent/children mechanism of JSON, this led to extra steps needed to be taken during the calculation. An example to this is the ThermalZone object, which might include a reference to ThermalBoundary and/or UsageZone object in its *children* property. Since these two types of objects were handled differently during the calculation, an additional check was needed to make sure only one type of object was processed at a time.

Yet another possibility to improve the Extension is related to structuring and removing duplicate information that is stored in the CityJSON + Energy Extension file. In the semi-direct translation, each numerical attribute is stored together with its unit of measurement. While it is crucial to include information about the units used in a 3D city model, this may result in duplicate information. For instance, all buildings in the study area include a *volume* attribute to store the value and the unit of measurement. While the volume value may change for each building, the unit of measurement stays the same. Therefore, the storage of the units of measurement would be another way to improve the CityJSON Energy Extension.

### 5.4 Improvements on the semi-direct translation

As a result of the experiments with storage of input data in the CityJSON Energy Extension, as well as with their retrieval for space heating demand calculation, drawbacks of the semi-direct translation were identified to be improved in the final version of the Extension. In the following paragraphs, it is described how these improvements were structured.

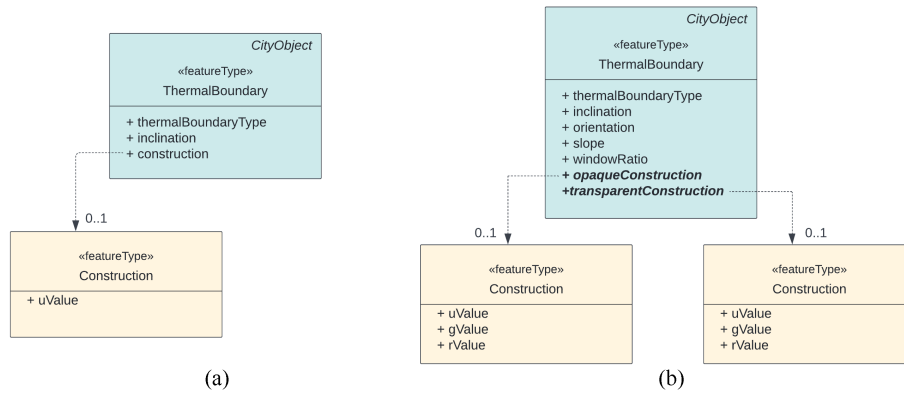
#### 5.4.1 Reconsidering the Energy ADE KIT profile attributes

The first step of improving the CityJSON Energy Extension was to provide full-support for the use case by including missing attributes and relations, as described in [Section 5.2](#). For this, all missing attributes that were added out-of-schema before were now included in the Extension schema, such as building perimeter, orientation and slope of a ThermalBoundary, g- and R-values of Construction objects, and the number of residential functions in UsageZones. Then, all Extension objects were reconsidered to find possibilities to make them more compatible for a steady-state space heating demand calculation. An example to this is the Construction object, which, in the semi-direct translation, had a relation with either a ThermalBoundary or a ThermalOpening object to store detailed building physics parameters of a surface or its openings. However, since ThermalOpening object is not used in this thesis due to the lack of information about individual windows, this relation needed to be updated to still be able to store building physics parameters of openings, independent from the ThermalOpening object. Therefore, the relation between ThermalBoundaries and Construction objects was provided with two additional attributes to ThermalBoundary objects, namely *opaque-Construction* and *transparentConstruction* to store the building physics parameters of the surface itself and its openings with two separate Construction objects ([Figure 5.3](#)).

#### 5.4.2 Associations among objects

Once the CityJSON Energy Extension was made fully compatible for the use case, all the associations among CityObjects and non-CityObjects were analysed, which, in the semi-direct translation,

## 5 Implementation



**Figure 5.3:** The relation between ThermalBoundary and Construction objects (a) with one additional attribute in the semi-direct translation, and (b) two additional attributes in the final version to support transparent parts of the ThermalBoundary.

was provided with parent/children mechanism and additional attributes respectively. For non-CityObjects, it was observed that storing a relation as an attribute instead of a property of the object resulted in an additional level of hierarchy to be passed before reaching the data, since JSON attributes are stored one level below properties. Therefore, it was decided to create properties instead, to store the relations among a CityObject and non-CityObject, or multiple non-CityObjects.

In this step, a limitation of the CityJSON's Extension mechanism was detected. If an existing CityObject has a relation with a non-CityObject, it is not possible to store this as a new property of the existing CityObject, since the Extension mechanism of CityJSON only allows new attributes to existing CityObjects. In the CityJSON Energy Extension, this is problematic for weatherData and energyDemand relations, both of which may be defined for *all* CityObjects. The first possible solution for this was to extend existing CityObjects with new properties by redefining them as new CityObjects in the Extension. However, this would mean redefining all the CityObjects in CityJSON, since the properties of weatherData and energyDemand would have to be added for all CityObjects. To avoid creating an almost duplicate schema, a second solution was formed. Accordingly, only the weatherData and energyDemand relations were kept as attributes for all existing and new CityObjects, while all other relations were defined as properties for Extension objects. An example is given in [Schema 5.8](#), where the new UsageZone CityObject includes weatherData and energyDemand relations with two attributes, while its relation to Occupants object is provided with a property.

After improving the relationships among non-CityObjects, the parent/children relationship was reconsidered for CityObjects to ensure that parent and children properties store only one type of object. Accordingly, the parent/children relationships between Building/BuildingPart, ThermalZone, UsageZone, and Facilities objects were maintained. On the other hand, the relationships between ThermalZone, ThermalBoundary, and ThermalOpening objects were instead formed using additional properties, as suggested in the Energy ADE KIT profile.

### 5.4.3 Removal of deep hierarchies

As discussed in [Section 5.3.1](#), WeatherData object was marked to be improved due to its deep hierarchical structure. Moreover, during the space heating demand calculation, the EnergyDemand object was found to have the same structure as well. Therefore, to simplify this hierarchy, WeatherData and EnergyDemand objects were instead defined as subschemas under the "definitions" keyword of the JSON schema. An example from the definition of WeatherData is given in [Schema 5.9](#). This way, instead of referring to the ID of these objects, the *weatherData* and *energyDemand* attributes of

## 5 Implementation

```
1 "extraCityObjects": {
2   "+UsageZone": {
3     ...,
4     {
5       "properties": {
6         "type": {...},
7         "attributes": {
8           "type": "object",
9           "properties": {
10            "usageZoneType": {"type": "string"},
11            "weatherData": {"type": "string"}, //ID of WeatherData object
12            "energyDemand": {"type": "string"} //ID of EnergyDemand object
13          }
14        },
15        "occupants": {"type": "string"} //ID of Occupant object
16      }
17    }
18 }
```

**Schema 5.8:** UsageZone CityObject with its weatherData and energyDemand relations as attributes, and the occupants relation defined as a property.

CityObjects can directly store the data in these attributes. Furthermore, an additional attribute called weatherDataLocation was added to provide more detail about WeatherData, such as whether it has been obtained from indoor or outdoor calculations.

```
1 "definitions": {
2   "WeatherData": {
3     "type": "object",
4     "properties": {
5       "weatherElement": "...",
6       "weatherDataLocation": "...",
7       "values": {...} //ID of
8         TimeSeries object
9     }
10 }
11 }
```

```
"Build1": {
  "type": "Building",
  "geometry": [...],
  "attributes": {
    "weatherData": [
      {"weatherElement": "airTemperature",
        "weatherDataLocation": "indoor",
        "values": "RegularTimeSeries1"}
    ]
  }
  ...
}
```

**Schema 5.9:** WeatherData, defined as a subschema (left), and its use in a weatherData attribute for a Building object (right).

In addition, to simplify the hierarchy further and to remove duplicate information stored with the CityJSON Energy Extension, it was decided not to store the units of measurement directly within the numerical attribute. To remove units of measurement from each numerical attribute, the data type of these attributes were simply changed to the *number* type of JSON. Moreover, an extra root property called "+unitOfMeasurement" was added in the schema, which enables the user to specify the units of measurement used in a CityJSON + Energy Extension file only once at the root of the document, instead of storing the same information for each attribute. An example of a simple CityJSON + Energy Extension file is shown in [Schema 5.10](#), where the "+unitOfMeasurement" root property is used to store the units used in the file for each numerical attribute.

```

1 {
2   "type": "CityJSON",
3   "version": "1.1",
4   "extensions": {"Energy": {...}},
5   "transform": {...},
6   "CityObjects": {...},
7   "vertices": [...],
8   "+unitOfMeasurement": { //Defined in the schema as an extraRootProperty
9     "energy-volume": "m^3",
10    "energy-floorArea": "m^2",
11    "energy-energyDemand": "kWh",
12    ...
13  }
14 }

```

**Schema 5.10:** The use of the "+unitOfMeasurement" root property in a CityJSON + Energy Extension file.

#### 5.4.4 Naming conventions

In the last step of improving the CityJSON Energy Extension, the Extension schema was checked for naming conventions. First, a prefix was added to all Extension objects to differentiate them from any other CityJSON extensions. Two different versions of this prefix was used in the Extension. The "+energy-" prefix was added to all attributes and properties. However, "+Energy-" prefix was used instead for subschemas and new objects, since, according to the CityJSON specification [Dukai and Ledoux, 2021], all object names must start with an uppercase letter. Second, the use of lower/uppercase letters throughout the schema was checked and corrected. Finally, any unclear names of objects, attributes and properties were altered to make the Extension easier to understand and use. For instance, instead of using the weatherDataType attribute of the WeatherData object, the name of this attribute was changed to weatherElement to indicate a specific element related to the weather, such as air temperature, humidity, and solar irradiation.



## 6 Results and Analysis

In this section the results of the thesis are presented and analysed. First, the effects of the improvements on the semi-direct translation are discussed in [Section 6.1](#). Second, a comparison between the Energy ADE KIT profile and the CityJSON Energy Extension is presented in [Section 6.2](#), in terms of the design of the two data models and their effects on data storage. Then, a comparison on the sizes of the input and output CityJSON files is presented in [Section 6.3](#). Finally, the results of the space heating demand calculation are demonstrated and analysed in [Section 6.4](#).

### 6.1 Semi-direct translation versus the final CityJSON Energy Extension

The comparison between the semi-direct translation and the final CityJSON Energy Extension was done based on the improvements described in [Section 5.4](#). This section focuses first on the effects of the efforts to simplify the hierarchical structure of the Extension, then on the impacts of changing the way relations between objects are stored.

