INTEGRATING DYNAMIC DATA AND SENSORS WITH SEMANTIC 3D CITY MODELS IN THE CONTEXT OF SMART CITIES

K. Chaturvedi[a]*, T. H. Kolbe[a]

[a] Technische Universität München, Chair of Geoinformatics, 80333 Munich, Germany -
(kanishk.chaturvedi,thomas.kolbe)@tum.de

KEY WORDS: Smart Cities, Semantic 3D City Models, CityGML, Dynamizers, Sensor Web Enablement

ABSTRACT:

Smart cities provide effective integration of human, physical and digital systems operating in the built environment. The advancements in city and landscape models, sensor web technologies, and simulation methods play a significant role in city analyses and improving quality of life of citizens and governance of cities. Semantic 3D city models can provide substantial benefits and can become a central information backbone for smart city infrastructures. However, current generation semantic 3D city models are static in nature and do not support dynamic properties and sensor observations. In this paper, we propose a new concept called Dynamizer allowing to represent highly dynamic data and providing a method for injecting dynamic variations of city object properties into the static representation. The approach also provides direct capability to model complex patterns based on statistics and general rules and also, real-time sensor observations. The concept is implemented as an Application Domain Extension for the CityGML standard. However, it could also be applied to other GML-based application schemas including the European INSPIRE data themes and national standards for topography and cadasters like the British Ordnance Survey Mastermap or the German cadaster standard ALKIS.

1. INTRODUCTION

1.1 Smart Cities

"A smart city is a system integration of technological infrastructure that relies on advanced data processing with the goals of making city governance more efficient, citizens happier, businesses more prosperous and the environment more sustainable" (Yin et al., 2015). Smart cities allow effective integration of human, physical and digital systems operating in the built environment, and thus, improve the support of citizens and governance of cities (Degbelo et al., 2016). This concept is emerging rapidly and many implementations or projects worldwide involve building smart city infrastructures. Commercial implementations include IBM Smarter Cities (IBM, 2016), and City Next (CityNext, 2015) from Microsoft. Some of the projects are run by consortia of universities, companies and city councils in collaborative manner such as Smart Sustainable Districts (SSD, 2015), under Climate-KIC of the European Institute of Innovation & Technology (EIT), and CitySDK (CitySDK, 2015). Nearly all smart city concepts are mostly looking at the Internet of Things (IoT), sensor observations and Big Data. However, it is also important to consider geographic information. Many of the simulations or planning scenarios for the cities need to work with models of the physical reality. Hence, we see semantic 3D city models as an important complementary asset. Semantic 3D city models (i) give the spatial context to all information that is related to the physical entities in cities, and (ii) provide a means for interactive and spatio-semantic queries and aggregations. Another important aspect of information technology frameworks for smart cities is that they require to link different stakeholders, who use sensors and databases from many different branches. Therefore, it is important to achieve interoperability between systems, so that they can be linked and used together within one infrastructure. The Open Geospatial Consortium (OGC) provides open standards for the global geospatial community dealing with interoperability. OGC recognizes the importance of open standards and proposes a "Smart Cities Spatial Information Framework" (Percivall, 2015) for urban systems and spatial decision-making processes. The framework is based on the integration of OGC open standards and geospatial technology and is critical to achieve the benefits of spatial communication for smart cities. Some of the potential geospatial OGC standards are CityGML (Gröger et al., 2012), IndoorGML (Lee et al., 2014), and Building Information Models (Eastman, 1999). OGC also provides a mature and well supported framework for a family of different web services (OGC, 2015) like the Catalogue Service, Web Feature Service, Web Coverage Service, Web Mapping Service, and Web Processing Service. Inclusion of sensor technologies is also essential for smart city concepts and is well covered by the OGC Sensor Web Enablement initiative (SWE) (SWE, 2015) . SWE is a set of standards, which not only allows to model sensor descriptions and observations, but also specifies web services to exchange sensor descriptions and observations in an interoperable way. The combination of open standards (and APIs) eases the delivery of geospatial features, and sensor observations in a coherent way, and thereby support interoperable and cross-domain city services.

Various researchers and organizations recognize the importance of such open and interoperable standards in the context of smart city applications. (Moshrefzadeh and Kolbe, 2016) propose a new concept called Smart District Data Infrastructure (SDDI), which is a distributed architecture based on the Service Oriented Architecture. SDDI is an effort to develop smart city infrastructures for specific districts within European cities. OGC Smart City Interoperability Initiatives (OGC, 2016) also include testbeds and pilots for smart city infrastructures. One of the first initiatives is the OGC Future City Pilot Phase 1 (FCP1, 2016). This pilot is based in Europe and its objective is to demonstrate how the use of international standards such as CityGML and Industry Foundation Classes (IFC) together can provide stakeholders with information, knowledge and insight enhancing financial, environmental, and social outcomes for citizens living in cities.

*Corresponding author

1.2 Role of semantic 3D city models in smart cities

"Spatial information is pervasive and primary" (Percivall, 2015) in the context of smart cities and, thus, semantic 3D city models play a crucial role. Semantic 3D city models (Kolbe, 2009) provide an inventory and description of the physical and built environment. They may include real world entities such as buildings, streets, vegetation, water bodies, utilities and terrains. Such entities are represented by the ontological structure including thematic classes, attributes, and their interrelationships. They cover thematic aspects, 3D geometry, 3D topology, and appearances of the physical environment. The objects can be represented in multiple scales (e.g. from generalized box models to highly detailed models). CityGML is an international standard for semantic 3D city models issued by the OGC. This standard defines a data model with different thematic classes and allows data exchange for the same in an open and interoperable way. Such semantic 3D city model standards can become the central information backbone for smart city infrastructures.

