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

Tufan, Ö; Arroyo Ohori, K.; León-Sánchez, C.; Agugiaro, G.; Stoter, J.

DOI
10.5194/isprs-archives-XLVIII-4-W4-2022-169-2022

Publication date
2022

Document Version
Final published version

Published in
International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives

Citation (APA)
Tufan, Ö., Arroyo Ohori, K., León-Sánchez, C., Agugiaro, G., & Stoter, J. (2022). Development and Testing of the CityJSON Energy Extension for Space heating Demand Calculation. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, *48*(4/W4-2022), 169-176. https://doi.org/10.5194/isprs-archives-XLVIII-4-W4-2022-169-2022

DEVELOPMENT AND TESTING OF THE CITYJSON ENERGY EXTENSION FOR SPACE HEATING DEMAND CALCULATION

Ö. Tufan[1,*], K. Arroyo Ohori[2], C. León-Sánchez[2], G. Agugiaro[2], J. Stoter[2]

[1] Delft University of Technology, Faculty of Architecture and the Built Environment, the Netherlands - ozgetufann97@gmail.com
[2] 3D Geoinformation group, Delft University of Technology, Faculty of Architecture and the Built Environment, the Netherlands - ken@ken.mx, (c.a.leonsanchez, g.agugiaro, j.e.stoter)@tudelft.nl

Commission IV, WG IV/9

KEY WORDS: 3D City Modelling, CityGML, CityJSON, Urban Energy Modelling, Energy ADE, Space Heating Demand.

ABSTRACT:

3D city models are frequently used to acquire and store energy-related information of buildings for energy applications. In this context, CityGML is the most common data model, and the Energy ADE, one of its most complex extensions, provides a systematic way of storing detailed energy-related data in XML format. Contrarily, even though CityGML's JSON-based encoding, CityJSON, has an extension mechanism, an energy-related CityJSON Extension is missing. This paper, therefore, presents the first results of the development of a CityJSON Energy Extension and space heating demand calculation is utilized as the use case. The simplified version of the Energy ADE, called the Energy ADE KIT profile, is used to create a semi-direct translation to the CityJSON Energy Extension. This Extension is then validated through the official validator of CityJSON and the use case, and improvements are made considering the validation results. The space heating demand is calculated according to the Dutch standard NTA 8800 for a subset of Rijssen-Holten in the Netherlands although the solar gains calculation requires further review. The results show that the final CityJSON Energy Extension provides full support for space heating demand calculations based on the NTA 8800 and eliminates the deep hierarchical structure of the Energy ADE. A comparison on CityJSON file sizes shows a 25.2 MB increase after the required input data is stored in a CityJSON + Energy Extension file, which is not significant considering the high amount of data stored in the file. Overall, this paper shows that the CityJSON Energy Extension could provide an easy-to-use alternative to the CityGML Energy ADE.

1. INTRODUCTION

The energy performance of buildings is a prevalent discussion all around the world, and *Urban Energy Modelling* has gained substantial importance over the years to model the current energy use of buildings and to predict future scenarios, in which semantic 3D city models are frequently used to obtain and store energy-related data in urban areas (Agugiaro, 2016).

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. While CityGML covers a wide variety of cityobjects with their geometries and semantics, the core data model can be extended for specific use cases with the concept of Application Domain Extensions (ADEs), 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 (Agugiaro et al., 2018). This information can then be used in both steady-state and dynamic energy simulations (León-Sánchez et al., 2021) to analyse the current state of buildings as well as to obtain future predictions.

While CityGML and the Energy ADE are used for numerous applications, the complexity and verbosity of GML and XML in general have led to the development of a JSON-based encoding, namely CityJSON, which consists of a subset of the CityGML data model and aims to provide a more efficient way of storing information (Ledoux et al., 2019). CityJSON has its own Extension mechanism as well, which in contrast to CityGML ADEs, requires the extensions to meet stricter guidelines so that they are more similar to regular CityJSON files in terms of structure, and therefore existing software tools can process them without requiring specific support for every Extension.

Our findings show that an energy-related CityJSON Extension is still missing, since only a draft by (Ledoux and Kumar, 2018) has been developed so far. The compact structure of CityJSON without deep hierarchies is believed to 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 paper presents the first results of a research to develop and test a CityJSON Energy Extension, focusing on the space heating demand calculation as the use case.