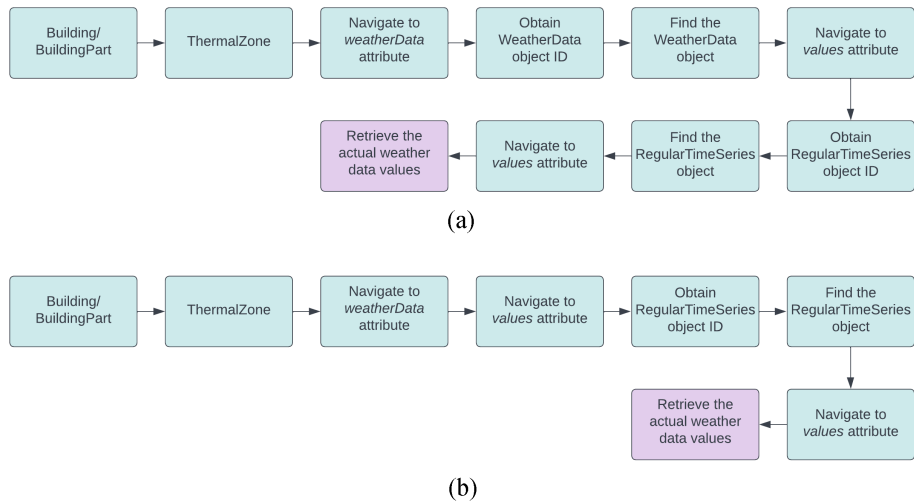
#### 6.1.1 Change in the hierarchical structure

Two main operations were performed to avoid deep hierarchies in the CityJSON Energy Extension. The first operation focused on the WeatherData and EnergyDemand objects, which were found to have a deep hierarchical structure in the semi-direct translation, as described in [Section 5.4](#). In these objects, their hierarchy required multiple lookups in the CityJSON + Energy Extension file to first navigate to the corresponding WeatherData or EnergyDemand object with their IDs, then to the corresponding TimeSeries objects to reach the actual values. To simplify these operations, WeatherData and EnergyDemand objects were instead defined as subschemas. To illustrate this change in hierarchy, [Figure 6.1](#) demonstrates an example from the WeatherData object, where the links to be passed to reach actual weather data values are shown for the semi-direct translation and the final CityJSON Energy Extension. It can be seen from the figure that the number of links decreased by 2 in the final Extension after the improvements. Since the WeatherData object is now stored as a subschema directly within the weatherData attribute, the eliminated operations included the lookup from all the objects in the CityJSON + Energy Extension file to navigate to the corresponding WeatherData object. As a result, the only remaining lookup was for the TimeSeries object to reach the actual weather data values. This way, this improvement resulted in easier and faster access to data stored with the Extension, while making the structure of the Extension simpler at the same time.

The second operation to reduce the hierarchy in the CityJSON Energy Extension, as well as to discard duplicate information, was to remove the unit of measurement properties from the numerical attributes. [Figure 6.2](#) demonstrates how this property was defined in the schema of the semi-direct translation, and the effect of removing it on the access to data after the improvements. It can be seen that the number of links to be passed decreased by 1 in the final CityJSON Energy Extension, since the data type of a numerical attribute was simply changed to the *number* type of JSON. This way, instead of having to navigate to another property, the actual attribute value could be directly obtained. While this may not seem like a significant change compared to the semi-direct translation,



## 6 Results and Analysis



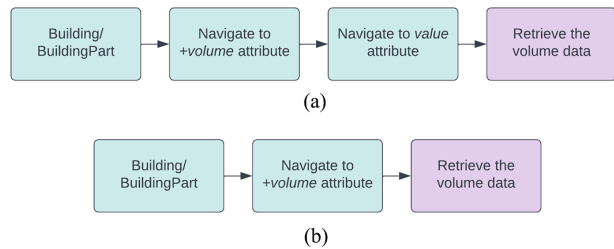
**Figure 6.1:** The links to be passed to reach weather data values stored with (a) the semi-direct translation, and (b) the final CityJSON Energy Extension.

it was highly important for this thesis, since the space heating demand calculation required using many of the numerical attributes in the Extension, such as volume, U-value, and floor area of buildings. Therefore, this improvement simplified the structure of the Extension schema, and resulted in faster access to data during calculations. In addition, the storage of units of measurement in a root property prevented the storage of duplicate information in the Extension, since the unit of each numerical attribute could be specified once in the "+unitOfMeasurement" root property.

```

1  "extraAttributes": {
2    "Building": {
3      "+volume": {
4        "type": "object",
5        "properties": {
6          "value": {"type": "number"},
7          "uom": {"type": "string"}
8        }
9      }
10   }
11  }

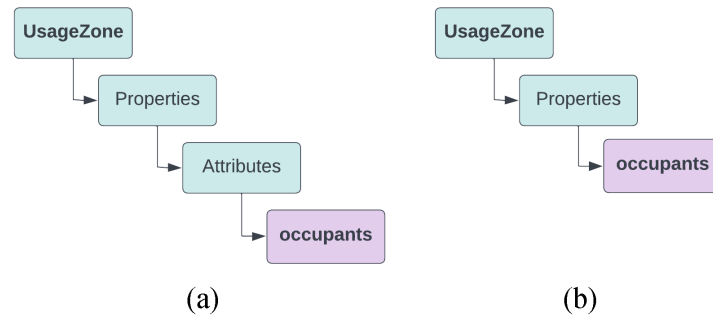
```



**Figure 6.2:** The definition of a *volume* attribute for Building objects with its value and unit of measurement as two separate properties (left), and the links needed to be passed to reach the volume data (a) when unit of measurement is stored, or (b) when only the numerical volume data is stored.

### 6.1.2 Change in the storage of relations

Since the relations between objects were stored differently in the semi-direct translation for CityObjects and non-CityObjects, the improvements also handled these cases separately. First, the associations between non-CityObjects were decided to be formed with additional properties instead of attributes. This operation resulted in a simpler hierarchy when the relation between two objects is queried, since CityJSON properties are stored one level higher than attributes. An example is shown for the relationship between UsageZone and Occupants objects in Figure 6.3, where the Occupants of a UsageZone object is more easily accessible after the improvements.



**Figure 6.3:** The relationship between a UsageZone object and its occupant(s) stored (a) in attributes, and (b) in properties of the object.

After the consideration of non-CityObjects, the relations between CityObjects were also changed for the ThermalZone, ThermalBoundary, and ThermalOpening objects. Accordingly, the associations between these objects were decided to be stored with additional properties as well, instead of the parent/children mechanism. This was due to an additional complexity introduced in the space heating demand calculation, since both ThermalBoundary and UsageZone objects were used, and both were stored as the children of ThermalZone objects in the semi-direct translation. On the other hand, in the final CityJSON Energy Extension, the parent/children relationship was maintained only between ThermalZone and UsageZone objects, while the ThermalBoundary objects of a ThermalZone were stored in the *boundedBy* property. Similarly, the *contains* property of ThermalBoundary object started to be used to store the ID of ThermalOpening objects (Figure 6.4). While this operation did not have a particular impact on the hierarchical structure of the objects, the overall structure of the Extension became more clear and easily understandable, since, with this operation, all parent/children properties started to store references to only one type of object.

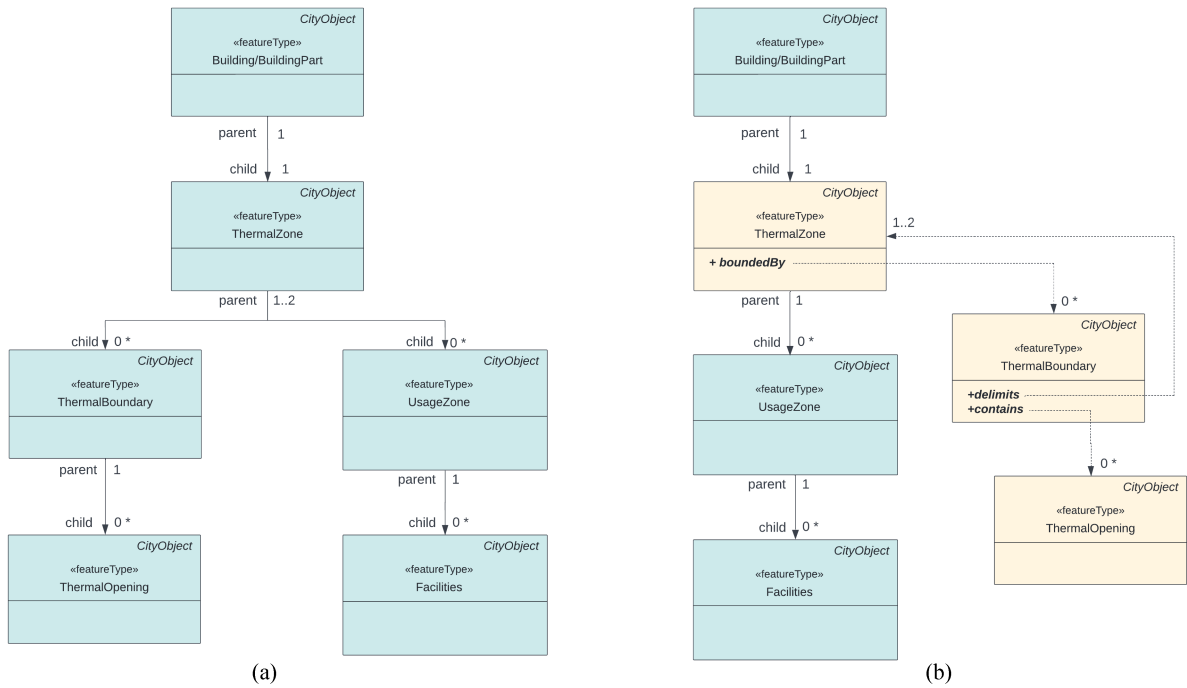
## 6.2 The Energy ADE KIT profile versus the CityJSON Energy Extension

The Energy ADE KIT profile was the starting point for this thesis; however, the final CityJSON Energy Extension is not a direct translation due to the distinct philosophies behind the CityGML and CityJSON data models, and the structural differences between the XML and JSON data formats. Following the design decisions behind CityJSON, one of the main considerations during the design of the Energy Extension was to avoid a deeply hierarchical structure as much as possible, as opposed to the XML-based format of the Energy ADE KIT profile in which the hierarchy is one of the main drawbacks. Therefore, most differences of the CityJSON Energy Extension from the Energy ADE KIT profile reflect this philosophy. In addition, the requirements for the use case, space heating demand calculation, had a significant impact on the design of CityJSON Energy Extension, since the main objective of this thesis was to provide an energy-related extension specifically for this use case.