However, current generation semantic 3D city models are static in nature and do not support time-dependent properties. There are many application and simulation scenarios (e.g. environmental simulations, disaster management, training simulators), where time plays an important role. It is also important to distinguish between different types of changes that take place in cities over time. Some of these changes may be slower in nature, e.g. (i) the history or evolution of cities such as construction or demolition of buildings, and (ii) managing multiple versions of the city models. (Chaturvedi et al., 2015) propose an approach to manage versions and history within semantic 3D city models. Some of the changes may also represent high frequent or dynamic variations of object properties, e.g. variations of (i) thematic attributes such as changes of physical quantities (energy demands, temperature, solar irradiation levels), (ii) spatial properties such as change of a feature's geometry, with respect to shape and location (moving objects), (iii) real-time sensor observations. In this case, only some of the properties of otherwise static objects need to represent such time-varying values. In this paper, we focus on highly frequent/dynamic variations of objects and their properties.

1.3 Integrating dynamic data with semantic 3D city models

We believe that an explicit support of dynamic properties within semantic 3D city models will be highly beneficial for smart city applications. In order to achieve the same objective, we propose a new concept called Dynamizers, which allows modeling and supporting such dynamic properties within semantic 3D city models. The approach allows representing dynamic data in the form of timeseries and provides a method to inject them into the static attributes of city models. An initial model has already been developed by (Chaturvedi and Kolbe, 2015) in the similar direction. However, it was restricted to support dynamic data from tabulated values. The present work provides substantial extensions to the previous model and allows representing dynamic values in different and more generic ways. The approach also allows integrating real time sensor observations with city objects in standardized ways. As a first step, the concept is implemented as an Application Domain Extension (ADE) for the CityGML standard. The ADE is used within the OGC Future Cities Pilot and, later, is intended to become an integral part of the next version of CityGML (version 3.0).

2. USE CASES

The support of real-time dynamic information within city models may provide substantial benefits to smart city projects. (Moshrefzadeh and Kolbe, 2016) highlight the importance of dynamic data within the proposed Smart District Data Infrastructure (SDDI). Its first implementation is based in the district Queen Elizabeth Olympic Park (QEOP), London. The SDDI introduces the concept of a virtual district model, which models the physical district's objects and can be enriched with semantic information. The virtual district model comprises of three components: (i) Topography Information Model, which provides description and inventory of the physical environment such as buildings, streets, vegetation etc. based on CityGML, (ii) Network Models, which defines functional behaviours, and resources and their flows, e.g., transportation, energy, water, or communication networks, and (iii) Visualization Model, which includes 3D computer graphic models (offline or browser based) for rendering the district objects or 2D maps (paper maps or web maps). Such district models are extensible for different applications and can be linked with other data/databases on the modelled objects. One of the main focuses is to provide links to real-time dynamic data such as consumption values measured by smart meters installed in important buildings or traffic/pedestrians flows in the streets. Figure 1 is an illustration of the virtual district model of QEOP based on CityGML LOD2. The figure provides a conceptual visualization of integrating real-time sensor observations with city objects.

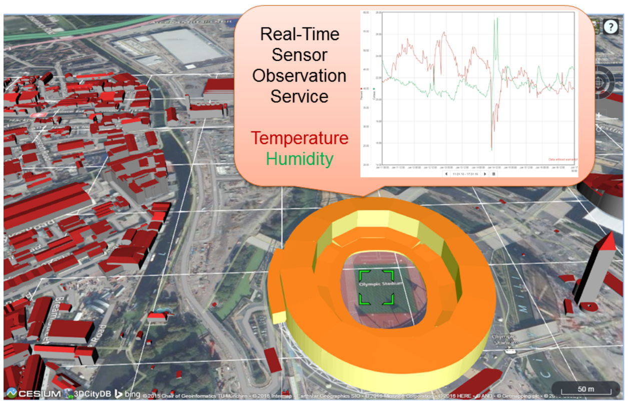

Figure 1. Sensor observations for temperature and humidity variations in a room of a building.

The Master thesis (Zahn, 2015) mentions benefits of semantic 3D city models in the applications of solar irradiation analysis. The work is based on the CityGML standard, where the analysis is performed by calculating the diffuse and direct solar irradiations. The results for different months are stored as generic attributes in CityGML. However, the current version of CityGML requires defining multiple generic attributes for different months. The usability of such analyses will drastically be improved, if city model standards allow time-dependent variations of such result values to a common generic attribute.

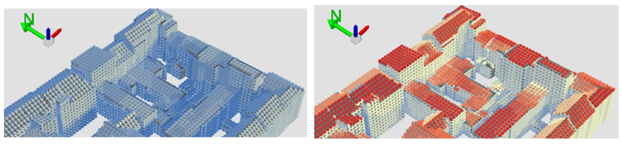

Figure 2. Visualization of global solar irradiation results for the months *(left)* January, and *(right)* July. Blue and red colours represent low and high solar radiation values respectively.

Other use case scenarios for modeling dynamic attributes within city models include (Kaden and Kolbe, 2013), (Morel and Gesquire, 2014), and (Prinaud et al., 2015).

3. RELEVANT STANDARDS

Before going into the details of Dynamizers, it is worth mentioning existing standards leveraged by Dynamizers.

3.1 ISO 19123 - Schema for coverages

(ISO19123, 2005) defines coverage as a function from spatial, temporal or spatio-temporal domain to an attribute range. A coverage associates a position within its domain to a record of values of defined data types. This ISO standard defines discrete and continuous coverages, containing three components: (i) a domain set, (ii) a range set, and (iii) a coverage function. The domain is a set of geometric and temporal objects described in terms of direct positions. The direct positions are associated with a spatial or temporal coordinate reference system. The range of a coverage is a set of feature attribute values (which may also be time-varying attribute values of geo-objects). The domain values are mapped to the range values using a coverage function.

3.2 TimeseriesML 1.0

TimeseriesML 1.0 (TimeseriesML, 2015) is a rather new OGC standard for the representation and exchange of observation data as timeseries. It is an extension of the work initially undertaken within (WaterML, 2014). However, the aim is to provide a domain-neutral model for the representation and exchange of timeseries data. TimeseriesML is a specialization of ISO 19123 Discrete Coverages, where domain is a set of ordered time instances where each is associated with a single value from the attribute space. The TimeseriesML schema supports two types of encodings: (i) an interleaved time-value pair encoding, whereby the time and value are coupled together and the coupling explicitly represents the mapping and (ii) a domain-range encoding, where the domain and range are encoded separately, with a mapping function that allows looking up the range value for a given domain value. Sample XML encodings of this standard are illustrated below in section 5.1.