2. THEORETICAL BACKGROUND

2.1 CityJSON's Extension mechanism

The Extension mechanism of CityJSON consists of three operations: creating new attributes, new CityObjects, and new root properties, which are included as three properties in the Extension schema (Figure 1). 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

* Corresponding author

recommended for all Extension object names to start with a "+" sign, and the object type must be included.

Even though CityJSON is a rather new encoding, 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 cityobjects (Vitalis et al., 2019), from storing information on building permits (Wu, 2021) to the mapping of the CityGML Noise ADE (CityJSON Development Team, 2021).

An analysis of these Extensions show that while basic operations, such as adding new CityObjects and attributes, can be easily done, it is not possible to define more complex elements and relations in a straightforward way, since these are not supported in the Extension mechanism of CityJSON. For instance, to extend an existing CityObject with new properties, a completely new one must be created since CityJSON does not support inheritance (Ledoux et al., 2019).

```
{
  "type": "CityJSONExtension",
  "name": "Traffic",
  "description": "Extension to model the traffic",
  "uri": "https://someurl.org/traffic.ext.json",
  "version": "1.0",
  "versionCityJSON": "1.1",
  "extraAttributes": {},
  "extraCityObjects": {},
  "extraRootProperties": {}
}
```

Figure 1. A basic CityJSON Extension schema.

2.2 Energy ADE

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 the creation of future refurbishment scenarios. The Energy ADE is designed in six thematic modules with different functionalities, each one focusing on a distinct aspect of buildings (Figure 2). While the broad scope of the Energy ADE and its detailed modularisation contribute to its use in a wide variety of applications, this complexity is usually not needed for simple applications. Thus, the Karlsruhe Institute of Technology created a subset for (semi) steady-state calculations, called the Energy ADE KIT profile, which is consequently easier to use (León-Sánchez et al., 2021).

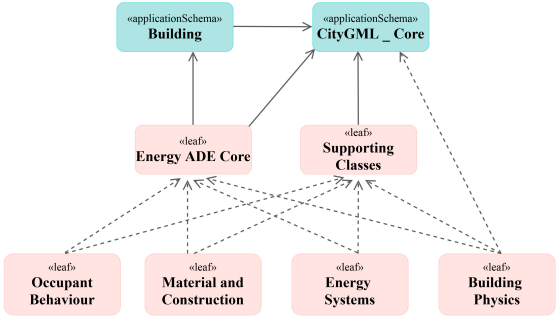


Figure 2. Modules and dependencies of the Energy ADE. Figure adapted from (Benner, 2018).

2.3 3D city models for energy applications

Steady-state models are one of the most frequently used approaches in Urban Energy Modelling for city-scale space heating demand calculations. The popularity of these models comes from the fact that they require less input parameters and simplifications can be used for the missing input data (Pasquinelli et al., 2019). Moreover, energy demand values calculated with steady-state models are presented with a low temporal resolution, such as monthly or annual values.

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, 2016) uses an energy balance method that follows the Italian standards to calculate monthly space heating demand values for the city of 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. 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. Later studies make use of the CityGML Energy ADE as well to calculate and store the input parameters. (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 again stored with the Energy ADE.

Well-known energy simulation tools make use of 3D city models as well, such as SimStadt (Duminil et al., 2021), which uses a steady-state model based on the German standard DIN V 18599 for automatically calculating monthly energy demand of buildings (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 built-in libraries (León-Sánchez et al., 2021).

3. METHODOLOGY

The methodology comprises two interrelated parts: the development of the CityJSON Energy Extension, and the space heating demand calculation as the use case to test and improve the Extension.

In the first step of the methodology, a semi-direct translation[1] from the Energy ADE KIT profile to a CityJSON Energy Extension was created to explore the possibilities and grasp the similarities and differences between the two data models. The KIT profile was preferred instead of the full Energy ADE, since the latter provides a much more complex option than needed for the use case. Moreover, the mapping of new CityObjects, attributes, non-CityObjects, data types, enumerations, and relations between objects were considered in this step. The created Extension was then validated first with *cjval*, the official validator of CityJSON, then through the use case in the second step. A CityJSON + Energy Extension file was created with all required

[1] Objects are adjusted as minimum as they are required in order to be compliant with the JSON or CityJSON schemas

input data to detect the limitations of the semi-direct translation, and improvements were made to eliminate the shortcomings of the Extension mechanism.