### 6.2.1 Changes in the used elements

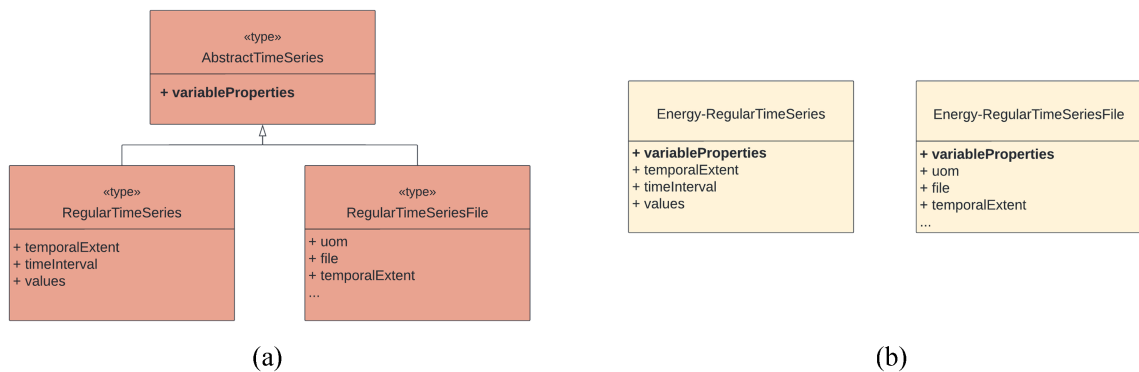
The main differences between the Energy ADE KIT profile and the CityJSON Energy Extension in terms of the used elements, such as classes, attributes, and data types, are presented in Appendix A. Accordingly, the first difference is that the CityJSON Energy Extension does not include abstract classes, such as AbstractThermalZone and AbstractUsageZone, to remove unnecessary hierarchies. In the CityJSON Energy Extension, this did not have a significant impact, since most abstract classes

## 6 Results and Analysis



**Figure 6.4:** Extension CityObjects (a) before and (b) after the parent/children relationships were reconsidered for the ThermalZone, ThermalBoundary and ThermalOpening objects. In this figure, only the corresponding properties are shown, while the other attributes are discarded for visualization purposes.

defined in the Energy ADE KIT profile do not include any attributes to pass on to their subclasses. The only exception to this is the AbstractTimeSeries object, which has a variableProperties attribute to pass on to RegularTimeSeries and RegularTimeSeriesFile objects (Figure 6.5). Only in this case, this attribute was defined for both objects in the Extension schema. However, in overall, it can be concluded that the lack of abstract classes contributed to the simplicity of the Extension schema, which at the same time resulted in less hierarchy.



**Figure 6.5:** TimeSeries objects defined (a) in the Energy ADE KIT profile with inheritance, and (b) in the CityJSON Energy Extension without the abstract class.

The second difference between the Energy ADE KIT profile and the CityJSON Energy Extension concerns the attributes defined in the former. In the CityJSON Energy Extension, certain attribute

names were modified to make them more compatible with the use case, numerical data types were simplified (e.g. no units of measurement), and new attributes were added to comply with the data requirements of the use case. As a result, compared to the Energy ADE KIT profile, the CityJSON Energy Extension consists of simpler and more concise attributes for the use case.

The third difference is related to the extension mechanisms for CityGML and CityJSON. While CityGML ADEs provide a flexible way to define distinct types of objects and data types in a structured way, the extension mechanism of CityJSON is only limited with defining new City Objects, attributes and root properties. One of the main impacts of this limitation on the CityJSON Energy Extension was that the non-CityObjects had to be defined under the `extraCityObjects` property of the extension mechanism, which was originally created to store only CityObject type elements. While this did not cause any validity problems, it can be discussed that it resulted in less clarity in the Extension schema, since the types of stored objects may not always match with the name of the container property in the Extension.

The final difference between the two Extensions is about the decision to store certain data or not. The main idea behind the Energy ADE in general is to be able to store *all* energy-related information in 3D city models so that external data sources are not needed. While this may be beneficial in certain cases, it is highly inefficient when the data to be stored takes up an extensive amount of space in the file, such as weather data. This was proven in the space heating demand calculation, even with a steady-state method, in which solar radiation and shading reduction factor data included different values for each cardinal and ordinal direction, as well as for numerous intervals of inclination and slope values of surfaces. Therefore, these values were not stored with the CityJSON Energy Extension, instead, an external database was used to extract only the needed information, which assured a more manageable CityJSON + Energy Extension file.

### 6.3 Comparison of file size

To analyse the impact of the CityJSON Energy Extension, the size of input, intermediary, and output files was compared, as demonstrated in Table 6.1. The first input file was the 3D city model in CityGML format, which included 3318 Building/BuildingPart objects with their geometries and attributes, including Generic Attributes to add extra energy-related information, which had a file size of 165 MB. When this file was converted to CityJSON, the file size dropped significantly, as expected, since CityJSON files have a much higher compression factor than XML-based CityGML files [Ledoux et al., 2019].

After all needed input data was stored in a CityJSON + Energy Extension file, the number of stored objects increased dramatically, from 3318 in the input CityJSON file to 106848 in the resulting CityJSON + Energy Extension file. As a result, the file size also increased by 23.1 MB in the resulting file. Then, after the calculations, 1884 more objects were added to the CityJSON + Energy Extension file, to create an EnergyDemand object for each Building/BuildingPart in the calculation, and to store the final space heating demand values. After this operation, the final output CityJSON file had a size of 65.8 MB. It can be discussed that the overall increase of 25.2 MB in file size is not significant, considering that 105414 more objects were added to the input file.

It is important to note that a comparison between the Energy ADE KIT profile and the CityJSON Energy Extension could not be done, since the input 3D city model in CityGML format only included Generic Attributes to add additional energy-related data. However, this information still provided valuable insight when compared with the final output CityJSON file. It can be seen that the input CityGML file with 3318 Building/BuildingPart objects and only a small subset of the needed input data takes up 165 MB of space, while the output CityJSON + Energy Extension file with 108732 objects to store all needed input data takes up only 65.8 MB of space. This result once again proves

	3DCM in CityGML	3DCM in CityJSON	CityJSON + Energy Extension (only input data)	CityJSON + Energy Extension (output file)
# of objects	3318	3318	106848	108732
File size	165 MB	40.6 MB	63.7 MB	65.8 MB

**Table 6.1:** File size and the number of objects stored in each input/output file used in the thesis.

the efficiency of CityJSON compared to the XML-based CityGML, even when the core data model of CityJSON is extended with additional objects and types.

## 6.4 Results of the space heating demand calculation

Out of the 3318 buildings in the study area, space heating demand calculation was performed for 1884 buildings while the remaining 1434 buildings were omitted in the calculations, because they either had non-residential functions or lacked required input data. [Figure 6.6a](#) demonstrates the distribution of the considered and omitted buildings in the center of Rijssen, and [Figure 6.6b](#) shows the same distribution in a residential area of the city.

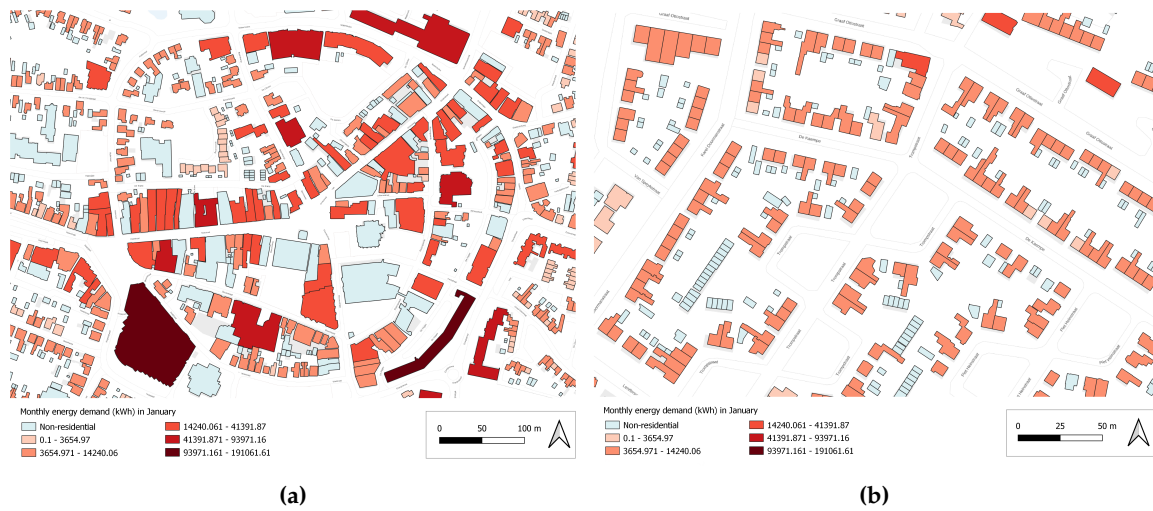
It can be seen that the omitted buildings in the two areas mostly differ in terms of size, as the center generally consists of larger structures compared to a purely residential area. In addition, when the functions of the omitted buildings are analysed in the 3D city model, it is seen that the excluded buildings in the center are mainly used for commercial purposes, such as shops or offices, while the ones in the residential area are mostly used as garages or storage rooms. As a result, space heating demand was calculated only for residential and mixed-use buildings in the area, as long as they included all required input data for the calculation.



**Figure 6.6:** Considered and omitted buildings during the space heating demand calculation (a) in the center and (b) in a residential area in Rijssen.

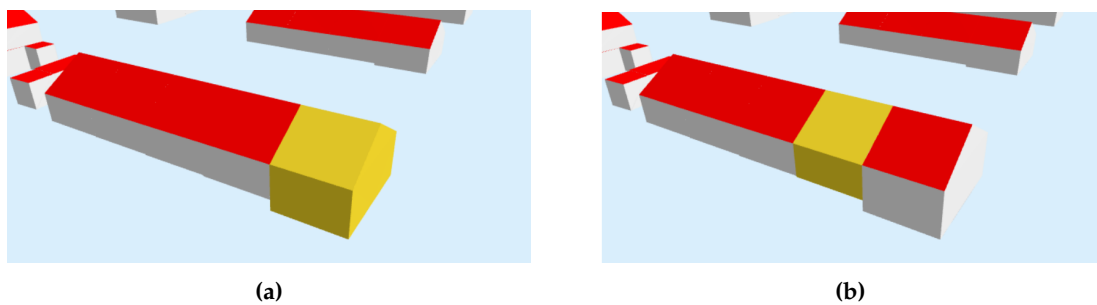
The resulting space heating demand values are displayed for the month January in [Figure 6.7](#) for the center of Rijssen and the same residential area of the city. While the space heating demand

calculation includes numerous non-geometrical factors, such as building usage and occupant behaviour, it can be directly seen from the figures that the building size is a significant factor in the determination of energy demand. While large buildings in the center mostly have a higher energy demand, the buildings in the purely residential area have lower and similar energy demand values, considering that their size and geometry are comparable, which, in the end, have an impact on the number of occupants and overall usage as well. Furthermore, it can be discussed that the adjacency between residential and non-residential buildings may have contributed to higher energy demand values, since the energy loss through the shared thermal boundaries are mainly considered in the center, compared to the residential area, where an energy loss between two residential buildings are neglected.