3.3 Observations and Measurements

Observations and Measurements (O&M) (O&M, 2013) is a generic information model for describing observations. The observation is modelled as a Feature within the context of the General Feature Model (ISO19101, 2014). An observation feature binds a result to a feature of interest, upon which the observation was made. The observed property is a property of the feature of interest. An observation uses a procedure to determine the value of the result, which may involve a sensor or an observer, analytical procedure, simulation or other numerical process. O&M is one of the core standards in the OGC Sensor Web Enablement suite, providing the response model for Sensor Observation Service (SOS) (SOS, 2012) and SensorThings API (Liang, 2016).

3.4 Sensor Web Enablement

The OGC SWE standards suite specifies interfaces and data encodings to enable real-time integration of heterogeneous sensor networks. In this way, most types of sensors can be discovered, accessed and reused for creating web-accessible sensor applications and services. SWE contains two important information models: (i) Sensor Model Language (SensorML), which defines an XML schema for describing the processes within sensor and observation processing systems, and provides information needed for discovery, georeferencing, and processing of observations, and (ii) Observations & Measurements (O&M), as described in section 3.3. In addition, SWE provides different interface models and web services. The most important service within the scope of this paper is the Sensor Observation Service (SOS). It defines an open interface by which a client can obtain observation data and sensor and platform descriptions from one or more sensors.

Furthermore, the OGC SensorThings API is a recent OGC standard, providing an open and unified way to interconnect resource-constrained Internet of Things (IoT) devices, data, and applications over the Web. It is a very lightweight standard built on Web protocols and OGC SWE standards, and applies an easy-to-use REST-ful interface and JSON-based data encodings.

4. DYNAMIZERS

4.1 Introduction

Dynamizer is a new concept, which allows modeling and integrating dynamic properties within semantic 3D city models. It serves three major purposes:

1. Dynamizer is a data structure to represent dynamic values in different and generic ways. Such dynamic values may be given by tabulation of time/value pairs; patterns of time/-value pairs; by referencing an external file; or by retrieving observations from sensors.
2. Dynamizer delivers a method to enhance static city models by dynamic property values. It references a specific attribute (e.g. geometry, thematic data or appearance) of an object within a 3D city model providing dynamic values overriding the static value of the referenced object attribute.
3. Dynamizer objects establish explicit links between sensor/observation data and the respective properties of city model objects that are measured by them. By making such explicit links with city object properties, the semantics of sensor data become implicitly defined by the city model.

In this way, Dynamizers can be used to inject dynamic variations of city object properties into an otherwise static representation. The advantage in using such approach is that it allows only selected properties of city models to be made dynamic. If an application does not support dynamic data, it simply does not allow/include these special types of features.

4.2 Modeling dynamic values within Dynamizers

As shown in figure 3, Dynamizers can represent dynamic values in generic ways. The source of dynamic data may vary for different applications. The values may be obtained from (i) external files (e.g., CSV files) or data from external files included inline, (ii) external databases (e.g. tabulated values of simulation specific data), or (iii) sensors (e.g. real time observations from SOS or SensorThings API encoded in O&M format). The Dynamizers provide an explicit way to model such dynamic variations. They utilize different encodings defined according to TimeseriesML 1.0. Utilizing TimeseriesML, the timeseries can be represented as interleaved time/value pairs or by a domain range encoding. Since TimeseriesML is an implementation of the ISO 19123 Discrete Coverages, it allows defining different data types for specific time positions. Such data types may be numeric values, categories, spatial coordinates, or links to external files. Dynamizers also utilize TimeseriesML in defining interpolation and aggregation types for each point in the timeseries. It helps in mapping missing values or multiple values to specific time points. Apart from TimeseriesML, Dynamizers also allow for the inline representation of O&M data within them. As explained in section 3.3, O&M is one of the core standards for the response models of OGC SWE based standards such as SOS and SensorThings API. In this way, Dynamizers model real-time sensor observations.