In the second part, space heating demand was calculated for a subset of the municipality of Rijssen-Holten in the Netherlands, consisting of an area of 3318 buildings. The calculation is based on the Dutch standard *NTA 8800* with a steady-state energy balance method (Royal Netherlands Standardization Institute, 2020), which considers the monthly heat losses and heat gains in a calculation zone (Equation 1). The required input data was retrieved from the CityJSON + Energy Extension file created in the previous step to be used in the calculations, and the resulting energy demand values were again stored in the same file.

$$Q_{H;nd} = (Q_{H;tr} + Q_{H;ve}) - n_{H;gn}(Q_{H;int} + Q_{H;sol}) \quad (1)$$

where $Q_{H;nd}$ = space heating demand
$Q_{H;tr}$ = heat losses through transmission
$Q_{H;ve}$ = heat losses through ventilation
$n_{H;gn}$ = dimensionless utilisation factor
$Q_{H;int}$ = internal gains
$Q_{H;sol}$ = solar gains

4. DATASETS AND PRE-PROCESSING

The main data source for this research was the 3D city model of Rijssen-Holten in CityGML format, which is a testbed for energy applications currently under development at the 3D Geoinformation Group at TU Delft[2] (Figure 3). In this model, each building includes a number of attributes beneficial for the use case, such as the year of construction, building function and typology, footprint area, building volume, and number of storeys. In addition, the LoD2 geometries are used, in which the buildings' surfaces are modelled as three types of *BoundarySurfaces*, namely *WallSurface*, *RoofSurface*, and *GroundSurface*. For each surface, generic attributes about the area, direction, inclination, and azimuth are included.

For pre-processing, a Safe Software FME Workbench was used to compute the missing geometrical information, namely the slope of surfaces and perimeter of buildings. Then, the model was converted from CityGML to CityJSON using *citygml-tools* (citygml-tools Development Team, 2022), and data cleaning was performed in Python to eliminate buildings with missing data.

Another data source was the BAG dataset (Basisregistratie Adressen en Gebouwen), from which the data on usable areas in buildings and number of residential units were obtained (Kadaster, 2022). In addition, the TABULA building physics library was used to retrieve the U-value, g-value and window ratio, which were provided for four building typologies (single-family house, multi-family house, terrace house, apartment block) and six construction periods (till 1964, 1965 – 1974, 1975 – 1991, 1992 – 2005, 2006 – 2014, 2015 – today) for the Netherlands. This data was then stored in a database to be later used during the calculations. Finally, the weather data was obtained from the (Meteorological Data Portal, 2022) developed at TU Delft and the Dutch standard *NTA 8800* itself. Monthly outdoor air temperature, solar irradiation, visibility factor, and shading reduction factor data were stored in the database as well to ensure easy access to data.

[2] `https://3d.bk.tudelft.nl`

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

5. IMPLEMENTATION

5.1 Mapping rules of the semi-direct translation

All excerpts of the CityJSON schema and the CityJSON file presented in this section correspond to the results of the first semi-direct translation. In order to create this translation, distinct mapping rules were determined for different types of objects in the Energy ADE KIT profile. First, new CityObjects are defined in a straightforward way with the "extraCityObjects" property of CityJSON's Extension mechanism (Figure 4), and parent/children properties were used to map the relations among CityObjects. However, abstract classes were omitted since CityJSON does not support inheritance and discourages deep hierarchies in the schema. Therefore, the "allOf" keyword of the JSON schema was used to represent the relationship between *AbstractCityObject* and newly defined CityObjects. Second, new attributes were defined for existing CityObjects with the "extraAttributes" property (Figure 5).

```
"extraCityObjects": {
  "+UsageZone": {
    "allOf": [
      {
      "$ref":"cityobjects.
         schema.json#/
         _AbstractCityObject"
      },
      {
        "properties": {...}
      }
    ]
  }
}
```

(a)

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

(b)

Figure 4. A new CityObject (a) defined in the Extension schema, (b) used in a CityJSON file.

```
"extraAttributes": {
  "Building": {
    "+buildingType": {...},
    "+constructionWeight":
       {...},
    "+volume": {...},
    "+floorArea": {...},
    "+heightAboveGround": {...}
  }
}
```

(a)

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

(b)

Figure 5. Extra attributes for Building objects (a) defined in the Extension schema, (b) used in a CityJSON file.