**Figure 6.7:** Energy demand (kWh) in the month January (a) in the center and (b) in a residential area in Rijssen.

While a visual inspection of the overall distribution of energy demand values provides valuable insight, it is crucial to analyse the impact of specific characteristics of buildings on the resulting energy demand. The first of these analyses focuses on the position of buildings, where [Figure 6.8](#) demonstrates a corner and a middle building with type Terrace House, and [Table 6.2](#) shows their characteristics as well as their energy demand in the month January. It can be seen that the main difference between the buildings is their position, while their other characteristics, such as year of construction, building type, building class, usable area and volume, are either the same or similar. Under these conditions, the corner building has a higher energy demand, since more of its surfaces are exposed to outside air for heat losses, compared to the middle building.



**Figure 6.8:** Examples of corner (a) and middle (b) buildings with the type Terrace House.

## 6 Results and Analysis

A similar analysis was conducted to see the impact of the building position for single-family houses. The energy demand of a stand-alone building was compared with that of a corner building (Table 6.2), and a higher energy demand was observed for the stand-alone building. Similar to the previous case, this impact is related to the overall surface area that is exposed to outside air for heat losses. While all wall surfaces are considered during the calculation of heat losses through transmission for a stand-alone building, at least one wall surface of a corner building is neglected, since it is shared with another residential building.

Building position	Year	Building type	Building class	Usable area	Building volume	Energy demand - January (kWh)
Corner	1974	TH	Residential	94	377.67	3945.98
Middle	1974	TH	Residential	92	308.52	2615.43
Stand-alone	1985	SFH	Residential	183	703.13	4373.22
Corner	1981	SFH	Residential	127	495.885	2895.05

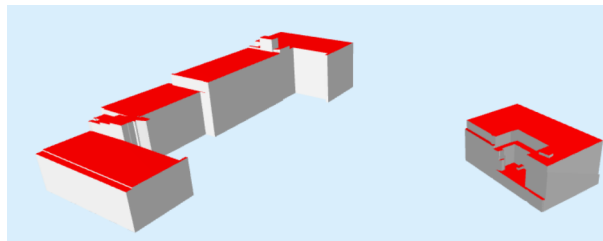
**Table 6.2:** Comparison of energy demand in January for corner, stand-alone and middle buildings. TH = Terrace House, SFH = Single Family House.

Yet another analysis was done to compare the effect of the year of construction on the resulting energy demand values. Table 6.3 shows the characteristics of two buildings with construction years of 1933 and 2004. It can be seen that while all other characteristics are same or very similar, the former building has a much higher energy demand compared to the latter building with a more recent year of construction. This difference in energy demand is expected, since older buildings usually contain more inefficient materials, which is reflected in the used building physics parameters (from TABULA) during the calculations as well.

Year	Building type	Building class	Usable area	Building volume	Building position	Energy demand - January (kWh)
1933	SFH	Residential	200	753.98	Stand-alone	9067.22
2004	SFH	Residential	174	755.29	Stand-alone	3483.54

**Table 6.3:** Comparison of energy demand in January for two buildings with distinct construction periods. SFH = Single Family House.

The final analysis on the building characteristics considered the impact of building volume on the resulting energy demand values. It can be seen from Figure 6.9 and Table 6.4 that a building with a higher volume requires much more energy for space heating compared to that with a smaller volume. Since a higher volume implicitly suggests a higher usable area in general, it is understandable that the building size has a significant impact on the resulting energy demand values.



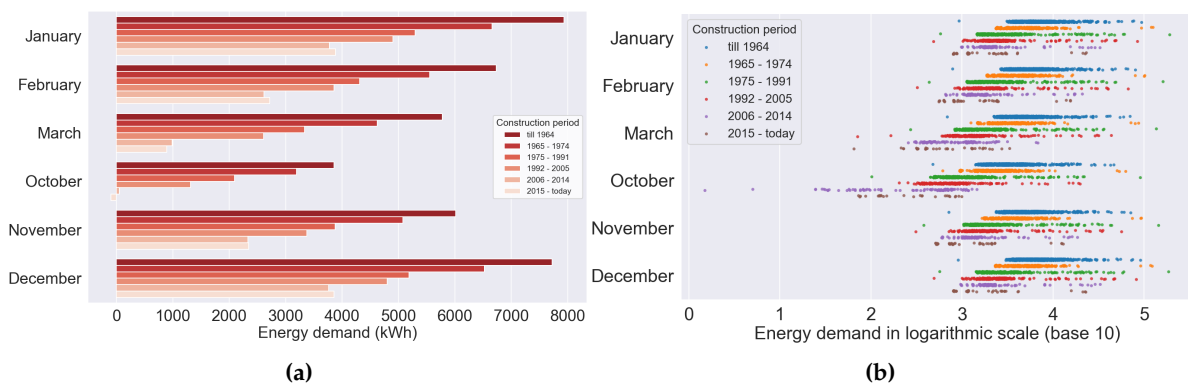
**Figure 6.9:** Two buildings with differing sizes from the study area.

Year	Building type	Building class	Usable area	Building volume	Building position	Energy demand - January (kWh)
1975	MFH	Mixed-use	3147	14962.58	Stand-alone	59094.54
1980	MFH	Mixed-use	2225	7556.95	Stand-alone	26861.04

**Table 6.4:** Comparison of energy demand in January for two buildings with distinct volumes. MFH = Multi Family House.

After the analysis of specific building characteristics on energy demand, a general analysis is provided on the following paragraphs considering the same building characteristics and the overall dataset. For this purpose, the monthly average energy demand values for the heating period of October to March are demonstrated in Figure 6.10a, which are grouped by construction periods that were determined in the TABULA building physics library. It can be seen from the graph that the energy demand values follow a logical order, where the demand is higher in colder months such as January and December, compared to the warmer periods of the year. In addition, it is seen that the average energy demand values are higher for older construction periods compared to newer ones. As discussed before, an important reason of this is the difference in used construction elements, which affect the used building physics parameters such as higher U-values.

In addition to an analysis on the relationship between construction periods and energy demand, it can be observed from Figure 6.10a that the average energy demand values are considerably high, even almost reaching 8000 kWh per month for the oldest construction period. In addition, a negative average of energy demand values is observed in October, which is against the logic of the space heating demand calculation based on an energy balance method. To further investigate these values, Figure 6.10b demonstrates the distribution of monthly energy demand values in logarithmic scale for each construction period. It is possible to detect from the graph that while a majority of the values are accumulated for each period, there are many outliers with much higher and lower values, affecting the mean of the distribution. Especially for the oldest period (till 1964), which has the highest average value for each month as shown in Figure 6.10a, almost all the outliers are positioned on the higher side of the graph, which explains the considerably high average monthly energy demand values. On the other hand, an analysis on the obtained negative energy demand values and possible solutions are included later in this section.



**Figure 6.10:** Average monthly energy demand (kWh) (a) and the distribution of values on logarithmic scale (b) depending on the construction period.

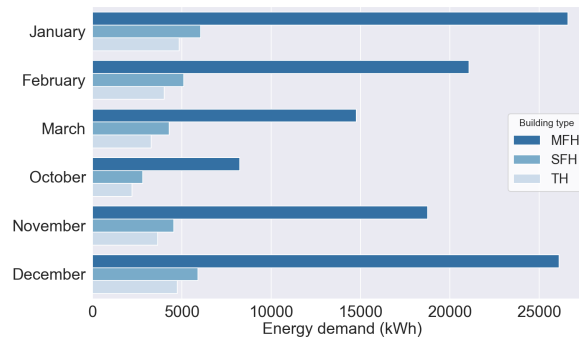
After the analysis on the monthly average energy demand depending on the construction periods of buildings, a similar analysis was performed to investigate the relationship between the monthly average energy demand and the building typology. While the initial TABULA building



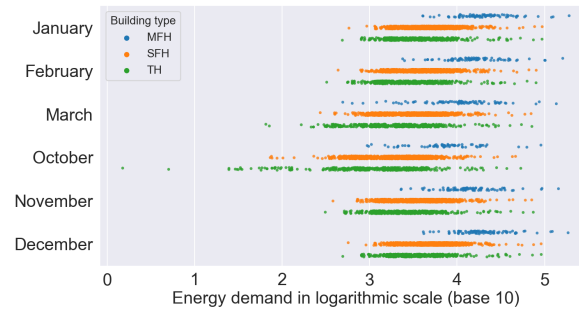
## 6 Results and Analysis

classification identified four types of houses for the Netherlands, namely single family house, multi family house, terrace house, and apartment blocks, only the first three categories were identified in the study area. Accordingly, out of the 1884 buildings considered in the calculations, 84 of them were labelled as multi family houses, 1295 of them as single family houses, and the remaining 505 as terrace houses.

It can be seen from [Figure 6.11a](#) that the monthly average energy demand is highest for the multi family house category, which is significantly high compared to the other categories. Single family house and terrace house categories, on the other hand, have similar values, with monthly averages of around 5000 kWh or lower. Similarly, [Figure 6.11b](#) demonstrates the distribution of monthly values for each building type in logarithmic scale. It can be seen that the values for multi family houses are dispersed more unevenly in the scale, compared to single family and terrace houses, which both have denser aggregations of values. However, for all three categories, it is still possible to detect outlier values on both the higher and lower parts of the scale, affecting the monthly averages. In addition, [Table 6.5](#) shows the average building volumes for each of the three building categories in the study area. It can be seen that the average volume is considerably high for multi family houses compared to the single family and terrace houses, which have similar average volumes. This comparison on building volume justifies the calculated energy demand values as well, as shown in [Figure 6.11a](#), since the multi family houses with largest volumes have the highest monthly average energy demand values.



(a)



(b)

**Figure 6.11:** Average monthly energy demand (kWh) (a), and the distribution of values on logarithmic scale (b), depending on the building type. SFH = Single Family House, MFH = Multi-Family House, TH = Terrace House

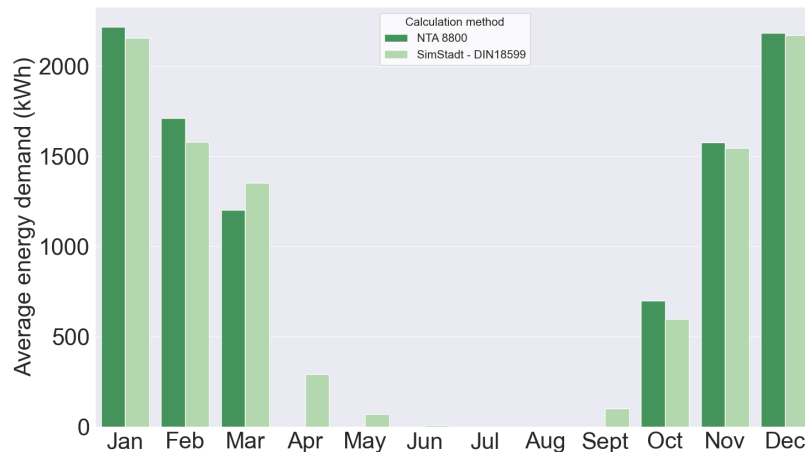
Building type	Average building volume ( $m^3$ )
Multi Family House	7694.98
Single Family House	592.35
Terrace House	537.21

**Table 6.5:** Average volume of the three building typologies in the study area.

#### 6.4.1 Comparison of results with energy simulation tools

After an analysis on the obtained space heating demand values based on different factors such as construction period and building typology, these values were compared with the energy demand values computed using the SimStadt software for the same study area. Table 6.6 presents the monthly space heating demand values calculated with the NTA 8800 standard as part of this thesis and the SimStadt software for a 10-building subset from the study area.