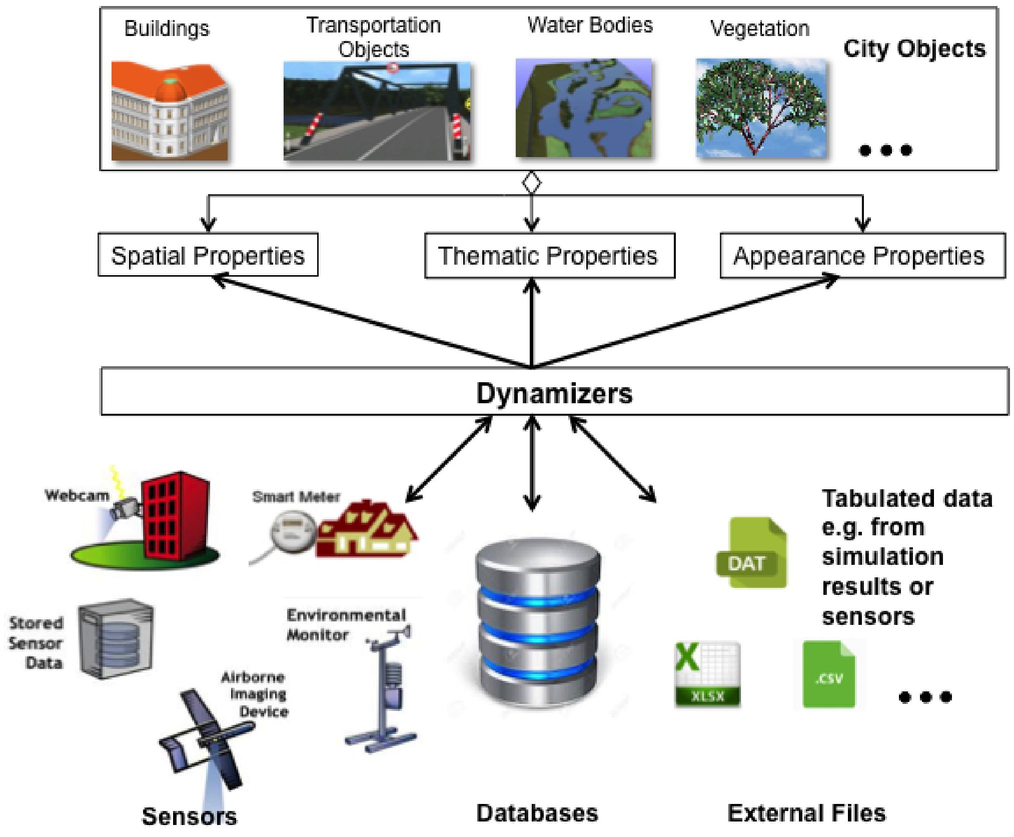

Figure 3. Conceptual representation of Dynamizers, allowing (i) the representation of time-variant values from sensors, simulation specific databases, and external files, (ii) enhancing the properties of city objects by overriding their static values

4.3 Repetitive Patterns (Atomic and Composite Timeseries)

Dynamizers support absolute start and end points referencing a specific time reference system. The absolute time points can be mapped to attribute values and can be represented as tabulation of measured data. One common example illustrating such scenario is mapping of energy consumption values of a building for every hour in a day. However, in many applications, it is not sufficient just to provide a means for the tabulation of time-value pairs. They may require patterns to represent dynamic variations of properties based on statistics and general rules. In such scenarios, time cannot be described by absolute positions, but relative to absolute positions. In these cases, time may be defined for a non-specific year (e.g. averages over many years), but still classified by relative time of an year. For example, January monthly summaries for the energy consumption of a building might be described as ”all-Januaries 2001-2010”. Similarly, the energy consumption values may reflect generic patterns for individual weekdays/weekends in a week or a month. Another example scenario may also be determining patterns for specific seasons (such as spring, summer, autumn and winter) over ten years.

In order to support such patterns, Dynamizers introduce the concept of atomic and composite timeseries. Atomic timeseries represent dynamic values having time points that can be based on local/relative time reference systems. This allows mapping customized time values such as weekdays, weekends, seasons etc. Furthermore, composite timeseries consist of nested timeseries with arbitrary depths. Such timeseries can be repeated several times to obtain specific patterns. For instance, an atomic timeseries can be defined for a working day, a Saturday, and a Sunday (represented by A, B, and C respectively). Now, a composite timeseries may contain five repetitions of atomic timeseries A to reflect a pattern of energy consumption values for all weekdays. However, energy demand estimations may require separate patterns for weekdays and weekends requiring composition of multiple variation behaviours. Such complex behaviours may be expressed using composite timeseries, wherein a weekly pattern can be defined containing the energy values for all seven days of a week (represented as AAAAABC). The advantage in using such composite timeseries is that we can express time-varying data for an arbitrary long time period by combining various patterns. For example, if we need to determine the pattern for energy consumption values for evening hours on weekdays, an atomic timeseries can be defined for 6 hours (e.g. from 6 PM to midnight). Further, the composite timeseries allows connecting additional 6 hours for all weekdays in a week. An illustration of the representation of such complex patterns by atomic and composite timeseries is given in (Chaturvedi and Kolbe, 2015).

4.4 Sensors and observations

Apart from different timeseries representations, Dynamizers provide support for sensors and sensor observations within them. Within the OGC SWE standards suite, sensor descriptions are encoded in SensorML format and sensor observations in O&M format. The web services such as Sensor Observation Service and SensorThings API allow the retrieval of sensor descriptions and observations using different requests. Dynamizers leverage such standards and allow integrating sensors with semantic 3D city models in two ways:

