Future City Pilot 1 Engineering Report

Table of Contents

Publication Date: 2017-10-20

Approval Date: 2017-08-17

Posted Date: 2017-06-27

Reference number of this document: OGC 16-098

Reference URL for this document: http://www.opengis.net/doc/PER/FCP1-ER

Category: Public Engineering Report

Editors: Kanishk Chaturvedi, Thomas H. Kolbe

Title: FCP1 Engineering Report

---

indicate compliance with any LICENSOR standards or specifications.

This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

Abstract

The Future City Pilot Phase 1 (FCP1) is an OGC Interoperability Program initiative in collaboration with buildingSMART International (bSI). The pilot aimed at demonstrating and enhancing the ability of spatial data infrastructures to support quality of life, civic initiatives, and urban resilience. During the pilot, multiple scenarios were set up based on real-world requirements and were put forward by the pilot sponsors: Sant Cugat del Vallès (Barcelona, Spain), Ordnance Survey Great Britain (UK), virtualcitySYSTEMS GmbH (Germany), and Institut National de l'Information Géographique et Forestière - IGN (France). The scenarios were focused on (i) the interoperability between the two international standards: Industry Foundation Classes (IFC) and CityGML; (ii) city flood modeling; and (iii) supporting real-time sensor readings and other time-dependent properties within semantic 3D city models. The solutions for the respective scenarios were developed by the pilot participants: University of Melbourne (Australia), Remote Sensing Solutions, Inc. (U.S.A), and Technical University of Munich (Germany). This Engineering Report (ER) focuses on the third scenario requiring the support of real-time sensors and other time-dependent properties within semantic 3D city models based on the CityGML standard. It highlights a new concept 'Dynamizer', which allows representation of highly dynamic data in different and generic ways and providing a method for injecting dynamic variations of city object properties into the static representations. It also establishes explicit links between sensor/observation data and the respective properties of city model objects that are measured by them. The Dynamizer concept has been implemented as an Application Domain Extension (ADE) of the CityGML standard. This implementation allows to use new dynamizer features with the current version of the CityGML standard (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. The details and results of the pilot are mentioned in the following YouTube video: https://youtu.be/aSQFIPwf2oM

Business Value

The current generation semantic 3D city models are static in nature and do not support time-dependent properties. However, in reality, there are different types of changes that take place in cities over time. Some of these changes are slower in nature, e.g., (i) history or evolution of cities such as construction or demolition of buildings and (ii) managing multiple versions of the city models. Some of the changes may also represent high frequency 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); and (iii) real-time sensor observations. In this case, only some of the properties of otherwise static objects need to represent such time-varying values. Dynamizer is a new concept supporting the latter types of changes and allowing for enriching the city model by data from dynamic data feeds.

What does this ER mean for the Working Group and OGC in general

This ER summarizes the work performed during the FCP1 and provides an outlook on possible future activities. It serves as a starting point for the OGC community in general, and the CityGML Working Group in particular, to understand some of the latest discussions on supporting real-time sensors and other time-dependent properties within the CityGML standard. It highlights a number of references to more detailed materials to facilitate more in-depth research and analysis.

How does this ER relates to the work of the Working Group

The Dynamizer concept is already being discussed within the CityGML Standard Working Group (SWG) and is intended to be included as a new module in the next major release of the CityGML standard.

Keywords

ogcdocs, FCP1, CityGML, Dynamizer, Timeseries, Sensor Web Enablement

Proposed OGC Working Group for Review and Approval

CityGML SWG

Chapter 1. Introduction

1.1. Scope

This Engineering Report (ER) focuses on a scenario requiring the support of real-time sensors and other time-dependent properties within semantic 3D city models based on the CityGML standard. During FCP1, guidelines were developed for the following concepts with respect to the scenario.

- Developing prototype capabilities that associate sensor readings (e.g. air quality sensors, weather information, electricity consumption) or other aggregated indicators (e.g. solar potential power, energy performance indicators) to elements in the city models.
- Making sensor and dynamic data available through interoperable OGC web services.
- Extending static city models by supporting highly dynamic properties and real-time sensor observations.
- Planning and conducting a final demonstration using the pilot scenarios.

1.2. Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Table 1. Contacts

Name	Organization
Kanishk Chaturvedi	Technical University of Munich, Germany
Thomas H. Kolbe	Technical University of Munich, Germany
Tatjana Kutzner	Technical University of Munich, Germany
Andreas Donaubauer	Technical University of Munich, Germany
Guy Schumann	Remote Sensing Solutions Inc., U.S.A.
Eve Ross	Remote Sensing Solutions Inc., U.S.A.
Mohsen Kalantari	University of Melbourne, Australia
Bruno Willenborg	Technical University of Munich, Germany
Maximilian Sindram	Technical University of Munich, Germany
Claus Nagel	virtualcitySYSTEMS GmbH, Germany
Emmanuel Devys	Institut National de l'Information Géographique et Forestière - IGN, France

1.3. Future Work

The work described by this document serves as a basis for the development of the next major version of CityGML (CityGML 3.0). The Dynamizer ADE is intended to become a part of the CityGML 3.0.

This work may also be of importance to the TimeseriesML Standards Working Group for future revisions to the TimeseriesML standard.

No future work is planned to this document, but a number of work items and recommendations have been identified that shall be addressed in future initiatives, see Section 10.2.