It can be noticed that the resulting values are quite similar for the two calculation methods in each month. However, in this thesis, a heating period of October - March was considered, while SimStadt considers the whole year for the space heating demand calculation. As a result, SimStadt shows energy demand values for the period of April - September as well, while this period was omitted in the calculations in this thesis. In addition to this analysis, Figure 6.12 demonstrates the difference between the monthly average energy demand values for the same 10-building subset for the two calculation methods. It can be more clearly seen from this figure that the difference between the values are highly small, where the NTA 8800 values are generally on the higher side.



**Figure 6.12:** The differences between the average energy demand values (kWh) calculated with NTA 8800 and Simstadt for a 10-building subset.

While the comparison between NTA 8800 and SimStadt results help to justify the space heating demand values calculated in this thesis, it is important to consider certain differences between the two calculation methods. Firstly, the NTA 8800 standard is used specifically in the Netherlands, while the SimStadt software uses the German standard DIN V 18599 for this calculation. Even though both methods are based on a steady state energy balance method, the additional data coming from SimStadt's pre-built libraries (building physics, occupancy, etc.) are based on German regulations as well. Therefore, this comparison should not be seen as a way of validating the NTA 8800 results, since the used data and the calculation methods might slightly change.

ID	Calculation method	Monthly space heating demand (kWh)											
		Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1	NTA 8800	2591.59	2034.86	1375.14	0.00	0.00	0.00	0.00	0.00	0.00	757.42	1819.62	2556.45
	SimStadt	2770	2034	1717	339	68	7	0	0	114	774	1975	2769
2	NTA 8800	1942.68	1418.77	848.89	0.00	0.00	0.00	0.00	0.00	0.00	348.17	1289.71	1917.29
	SimStadt	2309	1590	1282	190	42	4	0	0	57	473	1624	2388
3	NTA 8800	2753.34	2145.44	1520.96	0.00	0.00	0.00	0.00	0.00	0.00	917.06	1984.85	2725.71
	SimStadt	2628	1947	1694	402	112	15	0	1	154	784	1896	2636
4	NTA 8800	1892.72	1452.05	1059.21	0.00	0.00	0.00	0.00	0.00	0.00	614.47	1345.14	1856.11
	SimStadt	1736	1284	1111	242	56	6	0	0	76	479	1242	1742
5	NTA 8800	1797.70	1380.37	997.54	0.00	0.00	0.00	0.00	0.00	0.00	575.44	1275.39	1763.91
	SimStadt	1670	1237	1078	239	57	6	0	0	77	468	1196	1675
6	NTA 8800	2724.22	2151.14	1605.09	0.00	0.00	0.00	0.00	0.00	0.00	1033.08	1999.47	2686.14
	SimStadt	2406	1782	1535	344	84	10	0	1	120	695	1731	2414
7	NTA 8800	2039.21	1583.15	1153.75	0.00	0.00	0.00	0.00	0.00	0.00	688.79	1463.18	2006.01
	SimStadt	1895	1397	1204	267	66	8	0	0	91	533	1361	1907
8	NTA 8800	2025.75	1558.57	1121.38	0.00	0.00	0.00	0.00	0.00	0.00	663.15	1447.32	1993.33
	SimStadt	1870	1379	1191	260	62	7	0	0	87	525	1341	1880
9	NTA 8800	2063.41	1594.70	1151.01	0.00	0.00	0.00	0.00	0.00	0.00	687.70	1478.79	2031.04
	SimStadt	1938	1431	1236	277	68	8	0	0	94	549	1392	1948
10	NTA 8800	2330.16	1798.68	1190.06	0.00	0.00	0.00	0.00	0.00	0.00	707.08	1656.75	2306.40
	SimStadt	2322	1713	1457	340	87	13	0	1	123	679	1679	2345

**Table 6.6:** Comparison of monthly space heating demand values for a 10-building subset calculated with NTA 8800 and SimStadt. The real (pand) IDs of buildings are not included in the figure to anonymise the data.

## 6 Results and Analysis

It is important to note that the negative energy demand values obtained with NTA 8800, as shown in Figure 6.10a, were checked in the SimStadt values; however, it was found that SimStadt results did not include any negative values, as expected. Therefore, a more detailed analysis was made to solve the problem of negative values. It was found that 32 buildings out of 1884 had negative energy demand values, 10 of which are shown in Table 6.7. To investigate the negative values, the four main parameters of the calculation, namely transmission and ventilation losses, and internal and solar gains, were analysed separately. It was found that 1401 buildings in the study area out of 1884 had negative solar gains values, which contributed to the negative overall energy demand values for the 32 buildings. Therefore, all calculation steps were checked more in detail, with a focus on the solar gains calculations.

As the first step of this check, the formulas in the calculation method, all input data and the unit of measurement of each parameter were controlled. It was found that the window ratio data coming from the TABULA building physics library included very high values for certain building categories, such as a window ratio of 0.51 for multi-family houses and of 0.39 for terrace houses built after the year 2000. Moreover, when the 32 buildings with negative energy demand values were analysed, it was discovered that all buildings were built after the year 2000 and the buildings were either multi-family or terrace house types.

ID	Year	Class	Type	Monthly space heating demand (kWh)						
				Jan	Feb	Mar	Apr - Sep	Oct	Nov	Dec
11	2006	Mix.	MFH	4926.104	3164.618	493.2403	0	-480.732	2951.648	4907.25
12	2008	Mix.	MFH	9307.493	6201.38	1204.335	0	-1136.18	5344.322	9260.464
13	2011	Res.	TH	1756.013	1158.104	386.3949	0	-29.8774	1066.519	1750.983
14	2010	Res.	TH	1786.082	1181.133	257.1867	0	-98.3848	1081.698	1797.701
15	2009	Res.	TH	1161.864	742.2068	300.3615	0	-18.9162	694.6194	1144.829
16	2010	Mix.	MFH	26330.35	17157.38	3156.998	0	-4463.61	14623.96	26365.8
17	2012	Res.	MFH	7946.015	4610.537	-851.415	0	-2420.32	4212.729	8107.341
18	2012	Mix.	MFH	28914.34	17449.27	1529.095	0	-5327.15	16200.62	28993.83
19	2014	Mix.	MFH	19637.98	10367.67	-5014.41	0	-11479.8	8306.489	19928.46
20	2013	Mix.	MFH	32795.87	20979.23	2316.002	0	-5191.17	18565.13	32985.15

**Table 6.7:** Buildings with negative energy demand values. Year = Year of construction, TH = Terrace house, MFH = Multi family house, Res. = Residential, Mix. = Mixed-use.

The second step was to perform the calculation for a single residential building manually, to ensure that all formulas were implemented correctly in the Python script created for this thesis. However, the results of the manual calculation matched exactly with the output obtained from the automatic calculations in Python. Then, the impact of each parameter in the calculation was examined. It was found that the negative values in solar gains calculations were caused by the solar gains through opaque elements of the building. In this calculation (Equation 3.14), the extra heat flow due to the radiation to the sky ( $Q_{sky,wi,k;mi}$ ) is subtracted to consider this heat loss in the solar gains calculation. It was found that this parameter becomes too large for opaque parts of the building, resulting in a negative value in the overall solar gains. However, even though the cause of the problem was identified, a solution could not be formulated since no errors or wrong implementations were detected in the whole solar gains calculations.

While the negative energy demand and solar gains values could not be fixed in the given time frame of the thesis, the detected potential problem areas in the input data and the calculation method were modified to check their impact on the resulting values. First, an additional filter was implemented for the window ratio data, which would only consider the 70% of the given window ratio if the year of construction equals to or higher than 2000 for multi-family and terrace houses. This

## 6 Results and Analysis

way, the large window ratio values coming from the TABULA building physics library could be decreased. Second, since no error could be found in the "extra heat flow due to the radiation to the sky" parameter, it was completely removed from the calculation. As a result of the aforementioned tests, all negative energy demand and solar gains values were eliminated. An example is given in Table 6.8, where the building 13, which originally had negative energy demand values (shown in Table 6.7), no longer contains negative results. However, it is important to note that while these tests helped to eliminate negative values, more research and tests are needed to grasp the overall impact of the changes on the whole dataset.

ID	Year	Class	Type	Monthly space heating demand (kWh)						
				Jan	Feb	Mar	Apr - Sep	Oct	Nov	Dec
13	2011	Res.	TH	1770.98	1282.00	682.92	0	205.22	1131.74	1742.47

**Table 6.8:** Change in the energy demand values for one building after adjustments in the calculation method. Year = Year of construction, Res. = Residential, Mix. = Mixed-use.



# 7 Conclusions and Future Work

This chapter concludes the thesis by first providing an overview of the research, as well as the answers to the determined research questions (Section 7.1). Then, the main contributions of the thesis and its limitations are discussed in Section 7.2, and recommendations and new ideas for future work are presented in Section 7.3. Finally, a self-reflection is included in Section 7.4, which focuses on the reproducibility of the thesis, and provides a personal evaluation.

## 7.1 Research overview

The aim of this thesis was to develop and test an Energy Extension for CityJSON that supports the calculation of space heating demand of buildings. As the first step, a semi-direct translation was created from the Energy ADE KIT profile to explore the possibilities for a direct mapping as much as possible. The validation of this Extension was done in two steps: through the official validator of CityJSON and the use case. While the former was used to check validity against the JSON syntax and the official CityJSON schemas, the latter considered the storage of required input data, as well as the output values, with the Extension. For this, the space heating demand calculation was performed for the city of Rijssen within the Rijssen-Holten municipality of the Netherlands, with a steady-state energy balance method according to the Dutch norm NTA8800. Then, the validation results were utilised as guidelines to improve the CityJSON Energy Extension to make it more compatible with the philosophy of CityJSON while fully supporting the use case. The schema of the CityJSON Energy Extension as well as the Python script to calculate space heating demand are available at: <https://github.com/ozgetufan/cjenergy>

In this thesis, the main research question was determined as: *How can a CityJSON Energy Extension be used to support the calculation of space heating demand of buildings?* To answer this question, four sub-questions were defined, which will be answered in the following paragraphs.