Next, new data types were defined with the "definitions" keyword of the JSON schema so that these could be referenced multiple times, and the "enum" keyword was used to define new enumerations as presented in Figure 6. Code lists were omitted, since these are not a fixed list of values, but were instead specified in the documentation of the CityJSON Energy Extension.

```
"definitions": {
  "floorArea": {
    "type": "object",
    "properties": {
      "type": {
        "enum": ["netFloorArea",
            "grossFloorArea", "
           energyReferenceArea"]
      },
      "value": {...}
    }
  }
}
```
(a)

```
"Build1": {
  "type": "Building",
  "geometry": [...],
  "attributes": {
    "+buildingType": "
       singleFamily",
    ...
    "+floorArea": {
      "type": "netFloorArea",
      "value": {...}
    }
  }
}
```
(b)

Figure 6. New data types and enumerations (a) defined in the Extension schema, (b) used in a CityJSON file.

Conversely, non-CityObjects could not be created in a straightforward manner, because these are not supported in CityJSON extensions. Therefore, they were defined under the "extraCityObjects" property, even though they are not subclasses of *AbstractCityObject* in the CityGML data model. Furthermore, to map the relationships between non-CityObjects, additional attributes were created to store the object IDs (Figure 7).

```
"extraCityObjects": {
  "+EnergyDemand": {
    "type": "object",
    "properties": {
      "type": {...},
      "attributes": {
        "type": "object",
        "properties": {
          "energyAmount": {...},
          "endUse": {...},
          "maximumLoad": {...},
          "energyCarrierType": {...}
        }
      ...
}
```
(a)

```
"Demand1": {
  "type": "+EnergyDemand",
  "attributes": {
    "endUse": "spaceHeating",
    ...
  }
},
"Build1": {
  "type": "Building",
  "attributes": {
    "+buildingType": "singleFamily",
    ...
    "+energyDemand":["Demand1"]
  }
}
```
(b)

Figure 7. New non-CityObject (a) defined under "extraCityObjects" in the Extension schema, (b) used in a CityJSON file with a relationship to a Building object.

5.2 Experiments on data storage and improvements

After creating the initial mapping, the required input data for space heating demand calculation was stored on a CityJSON + Energy Extension file to test its suitability for the use case. Figure 8 demonstrates the CityJSON Energy Extension objects and attributes used to store the required input data. However, during this step we found that certain data could not be stored, as the attributes were not supported (shown in red). Therefore, the new attribute was defined in the schema so that the Extension could fully support the use case.

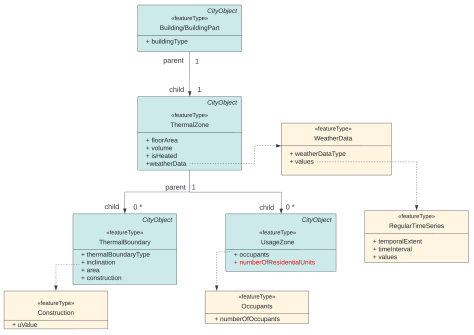


Figure 8. Used objects and attributes to store the required input data in the CityJSON + Energy Extension file

Next, the space heating demand was calculated by retrieving the input data from the CityJSON + Energy Extension file. In this step, certain limitations of the semi-direct translation and opportunities for improvements were detected. The first limitation concerned the relations among objects. In the semi-direct translation, all relations between CityObjects were defined with parent/children properties, which resulted in ambiguity for certain objects. For instance, the children of ThermalZone objects could be both ThermalBoundary and UsageZone objects, resulting in an extra step during the space heating demand calculation to check the object types of the children.

To eliminate this ambiguity, the relations between ThermalZone, ThermalBoundary and ThermalOpening objects were instead ensured with additional properties (Figure 9). This way, it was made sure that all parent/children properties stored only one type of object.

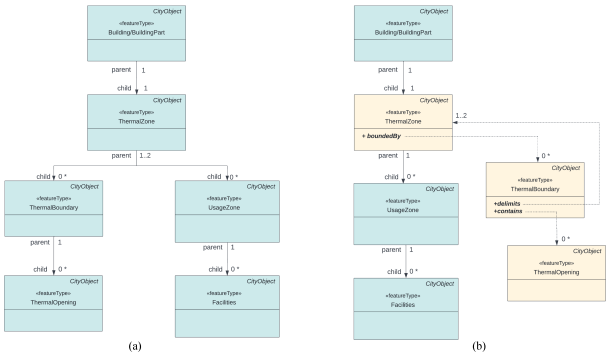


Figure 9. Relations among CityObjects (a) in the semi-direct translation, and (b) in the final version of the Extension.