1.4. Foreword

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

Chapter 2. References

The following documents are referenced in this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

- OGC: OGC 06-021r4, OGC® Sensor Web Enablement (SWE) Architecture, 2008
- OGC: OGC 10-004r3, OGC® Observations and Measurements, Version 2.0, 2013
- OGC: OGC 12-000, OGC® SensorML: Model and XML Encoding Standard, Version 2.0, 2014
- OGC: OGC 12-006, OGC® Sensor Observation Service Interface Standard, Version 2.0, 2012
- OGC: OGC 12-019, OGC® City Geography Markup Language (CityGML) Encoding Standard, Version 2.0.0, 2012
- OGC: OGC 15-043r3, OGC® TimeseriesML 1.0 - Timeseries Profile of Observations and Measurements, Version 1.0, 2016
- OGC: OGC 15-078r6, OGC SensorThings API Part 1: Sensing, Version 1.0, 2016
- OGC: OGC-16-097, FCP1: Recommendations on Mapping IFC/CityGML to 3DIM Engineering Report, 2017
- OGC: OGC-16-099, FCP1: Urban planning rules checking using WPS to WPS SWG Engineering Report, 2017
- OGC: OGC 16-130, The OGC CityGML EA UML Model – An ISO-compliant definition of the CityGML 2.0 UML model using Enterprise Architect [*Not yet published*]
- ISO: ISO 19108:2002, Geographic Information – Temporal Schema, 2002
- ISO: ISO 19123:2005, Geographic information — Schema for coverage geometry and functions, 2005

Chapter 3. Terms and definitions

For the purposes of this report, the definitions specified in Clause 4 of the CityGML Encoding Standard [OGC 12-019] shall apply.

3.1. Abbreviated terms

- ADE - Application Domain Extension
- API - Application Programming Interface
- bSI - buildingSMART International
- CityGML - City Geography Markup Language
- Climate KIC - Knowledge & Innovation Community on Climate Change and Mitigation
- COLLADA - COLLAborative Design Activity
- EIT - European Institute of Innovation & Technology
- ER - Engineering Report
- FCP1 - Future City Pilot Phase 1
- glTF - GL Transmission Format
- GML - Geography Markup Language
- IFC - Industry Foundation Classes
- IoT - Internet of Things
- ISO - International Organization for Standardization
- KML - Keyhole Markup Language
- LoD - Level of Detail
- OCL - Object Constraint Language
- OGC - Open Geospatial Consortium
- O&M - Observations & Measurements
- SOS - Sensor Observation Service
- SRDBMS - Spatial Relational Database Management System
- SSD - Smart Sustainable Districts
- SWE - Sensor Web Enablement
- UML - Unified Modeling Language
- W3C - Web 3D Consortium
- WFS - Web Feature Service
- XML - Extensible Markup Language
- XPath - XML Path Language

Chapter 4. Overview

This ER is a starting point to understand the Dynamizer concept. The concept has been implemented as an Application Domain Extension (ADE) of the CityGML standard. The ER describes the overall development and implementation process of this concept in order to solve the real life issues identified within the scope of the FCP1 scenarios.

The main document starts with a short overview of the FCP1, the scenarios put forward by the pilot sponsors, and the respective solutions developed by the pilot participants. However, the details of each solution are provided in the individual ERs prepared by the participants as mentioned in Chapter 5. This ER focuses on the scenario requiring the support of real-time sensor observations and other time-dependent properties within semantic 3D city models. Chapter 6 provides an overview of the new concept 'Dynamizer' which has been conceptualized in order to extend semantic 3D city models by representing highly dynamic data in different ways and providing a method for injecting dynamic variations of city object properties into the static representation. The Dynamizer concept has been implemented as an ADE of the CityGML standard. During the pilot, the UML model of the Dynamizer ADE was developed. The details and explanations of each UML class have been described in Chapter 7. Chapter 8 shows the XML schema definition files derived from the UML model. The XML schema definition files were used for creation of the CityGML instance documents used in the pilot. Chapter 9 lists the components and the software systems used within the pilot, followed by the implementations which were used for the result demonstrations. In the end, Chapter 10 provides an outlook on the achievements and future activities that require more research to further increase the usability of dynamizers in different applications.

Chapter 5. FCP1 scenarios and requirements

This section provides a brief overview of different scenarios and objectives defined within the FCP1. The objective of the OGC pilot project was to demonstrate how the use of the international standards such as CityGML and IFC together can provide stakeholders with information, knowledge and insight which enhances financial, environmental, and social outcomes for citizens living in cities. During the pilot, three scenarios were set up based on the real-world requirements put forward by the pilot sponsors and solutions were developed and demonstrated by the pilot participants. A short overview of the FCP1 scenarios is provided below.