- **How can different types of objects, other than CityObjects, be defined in a CityJSON Extension?**

The current extension mechanism of CityJSON only supports the addition of new CityObjects, attributes and root properties with the three corresponding members of the extension, namely *extraCityObjects*, *extraAttributes*, and *extraRootProperties*. On the other hand, it is currently not possible to define non-CityObjects separately in a CityJSON Extension. Therefore, in this thesis, non-CityObjects were defined under the existing *extraCityObjects* member, even though the defined features do not inherit the attributes and relations that CityObjects possess. While this solution did not result in invalidity problems, it can be discussed that it had a negative impact on the simplicity and understandability of the Extension, since the *extraCityObjects* member, which was designed to store only CityObject type objects, was used to store different types of features as well. Therefore, it can be concluded that the current extension mechanism of CityJSON limits the possibilities of defining distinct types of objects, and a workaround is needed to be able to include these objects in a CityJSON Extension.

- **How should the CityJSON Energy Extension differ from the Energy ADE?**

The literature study showed that the main difference between CityGML ADEs and CityJSON Extensions is that the former are generally built with deep hierarchical structures as a result of the XML data format. On the other hand, CityJSON Extensions follow the main philosophy of CityJSON, which is to avoid deep hierarchies to ensure efficiency. As a result, this was determined as the main consideration when creating the CityJSON Energy Extension as well. While the first step of the methodology focused on a semi-direct translation from the Energy ADE KIT profile, objects with deep hierarchical structures were detected to be improved in the final version of the CityJSON Energy Extension.

In addition to the structural contrasts between the CityJSON Energy Extension and the Energy ADE, the requirements of the use case, space heating demand calculation, resulted in further differences between the two extensions. It was observed that the Energy ADE KIT profile was missing certain attributes, such as the orientation and window ratio of surfaces and the perimeter of buildings, which were required for the use case. These attributes were included in the final version of the CityJSON Energy Extension to be able to fully support the use case. Therefore, it can be concluded that the main purpose of differentiating CityJSON Energy Extension from the Energy ADE KIT profile was to comply with the design decisions behind CityJSON while providing support for the use case.

- **How can space heating demand calculation be used during the design phase to test and improve the CityJSON Energy Extension?**

Space heating demand was chosen as the use case of this thesis, since it requires various types of data to be stored in the CityJSON Energy Extension. Space heating demand was first used to validate the Extension by checking whether it fully supports the use case or not. For this, the required input data was examined to detect any missing objects or attributes in the Extension. If certain data could not be stored, these were marked to be included in the final version of the Extension.

The second use of the space heating demand calculation was to improve the CityJSON Energy Extension by making it simpler and easier to use with less hierarchies. To achieve this, various tests were performed with the input data during the space heating demand calculation to measure the efficiency of storing data with existing Extension objects and attributes. The efficiency was measured according to the hierarchical structure of the object and the ease of retrieving data from the CityJSON + Energy Extension file. The objects with deep hierarchies were detected to be simplified to follow the philosophy of CityJSON.

- **To what extent is it possible to map CityGML ADEs to CityJSON Extensions? Should the CityJSON schema be extended to make this process more straightforward?**

The literature study, as well as the thesis itself, showed that the possibilities for a direct mapping from CityGML ADEs to CityJSON Extensions depend on the type of objects. Since the extension mechanism of CityJSON only considers *extraCityObjects*, *extraAttributes*, and *extra-RootProperties*, the CityGML ADEs including only these features can be easily mapped to a CityJSON Extension. However, if the CityGML ADE involves non-CityObjects, as is the case in the Energy ADE, these objects cannot be mapped to a CityJSON Extension in a straightforward manner.

While the existing mechanism can still be utilised to define non-CityObjects, this leads to less clarity in the overall structure of a CityJSON Extension because of the lack of a separate property to define non-CityObjects. It can be concluded that the existing extension mechanism of CityJSON restrains the development of a wide variety of CityJSON Extensions, since it limits



the type of features that can be added to three. Therefore, the extension mechanism of CityJSON should be extended to allow for the definition of non-CityObjects in a systematic way, which would enable a more straightforward mapping from CityGML ADEs, while contributing to the usability of CityJSON Extensions.

## 7.2 Discussion

### 7.2.1 Contributions

As a result of this thesis, a CityJSON Energy Extension was created to support the calculation of space heating demand of buildings. It can be discussed that the design and testing of such an Extension contributes to the overall development of CityJSON and related future work, as described below:

- **Preference over CityGML:** Prior to this thesis, an energy-related Extension was not available for CityJSON. Considering the current significance of Urban Energy Modelling and energy simulations, it can be said that the lack of such an Extension highly restricted the overall use of CityJSON compared to CityGML. With the development of the CityJSON Energy Extension, energy-related data of buildings can easily be stored in CityJSON files, making it possible to use CityJSON for energy simulations, and especially for energy demand calculations. This can be expected to increase the usability of CityJSON for a wider variety of applications, considering the advantages of the CityJSON Energy Extension over the CityGML Energy ADE, such as the file size and a less hierarchical structure.
- **Development of new CityJSON extensions:** The CityJSON Energy Extension can be said to be the most complex CityJSON extension developed so far, with distinct types of objects, data types and relations. The availability of such a complex extension may be beneficial for the development of new CityJSON extensions, where the design decisions behind the Energy Extension can be used as guidelines to create complex elements.
- **Development/improvement of CityJSON tools:** During the development of the CityJSON Energy Extension, many tools to manipulate and visualise CityJSON files were used, such as *cjval*<sup>1</sup>, *citygml-tools*<sup>2</sup>, and *ninja*<sup>3</sup>. The thesis helped to identify certain disadvantages and deficiencies of these tools, which were communicated to the developers for the further improvement of the tools. In addition, the thesis may contribute to the development of new CityJSON tools in the future, focusing on the visualisation and manipulation of extension objects and their relations to the core data model.
- **Further development of the Energy ADE:** In the thesis, an analysis of the Energy ADE KIT profile was provided during the semi-direct translation. This analysis showed the shortcomings of the data model, which can be used in the future for the further development of the KIT profile, as well as the full Energy ADE.

### 7.2.2 Limitations

While the thesis provides valuable contributions for the development and use of CityJSON, the used approach still has certain limitations, which can be described as follows:

---

<sup>1</sup><https://github.com/cityjson/cjval>

<sup>2</sup><https://github.com/citygml4j/citygml-tools>

<sup>3</sup><https://ninja.cityjson.org/>

- **Flexibility of the Extension schema:** Following the philosophy of CityJSON, the Energy Extension schema was developed in a way that ensures high flexibility, where the user is free to define new attributes or properties for the Extension objects, even though they are not officially defined in the schema. While this may contribute to the usability of the Energy Extension when the already defined objects and attributes are not sufficient for a specific use case, it may make CityJSON + Energy Extension files more susceptible to errors. Since the validator of CityJSON will not raise an error for objects or attributes/properties that are defined outside the schema, any misspelled object, attribute or property names will be overlooked. On the other hand, it can be discussed that this, at the same time, is a limitation of the validator, which can be improved to raise a warning when an object or attribute/property that is not defined in the schema is detected.
- **Definition of new objects in the Extension schema:** The limited extension mechanism of CityJSON resulted in an unclear schema for the Energy Extension, where the *extraCityObjects* member held definitions for both CityObjects and non-CityObjects. While this did not create invalidity problems, it may be confusing for the user to discover that non-CityObjects were not defined separately. However, this limitation is caused purely by the core data model of CityJSON, which currently does not provide an additional mechanism to define different types of objects in extensions. Since all possible options were investigated in this thesis to define non-CityObjects with the existing extension mechanism, it was found that this limitation can be avoided only if the extension mechanism of CityJSON is improved to allow for the definition of non-CityObjects in a systematic way.
- **Input data for the space heating demand calculation:** In this thesis, the main data source for the space heating demand calculation was the 3D city model of Rijssen-Holten, where each building had energy-related Generic Attributes, such as area, orientation, and aspect of surfaces. In addition, the 3D city model was used to obtain more of the required parameters, such as the perimeter of buildings and slope of surfaces. However, the fact that the building geometries were only available in LoD0 and LoD2 made it impossible to obtain certain input data from the 3D city model. For instance, since LoD2 geometries do not include openings on surfaces, detailed information about windows, such as their area, could not be used in the calculations. Furthermore, since LoD2 geometries do not provide information about the internal structure of buildings, simplifications had to be made to define the thermal zones within buildings, which was especially problematic for mixed-use buildings. Therefore, it can be concluded that a 3D city model with a higher LoD would improve the results of the space heating demand calculation.
- **Results of the space heating demand calculation:** The space heating demand calculation implemented in this thesis resulted in negative values for 32 buildings, as well as negative results in solar gains during the calculations, which is neither expected, nor possible in a steady-state energy balance method. While the potential reasons of this error were explained in [Section 6.4](#) and tests were performed to find a solution, the problem of negative values could not be fixed in the given time frame of the thesis.  
Another limitation of the space heating demand calculation is that there was no ground truth data to validate the resulting values. The results were only compared with the values obtained from the SimStadt software for the same study area. While this comparison provided insight on the general accuracy of the results, a real validation could not be performed since manually calculated energy demand data was not available for the area.

## 7.3 Future work

During the implementation of the thesis, new ideas and recommendations for future work arose, which could contribute to the further development and usability of the CityJSON Energy Extension. These potential research areas are described in the following paragraphs.

### 7.3.1 Further development of the Energy Extension

The developed CityJSON Energy Extension focuses on supporting the space heating demand calculation of buildings as the use case. To further develop the Extension and widen the range of fields and applications that it can be used in, the Extension could be tested with different use cases, such as solar potential analysis and future refurbishment scenarios. A variety of use cases could unveil the shortcomings of the Energy Extension in terms of supported classes and attributes, which could be the starting point for further development. In addition, since the use case was computed with a simplified (steady-state) method in this thesis, the Energy Extension was developed by considering the Energy ADE KIT profile instead of the full Energy ADE. As a result of the tests with distinct use cases, an assessment between the two versions of the Energy ADE could be made, and the missing objects and attributes from the full Energy ADE could be incorporated in a new version of the CityJSON Energy Extension to provide better support for additional use cases. Finally, it is important to keep in mind that both CityJSON and the Energy ADE data models are frequently updated and new versions are released with the developments in the field of Geomatics. Therefore, it is crucial to follow these developments, and update and improve the CityJSON Energy Extension accordingly.