The second consideration was about the storage of the relations between objects. In the semi-direct translation, parent/children "properties" and additional "attributes" were used. Since CityJSON properties are stored one level higher in the hierarchy than attributes, it was decided to store the additional attributes as properties to decrease the number of steps to reach the data (Figure 10), which ensured faster data retrieval.

```
"Zone1": {
  "type": "+ThermalZone",
  "attributes": {
    "floorArea": [...],
    "isHeated": True,
    "boundedBy": ["Boundary1", ...]
  },
  "parents": "Building1",
  "children": "UsageZone1"
}
```
(a)

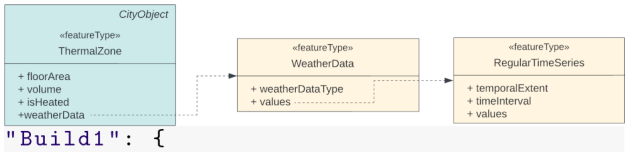
```
"Zone1": {
  "type": "+ThermalZone",
  "attributes": {
    "floorArea": [...],
    "isHeated": True
  },
  "parents": "Building1",
  "children": "UsageZone1",
  "boundedBy": ["Boundary1", ...]
}
```
(b)

Figure 10. Storage of the "boundedBy" relation (a) as an additional attribute, and (b) an additional property.

Another limitation was the deep hierarchical structures inherited from the KIT profile, and example to which was the WeatherData object. In the semi-direct translation, WeatherData was defined as a JSON object, which resulted in numerous steps to reach the data. For instance, to reach the outdoor air temperature data stored for each building, the following steps had to be taken:

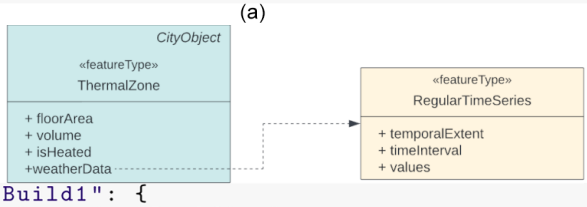
1. Obtain WeatherData object ID from the "+weatherData" attribute.
2. Navigate to the WeatherData object and obtain the RegularTimeSeries object ID from the "values" attribute.
3. Navigate to the RegularTimeSeries object and obtain the actual data from the "values" attribute.

To simplify this hierarchical structure in the final Extension, WeatherData was stored instead as a subschema, which enabled the user to obtain the RegularTimeSeries object ID directly from the "+weatherData" attribute of a Building (Figure 11). As a result, faster data retrieval with a simpler hierarchy was ensured.




```
"Build1": {
  "type": "Building",
  "geometry": [...],
  "attributes": {
    "+buildingType": "singleFamily",
    ...
    "+weatherData":["OutdoorTemperature"]
  }
}
```

(a)




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

(b)

Figure 11. Storage of WeatherData (a) as a JSON object, and (b) a subschema.

Yet another limitation is related to the duplicate information stored in the CityJSON + Energy Extension file, resulting from the storage of units of measurement (uom) for each numerical attribute. For instance, the unit m^3 was stored for each building volume attribute, even though the unit was the same for each of the 3318 buildings in the area. We create an extra root property called "+UnitOfMeasurement" to enable the storage of the uom only once per file, instead of storing it for each numerical attribute separately (Figure 12).

```
{
  "type": "CityJSON",
  "version": "1.1",
  "extensions": {"Energy": {...}},
  "transform": {...},
  "CityObjects": {...},
  "vertices": [...],
  "+unitOfMeasurement": {
    "volume": "m^3",
    "floorArea": "m^2",
    "energyDemand": "kWh",
    ...
  }
}
```

Figure 12. Definition of a root property to specify the used uom.

The final improvement considered the naming conventions in the Extension schema. An "energy" prefix was added to all Extension objects to differentiate them from any other CityJSON extensions, since namespaces are not supported in CityJSON.

5.3 Space heating demand calculation

After the input data was stored in a CityJSON + Energy Extension file, space heating demand was calculated with the following assumptions and simplifications:

1. Only residential and mixed-use buildings in the study area were considered in a heating period of October to March (inclusive).
2. A constant monthly value of 18 °C was assumed for indoor air temperature during the heating period, obtained from (Van den Brom, 2020).
3. Each building was modelled as a single thermal zone since the 3D city model in LoD2 did not contain data on the interior of buildings.
4. 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.
5. Heat losses between residential and/or mixed-use buildings were neglected.
6. Only the wall surfaces larger than 4 m^2 were considered for the calculation of solar gains through windows.

Once the space heating demand was calculated, the CityJSON + Energy Extension file was enhanced with the resulting values. 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.