1. Interoperability between the standards CityGML and IFC: CityGML [OGC 12-019] is an international standard for representing 3D city and landscape models. IFC [1] describes building information. This scenario is focused on providing interoperability between the CityGML and the IFC standards. During the pilot, the solutions were developed by University of Melbourne to map the IFC file to the CityGML file, which can further be validated based on Web Processing Service standard against a set of urban planning requirements. More details about this scenario are documented in [OGC 16-097] and [OGC 16-099].
2. Inundation Modeling with 3D city models: This scenario highlights on inundation models developed by Remote Sensing Solutions Inc. (U.S.A.) and collaborating partners that can simulate real-time water flow characteristics over time and space at multiple resolutions (e.g., cm to km grid spacing). The solutions involved transformation from traditional inundation display to an interoperable city model inundation layer in the CityGML standard. In this way, detailed flooding scenarios can be developed, e.g., to determine which city objects such as buildings are at risk of flooding. The details of this scenario are provided by by Remote Sensing Solutions Inc. (U.S.A.) and can be found in Appendix A.
3. Making city models dynamic: This ER focuses on the results and findings of this scenario, highlighting how dynamic city models can provide better services to the citizens as well as help for performing better analysis. The solutions were provided using the Dynamizer ADE of the CityGML standard, developed by the Chair of Geoinformatics, Technical University of Munich. During the pilot, two use cases were identified, as described below.

5.1. Integrating sensors with semantic 3D city models

This use case was set up by Ordnance Survey Great Britain and was based on the Royal Borough of Greenwich, London. The main objectives were (i) to provide better services to the citizens of the Royal Borough of Greenwich; (ii) to enable departments within the Royal Borough of Greenwich to share and coordinate data more effectively and to be recognized as an example of use of smarter working practices; and (iii) to improve collaboration between London boroughs by creating more interoperable data, content, and insight. The use of interoperable standards creates the opportunity to develop a cross-department platform to collaborate and share data. Furthermore, the potential integration with real-time sensor observations (e.g., monitoring humidity, temperature, etc.) within council housing can lead to decisions for matching human needs to the right housing/resources.

During the pilot, the 3D building objects were developed according to the CityGML LoD1 specifications, which were enriched by various thematic properties such as building address, details of the building residents, adult care, and housing stock information. The links to multiple sensors were defined in the CityGML building objects using dynamizers. The implementation and demonstration details can be found in Section 9.2.1.

5.2. Integrating time-dependent properties with semantic 3D city models

This use case was set up by IGN and virtualcitySYSTEMS GmbH and was based in the commune of Bruz, located 11 km southwest of Rennes in Brittany, France and is a part of Rennes Metropole. The main objectives were to provide better services to the citizens and the energy planners by making sophisticated solar potential analysis. The use case aimed at answering questions such as (i) *How many buildings have solar irradiation of more than a specific unit (e.g. 5000 MWh)?*, (ii) *Which buildings are well suited for installing solar panels?*, and *(iii) How does the irradiation for a building vary through the year?*.

During the pilot, the buildings and their wall and roof surfaces were registered according to the IGN REF3DNAT specification based on the CityGML LoD2 profile. The solar irradiation values for the roofs and facades were calculated using the Solar Potential Analysis tool (c.f. section 9.1.6). The simulation tool allows estimation of the solar power from direct, diffuse, and global sunlight irradiation for individual months and years. Such time-dependent values can now be represented within the CityGML datasets in standardized ways using dynamizers. The implementation details are provided in Section 9.2.2.

Chapter 6. Making city models dynamic

The current generation semantic 3D city models are static in nature and do not support time-dependent properties. Apart from the scenarios mentioned in Chapter 5, there are many other 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) history or evolution of cities such as construction or demolition of buildings and (ii) managing multiple versions of the city models. [2] propose an approach based on the CityGML standard to manage versions and history within semantic 3D city models. Some of the changes may also represent high frequency 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); and (iii) real-time sensor observations (e.g., air quality sensors, weather stations, or smart meters). In this case, only some of the properties of otherwise static objects need to represent such time-varying values. Dynamizer is a new concept supporting the latter types of changes. Within the FCP1, the concept has been implemented to extend the CityGML information model using the built-in mechanism of the Application Domain Extension (ADE) in order to dynamize features and properties allowing for enriching the city model by data from dynamic data feeds.

6.1. Dynamizer - Introduction

Dynamizer allows modeling and integrating dynamic properties within semantic 3D city models. As shown in figure 1, the dynamizer serves three main purposes, as described below.

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; or by referencing an external file. These values can be obtained from sensors, simulation specific databases, and also external files such as CSV or Excel spreadsheets.
2. Dynamizer delivers a method to enhance static city models by dynamic property values. It references a specific property (e.g. spatial, thematic, or appearance properties) 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 sensors. 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 an 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. More details on this concept can be found in peer-reviewed papers [3] and [4] .