### 7.3.2 Additional methods for validation and comparison

As discussed in [Section 3.2](#), the CityJSON Energy Extension was not validated against the Energy ADE KIT profile due to the diverse philosophies behind the CityGML and CityJSON data models, and the content and focus of the two extensions. However, it can still be discussed that validation against the Energy ADE KIT profile would highly contribute to the usability of the Energy Extension. For this, a tool could be developed that provides conversion between CityGML + Energy ADE KIT profile files and CityJSON + Energy Extension files. This tool would need to ensure a valid conversion between the two files, while supporting a lossless transformation of data at the same time. Since the CityJSON Energy Extension includes certain attributes and relations that are not present in the Energy ADE KIT profile, these new features should be tackled carefully in such a conversion tool, so that the data stored with the new features is not lost when converted to a CityGML + Energy ADE KIT profile file.

A conversion tool between the CityJSON Energy Extension and the CityGML Energy ADE KIT profile could be utilised further to compare the two data models quantitatively. In this thesis, a comparison in file size was conducted only between the input and output CityJSON files to see the impact of the Extension objects and attributes. An additional comparison could be done between a CityJSON + Energy Extension file and a CityGML + Energy ADE KIT profile file, which would store exactly the same data. This analysis would demonstrate the impact of the two extensions with the used objects and attributes, which would provide a better understanding about the advantages and disadvantages of the two extensions.

## 7.4 Self-reflection

This section provides a discussion on the reproducibility of the developed methods and obtained results in this thesis, and includes an evaluation of the thesis process from a personal point of view.

For the first part of the methodology, namely the design of a CityJSON Energy Extension, all steps and justifications followed to create the Extension schema are described in the thesis report in a detailed way so that users can easily understand the logic behind each design decision. This is expected to help developers create new (complex) CityJSON Extensions as well, since various methods were tested for different functionalities, and documented in the thesis report. In addition, the CityJSON Energy Extension schema is openly available on GitHub: <https://github.com/ozgetufan/cjenergy>, where a short description of the Extension is provided as well, and guidelines on how to define and use each type of Extension property are described. As a result, the first part of the methodology, as well as its implementation, enables a high level of reproducibility, since the users can implement, test, and use the CityJSON Energy Extension themselves by following the documentation provided in the report and the GitHub repository.

The second part of the methodology, namely the space heating demand calculation, was implemented based on the openly available Dutch standard *NTA 8800*. While the required input data was mostly collected from open datasets, such as the BAG and meteorological data portal, the main data source of this thesis, namely the 3D city model of Rijssen-Holten, is not publicly available at the moment. Therefore, this dataset is not directly shared in the GitHub repository, which decreases the reproducibility in terms of input data. However, the characteristics of the 3D city model and the attributes it contains are described in detail in the thesis report to help users shape their own 3D city models in the same way to be able to perform the space heating demand calculation with the same steps. In addition, example input data (including a 3D city model) is provided in the GitHub repository and integrated in the Python scripts to enable the user to test the calculation pipeline. In terms of the pre-processing of input data, open tools were mostly used, but it was also benefited from licensed software, namely Safe Software FME, to calculate the missing physical properties of buildings from the 3D city model, such as the slope of surfaces and the perimeter of buildings. In terms of the calculation, the followed steps are described thoroughly in the thesis report, and the Python scripts created to implement the formulas are openly shared on the GitHub repository. Therefore, the acquisition and pre-processing of required input data provide limited reproducibility, while the calculation itself can be easily reproduced from the developed code.

While the space heating demand calculation implemented in this thesis is reproducible, the results of the calculation indicate a limitation. As discussed in [Section 6.4](#), negative energy demand values were detected for 32 buildings in the study area, while the solar gains calculation resulted in negative values as well. Even though various tests were performed to identify the problem, a solution could not be found in the given time frame of the thesis. However, two problem areas were detected which might be the reasons of negative values, namely the window ratio of surfaces and the extra heat flow parameter in solar gains, and additional tests were performed to see the impact in these areas. Since the detected problem areas are related to the used input data and the calculation method itself, the user can still take these into consideration when preparing and pre-processing the input data, and to use the calculation method correctly. Moreover, since the comparison of results with SimStadt software showed little difference between the obtained values for the majority of the dataset, it can be discussed that the reason behind negative values might be related to the used additional input data instead of the geometry of buildings or the calculation method. However, a comparison with ground truth data is still needed to verify this theory, since SimStadt is based on German standards and regulations. Overall, in terms of reproducibility and transparency, all conducted tests to solve this problem and a discussion about the results are provided in the thesis report to guide the readers about the possible future steps.

## 7 Conclusions and Future Work

From a personal perspective, I have found the space heating demand calculation part of the thesis more challenging than the creation of CityJSON Energy Extension. The calculation required a strong understanding of the specific calculation method for the Netherlands, which was available only in Dutch. I had to translate all relevant parts of the document to understand the details of each formula, which resulted in a significant time loss during my thesis. In addition, the collection of required input data was not trivial, since different datasets had to be used and the obtained data had to be pre-processed by using various tools. Moreover, I have spend a lot of time trying to fix the problem of negative values in the energy demand results. Even though I have had help, the collective efforts, unfortunately, did not result in a solution in the given time frame of the thesis. However, I still have gained many valuable skills during this part of the thesis, such as making research, finding valuable information from numerous articles, and being critical about the results I have obtained. Compared to this part, the design of CityJSON Energy Extension was more straightforward and enjoyable for me. The biggest challenge I have faced in this part was (occasionally) having to deal with errors coming from the existing tools of CityJSON. However, these problems were quickly solved after I have contacted the developers of these tools.



## A Comparison of the Extension elements

A comparison of the Energy ADE KIT profile and the CityJSON Energy Extension elements is provided in [Figure A.1](#). The objects and attributes that were directly translated to the CityJSON Energy Extension are not included in the figure, only changes between the two Extensions are emphasised.

## A Comparison of the Extension elements

Energy ADE KIT Profile	Original data type	CityJSON Energy Extension	New data type
<b>Core Module</b>			
<i>AbstractBuilding</i>			
buildingType	codelist	energy-buildingType	string
volume	VolumeType (type+volume+uom)	energy-volume	VolumeType(type+volume)
referencePoint	GM_Point	energy-referencePoint	string
floorArea	FloorArea (type+area+uom)	energy-floorArea	FloorArea (type+area)
heightAboveGround	HeightAboveGround (heightReference+value+uom)	energy-heightAboveGround	HeightAboveGround (heightReference+value)
-	-	<b>energy-function</b>	<b>Array of strings</b>
<b>WeatherData</b>			
weatherDataType	enumeration	energy-weatherElement	enumeration
-	-	<b>weatherDataLocation</b>	<b>string</b>
<b>EnergyDemand</b>			
maximumLoad	Measure (number+uom)	energy-maximumLoad	number
energyCarrierType	codelist	energy-energyCarrierType	string
<i>AbstractThermalZone</i>		-	
<i>AbstractUsageZone</i>		-	
<i>AbstractConstruction</i>		-	
<b>Building Physics Module</b>			
<b>ThermalZone</b>			
floorArea	FloorArea (type+area+uom)	energy-floorArea	FloorArea (type+area)
volume	VolumeType (type+volume+uom)	energy-volume	VolumeType(type+volume)
infiltrationRate	Measure (number+uom)	energy-infiltrationRate	number
-	-	<b>energy-perimeter</b>	<b>number</b>
<b>ThermalBoundary</b>			
azimuth	Angle (number+uom)	energy-azimuth	number
inclination	Angle (number+uom)	energy-inclination	number
area	Area (number+uom)	energy-area	number
-	-	<b>energy-orientation</b>	<b>string</b>
-	-	<b>energy-slope</b>	<b>number</b>
-	-	<b>energy-windowRatio</b>	<b>number</b>
<b>ThermalOpening</b>			
area	Area (number+uom)	energy-area	number
<b>Material Module</b>			
<b>Construction</b>			
uValue	Measure (number+uom)	energy-uValue	number
-	-	<b>energy-rValue</b>	<b>number</b>
-	-	<b>energy-gValue</b>	<b>number</b>
<b>LayerComponent</b>			
areaFraction	Scale (number+uom)	energy-areaFraction	number



## A Comparison of the Extension elements

Energy ADE KIT Profile	Original data type	CityJSON Energy Extension	New data type
thickness	Scale (number+uom)	energy-thickness	number
<b>Gas</b>			
rValue	Measure (number+uom)	energy-rValue	number
<b>SolidMaterial</b>			
conductivity	Measure (number+uom)	energy-conductivity	number
density	Measure (number+uom)	energy-density	number
permeance	Measure (number+uom)	energy-permeance	number
specificHeat	Measure (number+uom)	energy-specificHeat	number
<b>Supporting Classes Module</b>			
<b>RegularTimeSeries</b>			
temporalExtent	TM_Period	energy-temporalExtent	start date: string - end date: string
timeInterval	TM_IntervalLength	energy-timeInterval	Energy-Measure (value+uom)
<b>RegularTimeSeriesFile</b>			
file	URI	energy-file	string (format URI)
temporalExtent	TM_Period	energy-temporalExtent	start date: string - end date: string
timeInterval	TM_IntervalLength	energy-timeInterval	Energy-Measure (value+uom)
<b>Occupant Behaviour Module</b>			
<b>UsageZone</b>			
usageZoneType	codelist	energy-usageZoneType	string
floorArea	FloorArea (type+area+uom)	energy-floorArea	FloorArea (type+area)
-	-	<b>energy-numberOfResidentialFunctions</b>	<b>number</b>

**Figure A.1:** A comparison of the Energy ADE KIT profile and CityJSON Energy Extension elements. Blue rows represent the newly added elements in the CityJSON Energy Extension, and orange rows indicate the elements that were not included in the CityJSON Energy Extension.