6. RESULTS AND ANALYSIS

6.1 Energy ADE KIT profile vs. CityJSON Energy Extension

The Energy ADE KIT profile was the starting point for this thesis however, this proposal for the CityJSON Energy Extension differentiate from the Energy ADE by the compliance of the CityJSON specification. The CityJSON Energy Extension provides a simpler schema with a less hierarchical structure by benefiting from the simplicity of JSON compared to XML and by reconsidering the deep hierarchies in the schema. With the scope of ensuring a fast data retrieval, we consider that the fewer hierarchical levels the less computational steps to access the values.

Another 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, as can be seen in the Energy ADE, the extension mechanism of CityJSON is only limited with defining new CityObjects, attributes and root properties. Therefore, 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 argued that it resulted in less clarity in the Extension schema, since a separate property for defining non-CityObjects was not present.

6.2 Comparison on file size

To analyse the impact of the CityJSON Energy Extension, the size of input, intermediate, and output files was compared, as demonstrated in Table 1. A comparison between the input 3D city model in CityJSON and the output CityJSON + Energy Extension file shows an increase of 25.2 MB in size. It can be discussed that this increase is not significant, considering that around 100 000 more CityJSON objects are stored in the output file to maintain all input and output data.

	3DCM in CityGML	3DCM in CityJSON	CityJSON + Energy Extension (only input data)	CityJSON + Energy Extension (output file)
No. of objects	3.318	3.318	106.848	108.732
File size	165 MB	40.6 MB	63.7 MB	65.8 MB

Table 1. File size and the number of objects stored in each input/output file

It is important to note that a comparison on file size between the Energy ADE KIT profile and the CityJSON Energy Extension was not performed, since the input 3D city model in CityGML did not include any KIT profile elements. However, this information still provides valuable insight when compared with the final output CityJSON file.

It can be seen that the input CityGML file with 3318 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 108.732 objects to store all needed input data takes up only 65.8 MB of space. This result illustrates 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.

6.3 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 either because they had non-residential functions or lacked required input data. The resulting values for January are displayed in Figure 13 for Rijssen downtown and a residential area.

It is crucial to evaluate the impact of specific building characteristics on the resulting energy demand. First, Figure 14 shows the monthly average energy demand values for each building typology. While the TABULA building classification identifies four types of houses, 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 that the monthly average energy demand is highest for the multi-family house category, while single-family and terrace houses have similar values, with monthly averages of around 5000 kWh or lower. In addition, Table 2 shows the average building volumes for each of the three building typologies in the study area. The average volume is considerably high for multi-family houses, which justifies the calculated energy demand values as well, since multi-family houses with largest average volume have the highest monthly average energy demand values.

Figure 13. Energy demand (kWh) in the month January in the center (top) and a residential area (bottom) in Rijssen.

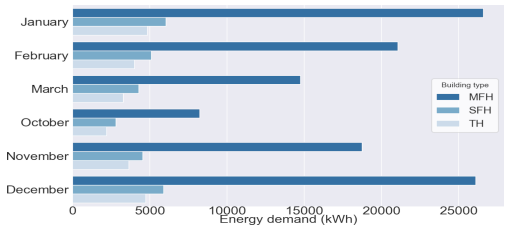


Figure 14. Average monthly energy demand (kWh/month) for each building typology

Building Type	Average building volume (m^3)
Multi Family House	7694,98
Single Family House	592,35
Terrace House	537,21

Table 2. Average volume of the three building typologies in the study area.

Second, the monthly average energy demand values for October to March and the overall distribution of values are demonstrated in Figure 15, which are grouped by construction periods that were determined in the TABULA building physics library. It can be seen from the graphs that the energy demand values follow a logical order, where the demand is higher in colder months compared to the warmer periods of the year. In addition, it is seen that the average energy demand values are higher for older construction epochs, which is expected since old houses

usually were built with outdated practices. However, the overall distribution shows many outliers on the higher side of the aggregation of values, which results in extremely high average values for some cases. In addition, a negative average energy demand value can be detected in October, which is neither expected nor possible for this calculation.