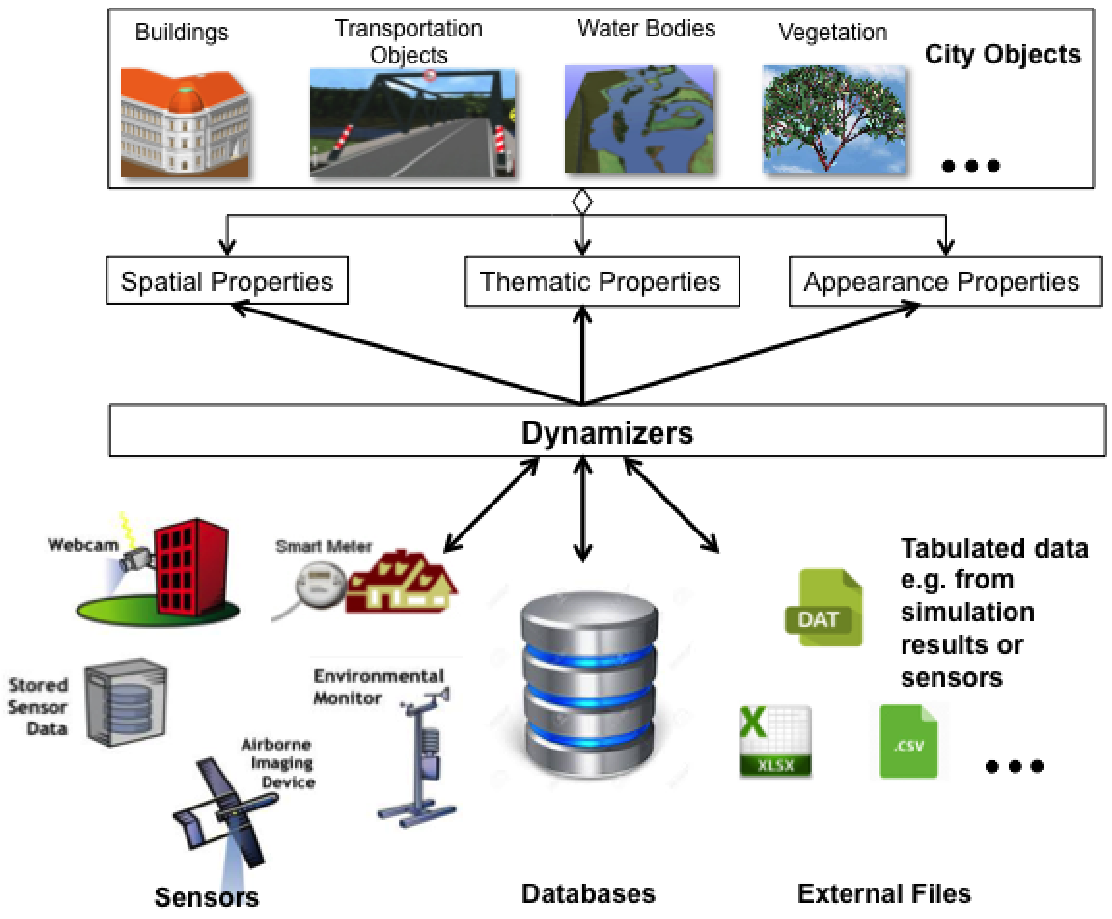


Figure 1. Conceptual representation of Dynamizers allowing (i) the representation of time-variant values from sensors, simulation specific databases, and external files and (ii) enhancing the properties of city objects by overriding their static values. Image taken from [3].

Chapter 7. Development of the UML Class Diagram for the Dynamizer ADE

Within the FCP1, dynamizers were implemented as an 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 the CityGML standard (version 2.0).

Figure 2 shows the complete UML model of the Dynamizer ADE. The color scheme paints classes in different colors, depending on the UML package they belong to, and is defined as follows. A Class painted in yellow belongs to a CityGML UML package. A Class painted in light blue belongs to the GML package different to that associated with the yellow color. Classes painted in orange are newly introduced classes implemented as an ADE of the CityGML standard. However, there are two new classes which are not a part of [OGC 12-019] and belong to other OGC standards. Classes shown in dark blue represent OGC TimeseriesML 1.0 [OGC 15-043r3] and classes in pink belong to OGC Sensor Observation Service [OGC 12-006].