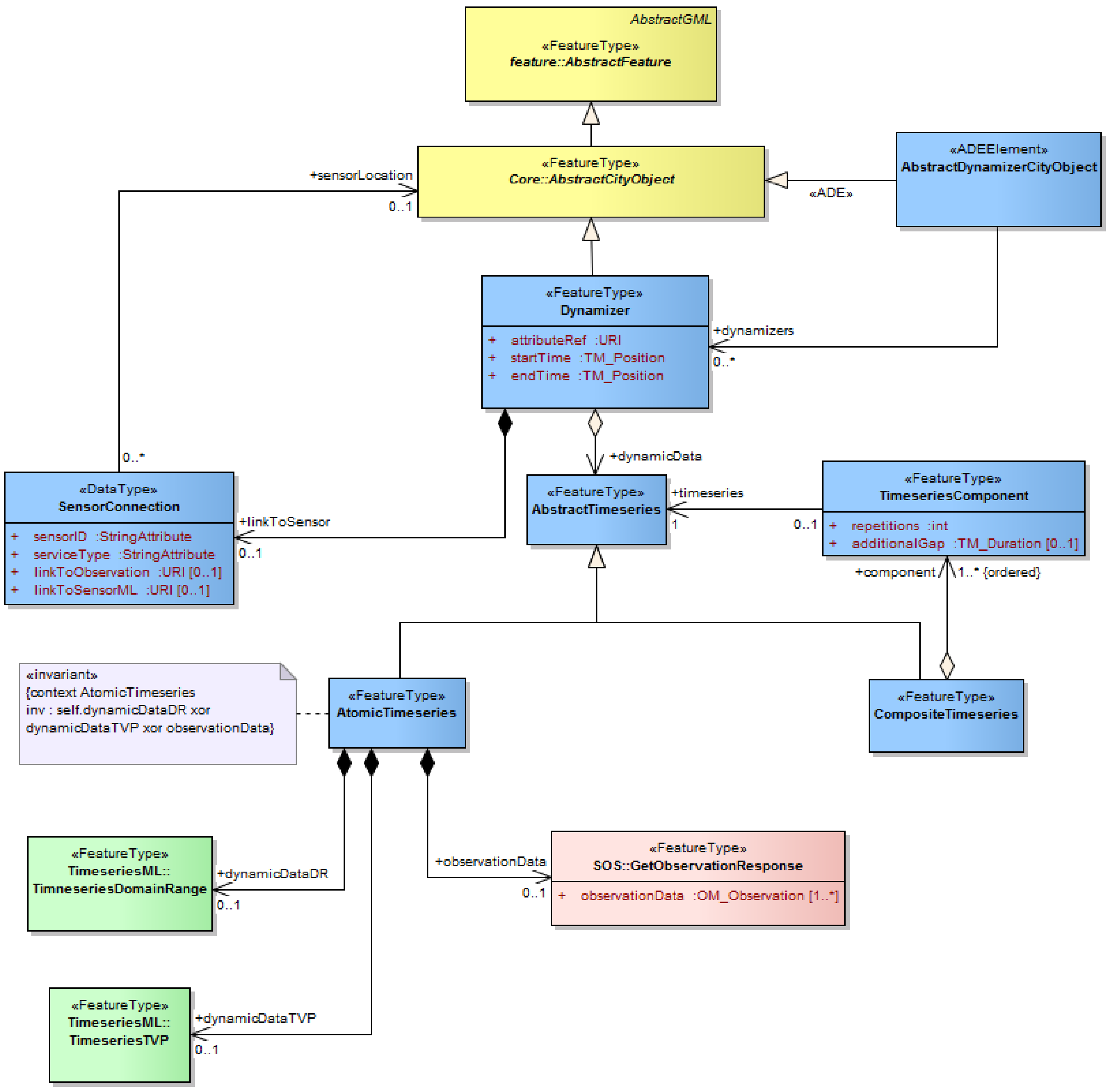
AbstractGML
«FeatureType» feature::AbstractFeature
«FeatureType» Core::AbstractCityObject
«ADEElement» AbstractDynamizerCityObject
«ADE»
+sensorLocation
0..1
«FeatureType» Dynamizer
+ attributeRef :URI
+ startTime :TM_Position
+ endTime :TM_Position
+dynamizers
0..*
+dynamicData
«DataType» SensorConnection
+ sensorID :StringAttribute
+ serviceType :StringAttribute
+ linkToObservation :URI [0..1]
+ linkToSensorML :URI [0..1]
+linkToSensor
0..1
0..*
«FeatureType» AbstractTimeseries
+timeseries
1
0..1
«FeatureType» TimeseriesComponent
+ repetitions :int
+ additionalGap :TM_Duration [0..1]
+component
1..* {ordered}
«invariant» {context AtomicTimeseries inv : self.dynamicDataDR xor dynamicDataTVP xor observationData}
«FeatureType» AtomicTimeseries
«FeatureType» CompositeTimeseries
«FeatureType» TimeseriesML:: TimneseriesDomainRange
+dynamicDataDR
0..1
+observationData
0..1
«FeatureType» SOS::GetObservationResponse
+ observationData :OM_Observation [1..*]
«FeatureType» TimeseriesML:: TimeseriesTVP
+dynamicDataTVP
0..1

Figure 4. Dynamizer defined as UML model. Newly introduced classes are shown in blue. Classes shown in green and pink represent TimeseriesML 1.0 and Sensor Observation Service. Classes shown in yellow are from GML and CityGML respectively.

i **By linking Dynamizers with sensors**: Dynamizers support direct links to sensors and observations utilizing different requests, e.g. the DescribeSensor and GetObservation operations in case of OGC SOS. With such requests, sensor descriptions and observations are retrieved and linked with city objects. A sensor connection is defined by a unique sensor ID, the type of sensor service (e.g. OGC SOS and Sensor Things API), and URL links for the two different requests named above. Applications consuming CityGML plus Dynamizer data can utilize these links to retrieve the sensor data for the referenced web services.

ii **By including sensor observations within Dynamizers**: Dynamizers allow representing sensor observations within them. Sensor observations are mainly encoded in O&M format. Dynamizers support inlining of O&M data representing sensor observation values, which then are injected into the attributes of specific city objects. This approach is useful to exchange sensor readings for a past time period together and consistently within the city model.

5. IMPLEMENTATION

Dynamizers are implemented as an Application Domain Extension (ADE) for the CityGML standard. The ADE mechanism allows for the systematic extension of each CityGML object type by additional attributes as well as the introduction of new object types. This implementation allows Dynamizers to be used with the current version of CityGML (version 2.0).

As shown in Figure 4, *Dynamizer* is a new feature type which is a sub-class of *AbstractCityObject*. In addition, *AbstractDynamizerCityObject* extends the class *AbstractCityObject* by the additional association *dynamizers*. That means that all city objects such as buildings, roads, vegetation etc. can now include links to their respective dynamizer features. The class *Dynamizer* consists of three attributes: (i) *attributeRef*, (ii) *startTime*, and (iii) *endTime*. *attributeRef* refers to a specific attribute of a city object by XPath (XPath, 2011). XPath is a W3C recommendation used to navigate through elements and attributes within an XML document. *startTime* and *endTime* are absolute time points denoting the time span for which the Dynamizer provides dynamic values. The time points are modeled as TM_Position defined by ISO 19108 (ISO19108, 2002), and are referenced to a specific time reference system (e.g. Gregorian Calendar). In addition, Dynamizers contain dynamic data in the form of a timeseries. The dynamic data is modeled as *AbstractTimeseries*, which is responsible for representing time-variant or dynamic values in different and generic ways. The timeseries may be modeled in two ways: (i) *AtomicTimeseries*, and (ii) *CompositeTimeseries*. *AtomicTimeseries* consists of either *dynamicDataDR*, *dynamicDataTVP*, or *observationData*. It allows different representations of timeseries such as TimeseriesML 1.0 encodings (interleaved time/value pair and domain-range), values from external files and databases, and observations encoded in O&M from sensors. Please note that these representations are mutually exclusive for each *AtomicTimeseries* object. This condition is formally expressed using the Object Constraint Language (OCL) in the UML diagram. OCL (OCL, 2014) is a declarative language for describing rules that apply to Unified Modeling Language (UML) models.