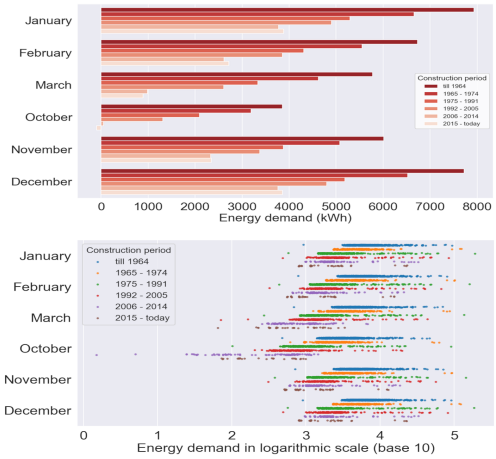


Figure 15. Average monthly energy demand (kWh) (top) and the distribution of values on logarithmic scale (bottom) for each construction period.

To investigate the negative values and to validate the overall results, the energy demand values were compared with the results obtained from the SimStadt software for the same study area. Figure 16 presents the monthly average space heating demand values calculated with the NTA 8800 standard as part of this research and the SimStadt software for a 10-building subset from the study area. It can be seen that the difference between the values are small ($< \ 10\%$) and our results are usually higher than the obtained with SimStadt.

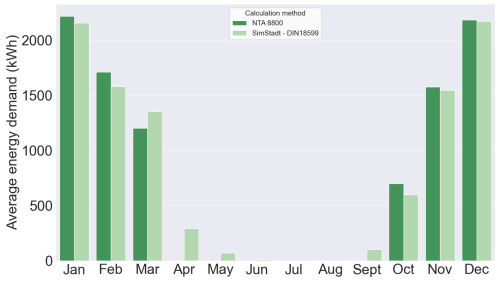


Figure 16. The differences between the average energy demand values calculated with NTA 8800 and SimStadt for a 10-building subset.

While the comparison between NTA 8800 and SimStadt results help to justify the space heating demand values, it is important to consider that the SimStadt software uses the German standard DIN V 18599 and the built-in libraries for the building physics definition for Germany for its calculation, while the NTA 8800 standard is used specifically in the Netherlands. Therefore, this comparison should not be seen as a way of validating the NTA 8800 results, since the used data and the calculation methods are different.

A comparison of the results also showed that the negative values obtained in this research were not present in the SimStadt's. Therefore, a more detailed analysis was made and it was found that 32 buildings out of 1884 had negative energy demand values, while 1401 buildings had negative solar gains values. When the 32 buildings with negative energy demand values were analysed, it was found that all buildings were built after the year 2000 and they were either multi-family or terrace house types. With this insight, all formulas, the input data and the uom were reviewed, and the calculation was manually done for a single building to control the Python script. While the manual calculation showed the exact same results as in the Python implementation, it was found that the window ratio data coming from the TABULA building physics library included considerably high values for multi-family and terrace houses built after the year 2000, 0.51 and 0.39 respectively. Therefore, this data was determined as a potential reason of the negative values; however, we lack of data to verify this theory.

7. CONCLUSIONS AND FUTURE WORK

This article presented the development and testing steps of a CityJSON Energy Extension to be used for space heating demand calculation. The Extension was first created as a semi-direct translation from the Energy ADE KIT profile, and tested for the use case by storing all required input data in a CityJSON + Energy Extension file. The shortcomings in this process were then taken into account to improve the Extension. For the use case, a subset of Rijssen-Holten in the Netherlands was used as the study area and the Dutch standard *NTA 8800* was used to calculate the space heating demand of buildings.

Compared to the Energy ADE KIT profile, the CityJSON Energy Extension provides a simple structure without deep hierarchies in the schema, which helps to obtain an easy-to-use Extension for the users and contributes to fast data retrieval. However, the restricted Extension mechanism of CityJSON creates a limitation for the schema, since it is not possible to directly define non-CityObjects in a CityJSON Extension. A workaround was introduced for this by defining non-CityObjects under the "extraCityObjects" property of the Extension mechanism; however, it was discussed that this negatively impacts the simplicity and clarity of the Extension schema. Therefore, the Extension mechanism of CityJSON should be extended with new functionalities so that a direct support for defining non-CityObjects is provided.

The space heating demand calculation resulted in negative values for 32 buildings in the study area, all of which were built after the year 2000 and had a building typology of multi-family or terrace house. The window ratio data, obtained from the TABULA building physics library, was determined as a potential reason of this. While a smaller window ratio helped to eliminate the negative values, a validation was not performed because it was not possible to obtain window ratio data from other sources. In addition, even though the overall space heating demand values were compared with SimStadt software, it was not possible to validate the results since we lack of ground truth data.