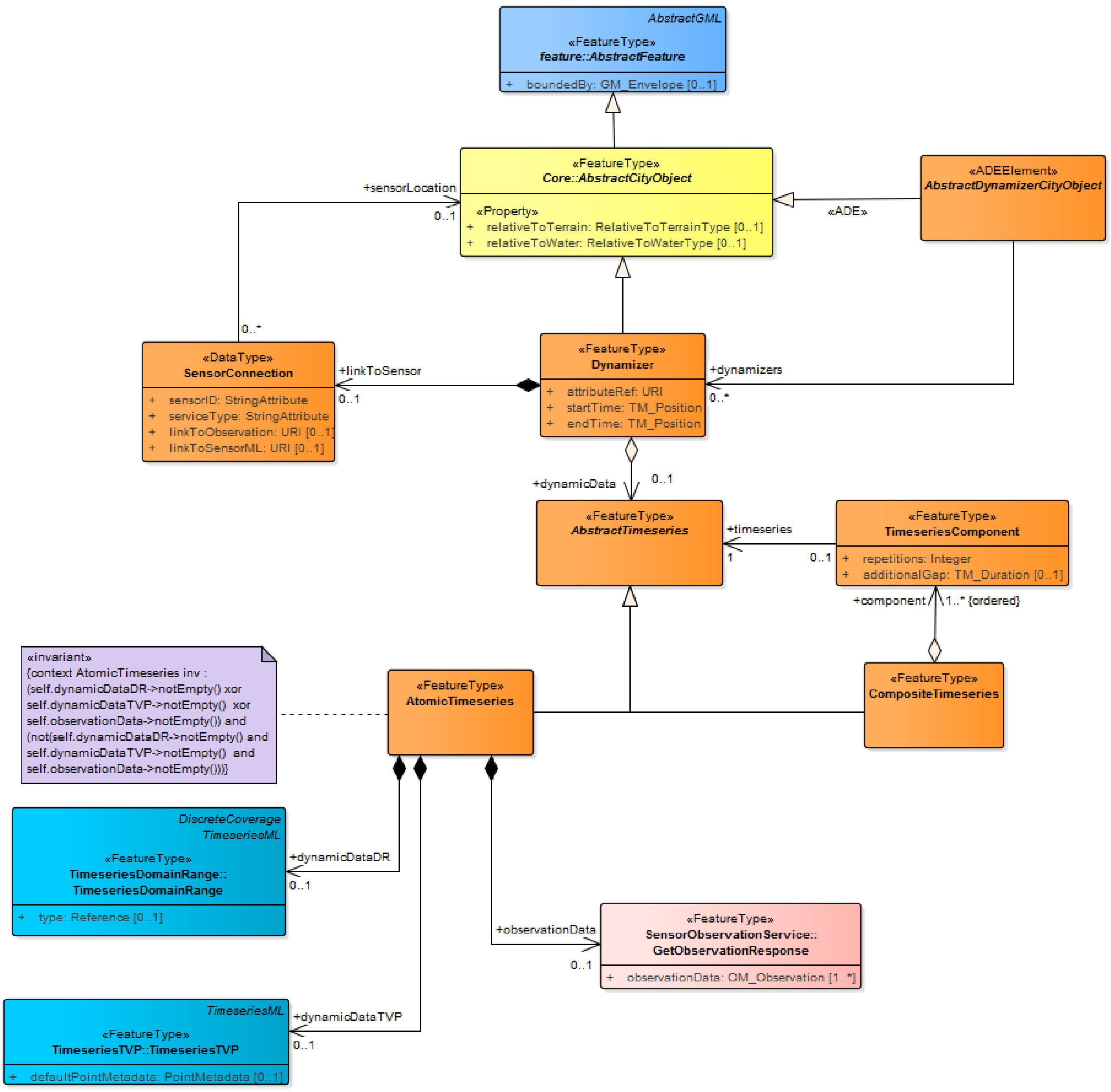


Figure 2. Complete UML model of the Dynamizer ADE.

The individual components of the Dynamizer ADE UML model are explained in the following sub-sections.

7.1. Dynamizer - A new FeatureType

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*. This means that all the city objects such as buildings, roads, vegetation, etc. can now include their Dynamizer features either inline or as 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 [5]. XPath is a W3C recommendation used to navigate through the 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 [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 allows representing time-variant or dynamic values in different and generic ways. The timeseries may be modeled in two ways: (i) *AtomicTimeseries* and (ii) *CompositeTimeseries* (see Figure 3).