CompositeTimeseries is modeled in such a way that it composes of an ordered list of *AbstractTimeseries*. *CompositeTimeseries* includes a component called *TimeseriesComponent*, which denotes the number of repetitions for a timeseries component. It also contains an attribute *additionalGap*, which allows two non-overlapping timeseries, that have been separately collected, to connect in order to make a single timeseries. For example, if the latest two months of timeseries data is transferred from one system to a major archive, the series must be connected in order to make a full series over which the patterns can be determined (e.g. to determine yearly patterns). However, for a composite timeseries, it is necessary to model time positions according to relative time reference system. According to ISO 19108, they may be defined as TM_OrdinalEras within TM_OrdinalReferenceSystems. It allows defining local reference systems for specific use cases.

Additionally, Dynamizers provide links to external sensors with the help of the datatype *SensorConnection*. It contains four attributes (i) *linkToObservation* - link to observation response, (ii) *linkToSensor* - link to Sensor description encoded in SensorML format, (iii) *sensorId* - a unique identification of the sensor, and (iv) *serviceType* - the type of service such as SOS and SensorThings API. OGC SOS involves different requests for retrieving sensor descriptions and observations. DescribeSensor is used to retrieve sensor description in SensorML format. GetObservation is used to retrieve sensor observations encoded in O&M format. The request parameter also allows to include the specification of spatial and temporal filters.

5.1 Illustration of concept

This section shows an example for representing and supporting dynamic data within CityGML documents. Figure 5 shows the hourly reading of the energy consumption retrieved from a smart meter installed in a residential building. The usage (in kwh) is given against the time periods. The values of smart meters are usually stored and updated in a file or a database table. By using OGC SWE standards, it is also possible to create an interoperability layer, so that real-time consumption values can be retrieved in a standardized way, e.g. by utilizing the SOS or SensorThings API. These values may be stored as generic attributes within CityGML files. However, such generic attributes are static in the current version of CityGML, which means, all readings of smart meters would be required to be stored as separate attributes. Considering the highly dynamic nature of such values, it is cumbersome to store them in CityGML and other standards

Start Time	End Time	Consumption (in kwh)
2016-05-12T00:00:00Z	2016-05-12T01:00:00Z	0.385
2016-05-12T01:00:00Z	2016-05-12T02:00:00Z	0.365
2016-05-12T02:00:00Z	2016-05-12T03:00:00Z	0.425
⋮	⋮	⋮
2016-05-12T23:00:00Z	2016-05-13T00:00:00Z	0.380

Figure 5. Hourly energy consumption values from a smart meter.

of semantic 3D city models. The new Dynamizer feature allows us to map such highly dynamic values in different ways. At the same time, it refers to a specific attribute of a CityGML feature (here: a generic attribute) and overrides its value as per variations in time.

Below is an example illustrating how such dynamic values of a smart meter can be supported within CityGML.

```
<cityObjectMember>
 <!--CityGML Building feature-->
 <bldg:Building gml:id = "buildingA">
  <gen:doubleAttribute name = "ElecConsumption">
   <gen:value>0.361</gen:value>
  </gen:doubleAttribute>
 </bldg:Building>
</cityObjectMember>
<cityObjectMember>
  <!--New Dynamizer feature-->
  <dyn:Dynamizer gml:id = "ElecConsumptionTimeseries" >
    <dyn:attributeRef>//bldg:Building[@gml:id =’
         buildingA’]/doubleAttribute[@name = ’
         ElecConsumption’]/gen:value</dyn:attributeRef>
    <dyn:startTime>2016-05-12T00:00:00Z</startPoint>
    <dyn:endTime>2016-05-13T00:00:00Z</endPoint>
    <dyn:dynamicData>
     <dyn:AtomicTimeseries>
      <dyn:dynamicDataTVP>
       <tsml:TimeseriesTVP gml:id="tsml.
            measurementimeseries.elecConsumption">
        <!--Time/Value Pair 1-->
        <tsml:point>
         <tsml:MeasurementTVP>
          <tsml:time>2016-05-12T00:00:00Z</tsml:time>
          <tsml:value>0.385</tsml:value>
         </tsml:MeasurementTVP>
        </tsml:point>
        ........
        <!--Multiple values can be represented-->
       </tsml:TimeseriesTVP>
      </dyn:dynamicDataTVP>
     </dyn:AtomicTimeseries>
    </dyn:dynamicData>
  </dyn:Dynamizer>
</cityObjectMember>
```

A Building feature is represented as *buildingA* and it contains a generic attribute named ElecConsumption, which stores hourly values of electricity consumption from a smart meter. Dynamizer is a new feature type, which contains a reference to the generic attribute *ElecConsumption* in XPath expression defined as:

//bldg:Building [@gml:id = ’buildingA’]/doubleAttribute[@name = ’ElecConsumption’]/gen:value

Such expressions are more specific than just pointing to a feature of interest as in the O&M model. *startTime* and *endTime* are absolute time positions for the dynamic values represented in the Dynamizer. *dynamicDataTVP* contains dynamic values in time/value pair encoding as defined by TimeseriesML. The values are interleaved with the absolute time positions. However, it also allows to provide regular spacing where it is only required to provide a start time and a duration of regular spacing. The next time positions are automatically determined with the help of spacing. It is also possible to define such dynamic values in TimeseriesML domain-range encoding, tabular form retrieved from an external file or a database, and observations from sensors encoded in O&M format.

A Dynamizer allows injecting dynamic values into a static attribute of a specific CityGML object. The advantage with this approach is that it does not affect other (static) attributes of the CityGML object. If an application does not support dynamic data, it simply does not consider the Dynamizer features. The given example is an illustration of a variation of a thematic attribute value of the CityGML feature. However, it is also possible to support dynamic variations in geometry and appearance of city objects.