# Bibliography

- Agugiaro, G. (2016a). Enabling “energy-awareness” in the semantic 3D city model of Vienna. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-4/W1.
- Agugiaro, G. (2016b). Energy planning tools and CityGML-based 3D virtual city models: experiences from Trento (Italy). *Applied Geomatics*, 8(1):41–56.
- Agugiaro, G., Benner, J., Cipriano, P., and Nouvel, R. (2018). The Energy Application Domain Extension for CityGML: enhancing interoperability for urban energy simulations. *Open Geospatial Data, Software and Standards*, 3(1):1–30.
- Agugiaro, G., Hauer, S., and Nadler, F. (2015). Coupling of CityGML-based semantic city models with energy simulation tools: some experiences. pages 191–200. REAL CORP 2015 Proceedings/-Tagungsband.
- Agugiaro, G. and Holcik, P. (2017). *3D City Database extension for the CityGML Energy ADE 0.8 (PostgreSQL Version)*.
- Ali, U., Shamsi, M. H., Hoare, C., Mangina, E., and O’Donnell, J. (2021). Review of urban building energy modeling (UBEM) approaches, methods and tools using qualitative and quantitative analysis. *Energy and Buildings*, 246:111073.
- Beckman, W. A., Broman, L., Fiksel, A., Klein, S. A., Lindberg, E., Schuler, M., and Thornton, J. (1994). TRNSYS The most complete solar energy system modeling and simulation software. *Renewable Energy*, 5(1):486–488.
- Benner, J. (2018). CityGML Energy ADE V. 1.0 Specification. [http://www.citygmlwiki.org/images/3/38/Energy\\_ADE\\_specification\\_2018\\_03\\_25.pdf](http://www.citygmlwiki.org/images/3/38/Energy_ADE_specification_2018_03_25.pdf).
- Benner, J., Geiger, A., and Häfele, K.-H. (2016). Virtual 3D City Model Support for Energy Demand Simulations on City Level – The CityGML Energy Extension. *REAL CORP 2016 – SMART ME UP! How to become and how to stay a Smart City, and does this improve quality of life? Proceedings of 21st International Conference on Urban Planning, Regional Development and Information Society*, pages 777–786.
- Biljecki, F., Kumar, K., and Nagel, C. (2018). CityGML Application Domain Extension (ADE): overview of developments. *Open Geospatial Data, Software and Standards*, 3(1):13.
- Biljecki, F., Ledoux, H., and Stoter, J. (2016). An improved LOD specification for 3D building models. *Computers, Environment and Urban Systems*, 59:25–37.
- Biljecki, F., Lim, J., Crawford, J., Moraru, D., Tauscher, H., Konde, A., Adouane, K., Lawrence, S., Janssen, P., and Stouffs, R. (2021). Extending CityGML for IFC-sourced 3D city models. *Automation in Construction*, 121:103440.
- Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S., and Çöltekin, A. (2015). Applications of 3D City Models: State of the Art Review. *ISPRS International Journal of Geo-Information*, 4(4):2842–2889.
- Bray, T., Hollander, D., Layman, A., Tobin, R., and Thompson, H. (2009). Namespaces in XML 1.0 (Third Edition), W3C Recommendation. <https://www.w3.org/TR/xml-names/>.

## Bibliography

- Cerný, R. and Kocí, V. (2015). Traditional fired-clay bricks versus large and highly perforated fired-clay bricks masonry: Influence on buildings thermal performance. pages 63–81. Woodhead Publishing, Oxford.
- Crawley, D. B., Lawrie, L. K., Winkelmann, F. C., Buhl, W. F., Huang, Y. J., Pedersen, C. O., Strand, R. K., Liesen, R. J., Fisher, D. E., Witte, M. J., and Glazer, J. (2001). EnergyPlus: creating a new-generation building energy simulation program. *Energy and Buildings*, 33(4):319–331.
- Dalla Mora, T., Teso, L., Carnieletto, L., Zarrella, A., and Romagnoni, P. (2021). Comparative Analysis between Dynamic and Quasi-Steady-State Methods at an Urban Scale on a Social-Housing District in Venice. *Energies*, 14(16):5164.
- Dukai, B. and Ledoux, H. (2021). CityJSON Specifications 1.1.0. <https://www.cityjson.org/specs/1.1.0/>.
- Ferrando, M. and Causone, F. (2019). An overview of urban building energy modelling (UBEM) tools. In *16th IBPSA International Conference and Exhibition*. International Building Performance Simulation Association.
- Ghiassi, N. and Mahdavi, A. (2017). Reductive bottom-up urban energy computing supported by multivariate cluster analysis. *Energy and Buildings*, 144:372–386.
- Gröger, G., Kolbe, T. H., Nagel, C., and Häfele, K.-H. (2012). OGC City Geography Markup Language (CityGML) Encoding Standard. [https://portal.ogc.org/files/?artifact\\_id=47842](https://portal.ogc.org/files/?artifact_id=47842).
- Gröger, G. and Plümer, L. (2012). CityGML – Interoperable semantic 3D city models. *ISPRS Journal of Photogrammetry and Remote Sensing*, 71:12–33.
- Ilizirov, G. and Dalyot, S. (2022). Data Quality Extension. <https://github.com/TheGreatWizard/data-quality>.
- INSPIRE Thematic Working Group Buildings (2013). D2.8.III.2 INSPIRE Data Specification on Buildings – Technical Guidelines. Technical report, INSPIRE Infrastructure for Spatial Information in Europe.
- Kadaster (2022). Basisregistraties Adressen en Gebouwen (BAG). <https://www.kadaster.nl/zakelijk/producten/adressen-en-gebouwen/bag-2.0-extract>.
- Kaden, R. and Kolbe, T. H. (2014). Simulation-based total energy demand estimation of buildings using semantic 3D city models. *International Journal of 3-D Information Modeling (IJ3DIM)*, 3(2):35–53.
- Kippers, R., Koeva, M., Keulen, M., and Oude Elberink, S. (2021). Automatic 3D Building Model Generation Using Deep Learning Methods Based On CityJSON and 2D Floor Plans. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLVI-4/W4-2021:49–54.
- Kolbe, T. (2009). Representing and Exchanging 3D City Models with CityGML. In *3D Geo-Information Sciences*, pages 15–31.
- Labetski, A., Kumar, K., Ledoux, H., and Stoter, J. (2018). A metadata ADE for CityGML. *Open Geospatial Data, Software and Standards*, 3(1):16.
- Ledoux, H., Ogori, K. A., Kumar, K., Dukai, B., Labetski, A., and Vitalis, S. (2019). CityJSON: A compact and easy-to-use encoding of the CityGML data model. *Open Geospatial Data, Software and Standards*, 4(1):1–12.

## Bibliography

- León-Sánchez, C., Giannelli, D., Agugiaro, G., and Stoter, J. (2021). Testing the New 3D BAG Dataset for Energy Demand Estimation of Residential Buildings. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLVI-4/W1-2021:69–76.
- Monien, D., Strzalka, A., Koukofikis, A., Coors, V., and Eicker, U. (2017). Comparison of building modelling assumptions and methods for urban scale heat demand forecasting. *Future Cities and Environment*, 3(0):2.
- Nys, G.-A., Kharroubi, A., Poux, F., and Billen, R. (2021). An Extension of CityJSON to Support Point Clouds. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 42:301–306.
- Nys, G.-A., Poux, F., and Billen, R. (2020). CityJSON Building Generation from Airborne LiDAR 3D Point Clouds. *ISPRS International Journal of Geo-Information*, 9(9):521.
- Pasquinelli, A., Agugiaro, G., Tagliabue, L. C., Scaioni, M., and Guzzetti, F. (2019). Exploiting the Potential of Integrated Public Building Data: Energy Performance Assessment of the Building Stock in a Case Study in Northern Italy. *ISPRS International Journal of Geo-Information*, 8(1):27.
- Portele, C. (2007). OpenGIS® Geography Markup Language (GML) Encoding Standard. Version 3.2.1. [https://portal.ogc.org/files/?artifact\\_id=20509](https://portal.ogc.org/files/?artifact_id=20509).
- Pratavia, E., Romano, P., Carnieletto, L., Pirotti, F., Vivian, J., and Zarrella, A. (2021). EURECA: An open-source urban building energy modelling tool for the efficient evaluation of cities energy demand. *Renewable Energy*, 173:544–560.
- Radwan, Y. M. G. (2021). Automation of Building Energy Efficiency Using BIM: A decision support BIM-based tool to optimize the EI of buildings. Master's thesis, Eindhoven University of Technology.
- Robinson, D., Haldi, F., Leroux, P., Perez, D., Rasheed, A., and Wilke, U. (2009). CITYSIM: Comprehensive Micro-Simulation of Resource Flows for Sustainable Urban Planning.
- Rosknecht, M. and Airaksinen, E. (2020). Concept and Evaluation of Heating Demand Prediction Based on 3D City Models and the CityGML Energy ADE—Case Study Helsinki. *ISPRS International Journal of Geo-Information*, 9(10):602.
- Royal Netherlands Standardization Institute (2022). NTA 8800:2022 Energy performance of buildings - Determination method. <https://www.nen.nl/nta-8800-2022-nl-290717>.
- Sanhudo, L., Ramos, N. M. M., Poças Martins, J., Almeida, R. M. S. F., Barreira, E., Simões, M. L., and Cardoso, V. (2018). Building information modeling for energy retrofitting – A review. *Renewable and Sustainable Energy Reviews*, 89:249–260.
- Scartezzini, J.-L., Nouvel, R., Brassel, K.-H., Bruse, M., Duminil, E., Coors, V., Eicker, U., and Robinson, D. (2015). SimStadt, a new workflow-driven urban energy simulation platform for CityGML city models.
- Skarbal, B., Peters-Anders, J., Faizan Malik, A., and Agugiaro, G. (2017). How to Pinpoint Energy-Inefficient Buildings? AN Approach Based on the 3d City Model of Vienna. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 44W3:71–78. ADS Bibcode: 2017IS-PAn44W3...71S.
- Swan, L. G. and Ugursal, V. I. (2009). Modeling of end-use energy consumption in the residential sector: A review of modeling techniques. *Renewable and Sustainable Energy Reviews*, 13(8):1819–1835.

## Bibliography

- TABULA (2012). TABULA: Typology Approach for Building Stock Energy Assessment. <https://episclope.eu/iee-project/tabula/>.
- United Nations Environment Programme (2020). 2020 Global Status Report for Buildings and Construction: Towards a Zero-Emission, Efficient and Resilient Buildings and Construction Sector. Technical report.
- Van den Brom, P. (2020). *Energy in Dwellings: A comparison between Theory and Practice*. PhD thesis, Delft University of Technology.
- Verwiebe, P. A., Seim, S., Burges, S., Schulz, L., and Müller-Kirchenbauer, J. (2021). Modeling Energy Demand—A Systematic Literature Review. *Energies*, 14(23):7859.
- Vitalis, S., Arroyo Ohori, K., and Stoter, J. (2019a). Incorporating Topological Representation in 3D City Models. *ISPRS International Journal of Geo-Information*, 8(8).
- Vitalis, S., Labetski, A., Arroyo Ohori, K., Ledoux, H., and Stoter, J. (2019b). A Data Structure to Incorporate Versioning in 3D City Models. In *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume IV-4-W8, pages 123–130. Copernicus GmbH.
- Wilhelm, L., Donaubaauer, A., and Kolbe, T. H. (2021). Integration of BIM and Environmental Planning: The CityGML EnvPlan ADE. *Journal of Digital Landscape Architecture*.
- Wu, J. (2021). Automatic building permits checks by means of 3D city models. Master's thesis, Delft University of Technology.
- Yao, Z., Nagel, C., Kunde, F., Hudra, G., Willkomm, P., Donaubaauer, A., Adolphi, T., and Kolbe, T. H. (2018). 3DCityDB - a 3D geodatabase solution for the management, analysis, and visualization of semantic 3D city models based on CityGML. *Open Geospatial Data, Software and Standards*, 3(1):5.

## **Colophon**

This document was typeset using L<sup>A</sup>T<sub>E</sub>X, using the KOMA-Script class `scrbook`. The main font is Palatino.