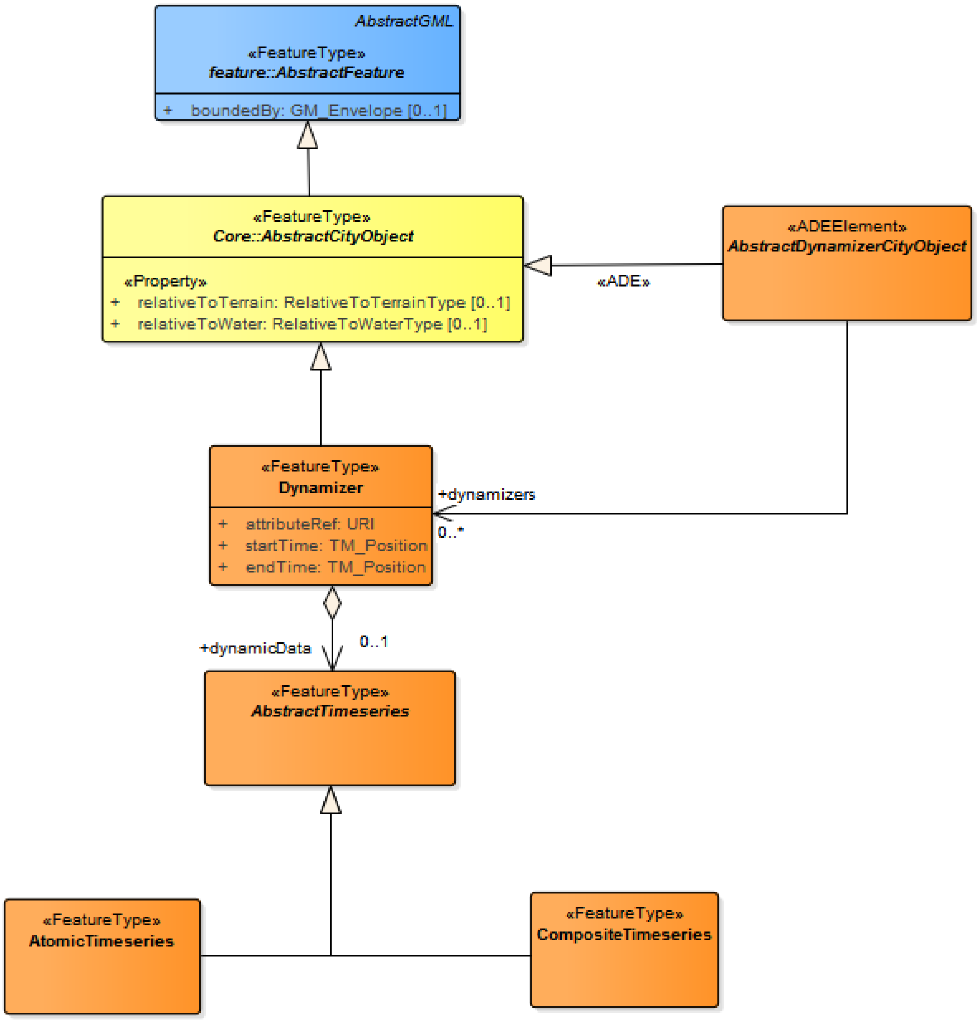


Figure 3. Dynamizers modeled as a new FeatureType

7.2. Atomic Timeseries

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) real-time sensor observations (e.g., air quality sensors and smart meters). The dynamizers provide an explicit way to model such dynamic variations using *AtomicTimeseries*. They utilize different encodings defined according to OGC TimeseriesML 1.0 [OGC 15-043r3]. Utilizing TimeseriesML, the timeseries can be represented as interleaved time/value pairs or by a domain range encoding which is supported by *dynamicdataTVP* and *dynamicDataDR* (see figure 4). Since the TimeseriesML domain range encoding is an implementation of the [ISO 19123:2005] Discrete Coverages, it allows defining different data types for the 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, which helps in mapping missing values or multiple values to specific time points.

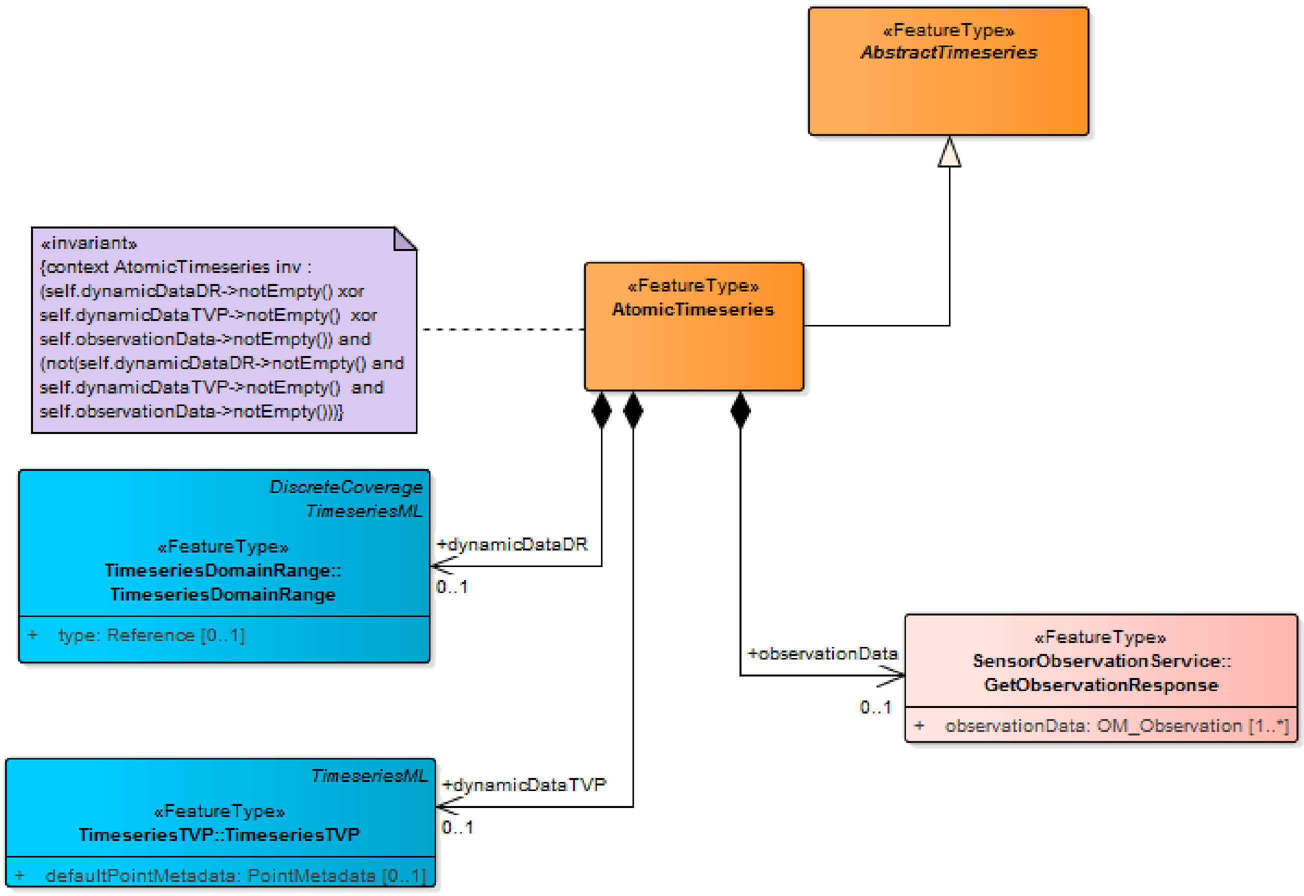

«FeatureType»
AbstractTimeseries
«invariant»
{context AtomicTimeseries inv :
(self.dynamicDataDR->notEmpty() xor
self.dynamicDataTVP->notEmpty() xor
self.observationData->notEmpty()) and
(not(self.dynamicDataDR->notEmpty() and
self.dynamicDataTVP->notEmpty() and
self.observationData->notEmpty()))}
«FeatureType»
AtomicTimeseries
DiscreteCoverage
TimeseriesML
«FeatureType»
TimeseriesDomainRange::
TimeseriesDomainRange
+ type: Reference [0..1]
+dynamicDataDR
0..1
+observationData
0..1
«FeatureType»
SensorObservationService::
GetObservationResponse
+ observationData: OM_Observation [1..*]
TimeseriesML
«FeatureType»
TimeseriesTVP::TimeseriesTVP
+ defaultPointMetadata: PointMetadata [0..1]
+dynamicDataTVP
0..1

Figure 4. Representation of Atomic Timeseries, supporting different encodings from OGC TimeseriesML 1.0 as well as supporting sensor observations encoded in OGC O&M from a Sensor Observation Service response.