There may also be scenarios when dynamic data for the same feature is obtained from different sources for different time intervals. For example, energy consumption values are retrieved in a tabular data format from an external file for a time period A. For another time period B, such values are obtained from a statistical pattern, and for a third time period C, SOS observations from a smart meter are available. As shown below, our approach allows defining multiple Dynamizer features for the same CityGML attribute for non-overlapping time periods.

Time Period A: Inline data included from an external file

```
<dyn:Dynamizer gml:id = "DynamizerA" >
 <dyn:attributeRef>//bldg:Building[@gml:id ='buildingA
      ']/doubleAttribute[@name = 'ElecConsumption']/gen
      :value</dyn:attributeRef>
 <dyn:startTime>2015-01-01T00:00:00Z</startPoint>
 <dyn:endTime>2015-05-01T00:00:00Z</endPoint>
 <dyn:dynamicData>
  <dyn:AtomicTimeseries>
   <dyn:dynamicDataTVP>
    <tsml:TimeseriesTVP>
     <tsml:point>
      <tsml:MeasurementTVP>
        <tsml:time>2015-01-01T00:00:00Z</tsml:time>
        <tsml:value>39.97</tsml:value>
      </tsml:MeasurementTVP>
     </tsml:point>
     <!--More Time/value pair encodings-->
    </tsml:TimeseriesTVP>
   <dyn:dynamicDataTVP>
  </dyn:AtomicTimeseries>
 </dyn:dynamicData>
</dyn:Dynamizer>
```

Time Period B: Statistical patterns for weekdays and weekends

```
<dyn:Dynamizer gml:id = "DynamizerB" >
 <dyn:attributeRef>//bldg:Building[@gml:id ='buildingA
      ']/doubleAttribute[@name = 'ElecConsumption']/gen
      :value</dyn:attributeRef>
 <dyn:startTime>2015-05-01T00:00:00Z</startPoint>
 <dyn:endTime>2015-09-01T00:00:00Z</endPoint>
 <dyn:dynamicdata>
  <!--Composite timeseries for weekdays patterns-->
  <dyn:CompositeTimeseries>
   <dyn:component>
    <dyn:TimeseriesComponent gml:id="Weekdays">
     <dyn:repetitions>5</dyn:repetitions>
     <dyn:timeseries>
      <dyn:AtomicTimeseries>
       <tsml:TimeseriesTVP>
        <tsml:metadata>
         <tsml:TimeseriesMetadata>
          <!--Local time reference system-->
          <tsml:baseTime frame = "#localReference">
               00:00:00</tsml:baseTime>
          <!--Regular spacing of 1 hour-->
          <tsml:spacing>PT1H</tsml:spacing>
         </tsml:TimeseriesMetadata>
        </tsml:metadata>
        <tsml:point>
         <tsml:MeasurementTVP>
          <tsml:value>39.97</tsml:value>
         </tsml:MeasurementTVP>
        </tsml:point>
        <!--Further values of MeasurementTVP-->
       </tsml:TimeseriesTVP>
      </dyn:AtomicTimeseries>
     </dyn:timeseries>
    </dyn:TimeseriesComponent>
   </dyn:component>
  <dyn:CompositeTimeseries>
 <dyn:dynamicdata>
</dyn:Dynamizer>
```

Time Period C: Sensor observations retrieved from OGC Sensor Observation Service

```
<dyn:Dynamizer gml:id = "DynamizerC" >
 <dyn:attributeRef>//bldg:Building[@gml:id ='buildingA
      ']/doubleAttribute[@name = 'ElecConsumption']/gen
      :value</dyn:attributeRef>
 <dyn:startTime>2015-09-01T00:00:00Z</startPoint>
 <dyn:endTime>2016-01-01T00:00:00Z</endPoint>
 <dyn:dynamicData>
  <dyn:AtomicTimeseries>
   <dyn:observationData>
   <!--Reference to response from OGC SOS-->
    <sos:GetObservationResponse>
     <sos:observationData>
      <om:OM_Observation gml:id="o_1">
       <om:type xlink:href="OM_Measurement"/>
       <om:phenomenonTime>
        <gml:TimeInstant gml:id="phenomenonTime_1">
         <gml:timePosition>2015-09-01T01:00:00.000Z</
              gml:timePosition>
        </gml:TimeInstant>
       </om:phenomenonTime>
       <om:resultTime xlink:href="#phenomenonTime_1"/>
       <om:procedure xlink:href="Smart_Meter"/>
       <om:observedProperty xlink:href="ElecConsumption
            "/>
       <om:featureOfInterest xlink:href="#buildingA"/>
       <om:result>27.7</om:result>
      </om:OM_Observation>
     </sos:observationData>
     <!--More observations-->
    </sos:GetObservationResponse>
   <dyn:observationData>
  </dyn:AtomicTimeseries>
 </dyn:dynamicData>
</dyn:Dynamizer>
```

6. CONCLUSIONS AND FUTURE WORK

In this paper, we propose a new concept called Dynamizers extending static 3D city models by supporting variations of individual feature properties and associations over time. It provides a data structure to represent dynamic values in different and generic ways. Such dynamic values may be given by tabulation of time/value pairs; patterns of time/value pairs; by referencing an external file; or by retrieving observations from sensor services. In principle, Dynamizers inject dynamic variations of city object properties into the static representation. These variations are supported for thematic, geometry, and appearance properties of city objects.

The concept has been implemented as an Application Domain Extension (ADE) for the CityGML standard. This implementation allows to use such Dynamizer features with the current version CityGML 2.0. The advantage with this approach is that it allows for selected properties of city models to become dynamic without changing the original CityGML data model. If an application does not support dynamic data, it simply does not allow/include these special types of features. Dynamizer is a feature type, which even allows to be used with the OGC Web Feature Service. No extension of other OGC standards is required.