This research is expected to increase the usability of CityJSON for a larger range of fields, since one energy application is now supported with the CityJSON Energy Extension. Moreover, since the definition and testing of complex elements, such as non-CityObjects, are discussed, the development steps in this article may be used as guidelines to develop new complex CityJSON Extensions. In addition to this, new tools may be developed for CityJSON that consider the Extension elements

for visualisation and processing, and existing tools may be improved to ensure compatibility. Finally, the findings in this article about the current structure of the Energy ADE may be used to further improve the data model for easier use and to obtain a simpler structure.

We are aware that our proposal for the units of measurement as an extra root property might be a too strong simplification as it hinders to store of multiple energy demands with different uom i.e., Gas usage is measure in Megajoules (MJ) and when a Gas appliance list its units in kilowatt-hour (kWh) it refers to the heat output and not gas consumption. Therefore, future work will test if it may make sense to change the mapping rules, in order to tackle this potential issue.

Since this first proposal of the CityJSON Energy Extension is tailored to the NTA 8800 standard, modifications took place like the inclusion of new attributes or the design decisions to manage the relationship between CityObjects (Figure 9). Following research steps should aim on the full incorporation of the Energy ADE v1.0 which would contribute to the testing with other use cases and to evaluate if the current taken decisions are correct or could lead to incompatibilities.

In addition, a software tool may be developed to provide lossless conversion between CityGML + Energy ADE and CityJSON + Energy Extension files. This way, the latter may be validated against the Energy ADE, and further comparison between the two extensions would be possible.

ACKNOWLEDGEMENTS

This research was partly funded by the European Research Council (ERC) under the European Union's Horizon2020 Research & Innovation Programme (grant agreement no. 677312 UMnD: Urban modelling in higher dimensions).

REFERENCES

Agugiaro, G., 2016. 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., 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.

Benner, J., 2018. CityGML Energy ADE V. 1.0 Specification. http://www.citygmlwiki.org/images/3/38/Energy_ADE_specification_2018_03_25.pdf.

citygml-tools Development Team, 2022. citygml-tools, Version 1.4.4. https://github.com/citygml4j/citygml-tools.

CityJSON Development Team, 2021. Mapping the Noise ADE to a CityJSON Extension. https://www.cityjson.org/tutorials/extension/.

Duminil, E., Brassel, K., Nouvel, R., Benoit, A., Bruse, M., Betz, M., Alam, N., Dastageeri, H., Wate, P., Debue, P., Köhler, S., Weiler, V., Monsalvete, P., Zirak, M., Bao, K., Schneider, S., Coors, V., 2021. SimStadt.

Gröger, G., Plümer, L., 2012. CityGML–Interoperable semantic 3D city models. *ISPRS Journal of Photogrammetry and Remote Sensing*, 71, 12–33.

Kadaster, 2022. Basisregistratie Adressen en Gebouwen (BAG). https://www.kadaster.nl/zakelijk/producten/adressen-en-gebouwen/bag-2.0-extract.

Ledoux, H., Arroyo Ohori, K., Kumar, K., Dukai, B., Labetski, A., 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.

Ledoux, H., Kumar, K., 2018. Updates on the CityJSON Energy Extension. *10th Workshop on the CityGML Energy ADE*, TUDelft, Delft, NL.

León-Sánchez, C., Giannelli, D., Agugiaro, G., Stoter, J., 2021. Testing the new 3D BAG dataset for energy demand estimation of residential buildings. *ISPRS Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*.

Meteorological Data Portal, 2022. https://www.tudelft.nl/?id=59090&L=1.

Nys, G., Kharroubi, A., Poux, F., 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.

Pasquinelli, A., Agugiaro, G., Tagliabue, L. C., Scaioni, M., 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.

Rossknecht, M., 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, 2020. NTA 8800+A1: Energy performance of buildings - Determination method. https://www.nen.nl/en/nta-8800-2020-a1-2020-nl-278296.

Scartezzini, J., Nouvel, R., Brassel, K., Bruse, M., Duminil, E., Coors, V., Eicker, U., Robinson, D., 2015. SimStadt, a new workflow-driven urban energy simulation platform for CityGML city models.

Van den Brom, P., 2020. Energy in Dwellings: A comparison between Theory and Practice. A+BE | Architecture and the Built Environment.

Vitalis, S., Arroyo Ohori, K., Stoter, J., 2019. Incorporating Topological Representation in 3D City Models. *ISPRS International Journal of Geo-Information*, 8(8).

Wu, J., 2021. Automatic building permits checks by means of 3D city models. Master's Thesis, Delft University of Technology.