Apart from TimeseriesML, *AtomicTimeseries* also allows for the inline representation of the Observations and Measurements (O&M) data within a timeseries using *observationData*. O&M [OGC 10-004r3] is one of the core standards for the response models of the OGC Sensor Web Enablement (SWE)-based standards such as Sensor Observation Service [OGC 12-006] and SensorThings API [15-078r6]. In this way, dynamizers can represent real-time sensor observations. 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 is a declarative language for describing rules that apply to UML models. The constraint is defined as:

```
context AtomicTimeseries inv :
    (self.dynamicDataDR->notEmpty() xor self.dynamicDataTVP->notEmpty() xor
self.observationData->notEmpty())
    and
    (not(self.dynamicDataDR->notEmpty() and self.dynamicDataTVP->notEmpty() and
self.observationData->notEmpty()))
```

According to the expression, this OCL constraint would result in true only when exactly one of the properties has a value [6].

7.3. Composite Timeseries

Dynamizers support absolute start and end points referencing a specific time reference system. The absolute time points can be mapped to the attribute values and can be represented as tabulation of the measured data. One common example illustrating such scenario is mapping of the 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. The applications 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 the 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.

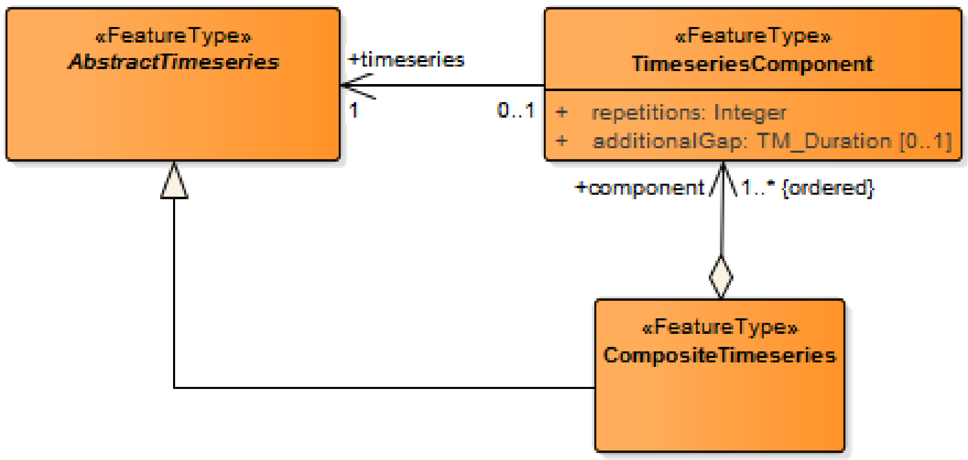


Figure 5. Representation of Composite Timeseries, allowing representing patterns of dynamic variations of properties.

In order to support such patterns, dynamizers include the concept of composite timeseries as shown in figure 5. *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. *repetitions* is an integer type which determines how many times the nested timeseries should be iterated. For example, in order to determine the pattern of a building’s electricity consumption for weekdays, a composite timeseries may include five repetitions of atomic timeseries of a single weekday consumption. It also contains an attribute *additionalGap*, which is a type TM_Duration. It allows defining customized patterns by providing the gaps within the existing timeseries. For instance, for an entire monthly timeseries of energy consumption for all days of a week, the gaps can be provided for the weekends in order to define the patterns of energy consumption only for the weekdays. Furthermore, this attribute also allows connecting nonoverlapping timeseries that have been separately collected, 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 the time positions according to a relative time reference system. According to ISO 19108, they may be defined as

TM_OrdinalEras within *TM_OrdinalReferenceSystems*. The use of composite timeseries allows defining local reference systems for the specific use cases. The demonstration of composite timeseries has not been covered within the FCP1, however, its details and examples can be found in [4].

7.4. Sensors and observations

Apart from different timeseries representations, dynamizers provide support for sensors and sensor observations within them. Within the OGC SWE standards suite [OGC 06-021r4], sensor descriptions are encoded in the SensorML format [OGC 12-000] and sensor observations in the O&M format [OGC 10-004r3]. The web services such as Sensor Observation Service and SensorThings API allow retrieval of the sensor descriptions and observations using different requests. Dynamizers leverage such standards and allow integrating sensors with semantic 3D city models in two ways.

- **By including sensor observations within dynamizers**: The different standards such as SOS and SensorThings API allow encoding sensor observations in the O&M format. Dynamizers support inlining of the O&M data representing sensor observation values using AtomicTimeseries [c.f. section 7.2], which, are then injected into the attributes of the specific city objects. This approach is useful to exchange sensor readings for a past time period together and consistently within the city model. However, in the case of very high frequent observations (e.g., sensor readings at every 30 seconds), data management within the databases may lead to storage related issues.
- **By linking dynamizers with sensors:** In order to avoid such storage related issues in the case of high frequent observations, dynamizers support direct links to sensors and observations utilizing different sensor based services such as SOS and SensorThings API. There are already well-defined data models and implementations available for storing and managing large and frequent sensor observations (see section 9.1.7). The sensor based services such as SOS include different requests: e.g., *DescribeSensor* and *GetObservation*. With such requests, sensor descriptions and observations are retrieved in standardized ways and their URL links can also be defined within the 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.