Dynamizers are a crucial concept for smart city applications and thus, are being tested in the OGC Future Cities Pilot Phase 1. Within the scope of the pilot, datasets and instance documents will be developed for conducting and demonstrating various pilot scenarios. Regarding sensors, the current Dynamizer model is limited to the OGC Sensor Observation Service. However, in the future, it will be interesting to extend support to other sensor APIs like the OGC Sensor Things API. It will also be important to investigate how Dynamizers can be interpreted by visualization clients. Dynamizers are intended to become part of the next version of CityGML (version 3.0).

The Dynamizer concept is general in the sense that it could also be applied to other GML-based application schemas including the European INSPIRE data themes and national standards for topography and cadasters like the British Ordnance Survey Mastermap or the German cadaster standard ALKIS.

ACKNOWLEDGEMENTS

Part of this work has been carried out within the project Smart, Sustainable Districts funded by the Climate-KIC of the European Institute of Innovation and Technology (EIT). We thank Climate-KIC and the EIT for supporting this work. We also acknowledge OGC Future Cities Pilot 1 for supporting this work.

REFERENCES

Chaturvedi, K. and Kolbe, T. H., 2015. Dynamizers - Modeling and Implementing Dynamic Properties for Semantic 3d City Models. In: *3rd Eurographics Workshop on Urban Data Modelling and Visualisation*, TU Delft, The Netherlands.

Chaturvedi, K., Smyth, C. S., Gesquiere, G., Kutzner, T. and Kolbe, T. H., 2015. Managing versions and history within semantic 3d city models for the next generation of citygml. In: *Selected papers from the 3D GeoInfo 2015 Conference*, Springer.

CityNext, 2015. Microsoft CityNext. https://partner.microsoft.com/en-US/Solutions/CityNext.

CitySDK, 2015. CitySDK City Service Development Kit. http://www.citysdk.eu/.

Degbelo, A., Granell, C., Trilles, S., Bhattacharya, D., Casteleyn, S. and Kray, C., 2016. Opening up Smart Cities: Citizen-Centric Challenges and Opportunities from GIScience. *ISPRS International Journal of Geo-Information* 5(2), pp. 16.

Eastman, C. M., 1999. *Building product models: computer environments, supporting design and construction.* CRC press.

FCP1, 2016. OGC Call for Proposals for a Future City Pilot Phase 1 project | OGC. http://www.opengeospatial.org/pressroom/pressreleases/2371.

Gröger, G., Kolbe, T. H., Nagel, C. and Häfele, K.-H., 2012. OGC City Geography Markup Language (CityGML) Encoding Standard | Version 2.0.0 | OGC Document No. 12-019.

IBM, 2016. IBM Smarter Cities - Future cities - United States. http://www.ibm.com/smarterplanet/us/en/smarter_cities/overview/.

ISO19101, 2014. ISO 19101-1:2014 - Geographic information – Reference model – Part 1: Fundamentals.

ISO19108, 2002. ISO 19108:2002 - Geographic information – Temporal schema.

ISO19123, 2005. ISO 19123:2005 - Geographic information – Schema for coverage geometry and functions.

Kaden, R. and Kolbe, T. H., 2013. City-wide total energy demand estimation of buildings using semantic 3d city models and statistical data. In: *Proc. of the 8th International 3D GeoInfo Conference*, Vol. 2, p. W1.

Kolbe, T. H., 2009. *Representing and exchanging 3D city models with CityGML.* 3D Geo-Information Sciences, Springer, pp. 15–31.

Lee, J., Li, K.-J., Zlatanova, S., Kolbe, T. H., Nagel, C. and Becker, T., 2014. Indoor GML | OGC 14-005r3.

Liang, S., 2016. SensorThings SWG | OGC. http://www.opengeospatial.org/projects/groups/sensorthings.

Morel, M. and Gesquire, G., 2014. Managing Temporal Change of Cities with CityGML. In: *Eurographics Workshop on Urban Data Modelling and Visualisation*, Strasbourg, France.

Moshrefzadeh, M. and Kolbe, T. H., 2016. Smart District Data Infrastructure for Smart and Sustainable Cities. In: *3th International Conference on Design & Decision Support Systems in Architecture and Urban Planning*.

OCL, 2014. Object Constraint Language. http://www.omg.org/spec/OCL/.

OGC, 2015. OGC Standards and Supporting Documents | OGC. http://www.opengeospatial.org/standards.

OGC, 2016. OGC Interoperability Program | OGC. http://www.opengeospatial.org/ogc/programs/ip.

O&M, 2013. Observations and Measurements | OGC 10-004r3.

Percivall, G., 2015. OGC Smart Cities Spatial Information Framework | OGC 14-115. http://www.opengeospatial.org/pressroom/pressreleases/2181.

Prinaud, C., Gay, G. and Gesquire, G., 2015. Exploration of the changing structure of cities: Challenges for temporal city models. In: *2015 Digital Heritage*, Vol. 2, pp. 73–76.

SOS, 2012. Sensor Observation Service | OGC 12-006.

SSD, 2015. Smart Sustainable Districts. http://www.climate-kic.org/wp-content/uploads/2013/04/Smart-Sustainable-Districts_Climate-KIC_external.pdf.

SWE, 2015. Sensor Web Enablement (SWE) | OGC 06-021r4.

TimeseriesML, 2015. Timeseries Profile of Observations and Measurements | OGC 15-043r3.

WaterML, 2014. OGC WaterML | OGC 10-126r4.

XPath, 2011. XML Path Language (XPath) 2.0 (Second Edition). http://www.w3.org/TR/xpath20/.

Yin, C., Xiong, Z., Chen, H., Wang, J., Cooper, D. and David, B., 2015. A literature survey on smart cities. *Science China Information Sciences* 58(10), pp. 1–18.

Zahn, W., 2015. Sonneneinstrahlungsanalyse auf und Informationsanreicherung von großen 3d-Stadtmodellen im CityGML-Schema, Masters Thesis, Technische Universität München. https://mediatum.ub.tum.de/download/1276236/1276236.pdf.