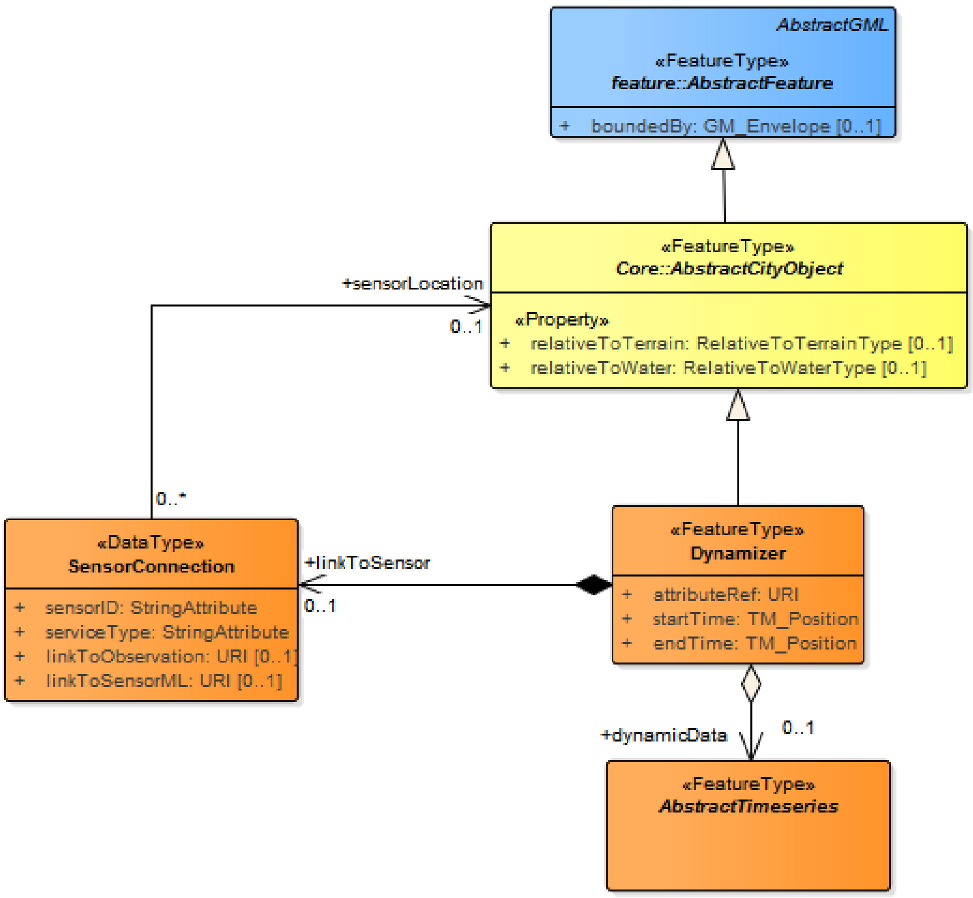


Figure 6. Representation of the data type SensorConnection, allowing to define the details of sensor based services within the CityGML document.

As shown in figure 6, dynamizers provide links to external sensors with the help of the new datatype *SensorConnection*. It contains four attributes: (i) *sensorID* - the unique identification of the sensor within the referenced sensor service, (ii) *serviceType* - the type of sensor based service such as SOS and SensorThings API, (iii) *linkToObservation* - URL link to the request for retrieving sensor observations, and (iv) *linkToSensorML* - URL link to the request for retrieving the sensor descriptions and metadata. OGC SOS involves different requests for retrieving the sensor descriptions and observations. *DescribeSensor* is used to retrieve sensor descriptions and metadata in the SensorML format. *GetObservation* is used to retrieve sensor observations encoded in the O&M format. The request parameter also allows the specification of spatial and temporal filters. The association *sensorLocation* allows specifying which city object this sensor belongs to. For example, if a solar panel is installed on a building roof surface with a sensor estimating heat demand of the building, the datatype *SensorConnection* would not only provide direct links to the sensor description and observations but also specify the link to the roof surface of the CityGML building object on which the solar panel is installed.

Chapter 8. Dynamizer ADE XML Schema

This section includes the XML schema definition for the Dynamizer ADE. For the conceptual schema development, the model driven approach based on the ISO 19100 series of standards is applied, which allows for automatically deriving the XML schema documents from the conceptual schema. Upon development of the UML conceptual schema of the Dynamizer ADE, the XML schema was derived using the tool ShapeChange (see section 9.1.2). The details of the automatic derivation of the XML schema will be published in [OGC 16-130]. Based on the XML schema documents, valid CityGML Dynamizer ADE instance documents were developed according to specified use cases and used for the demonstrations as described in section 9.2.

Header of the Dynamizer ADE Schema definition file

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:dyn="http://www.citygml.org/ade/dynamizer_ade/1.0"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.citygml.org/ade/dynamizer_ade/1.0"
    elementFormDefault="qualified"
    version="1.0"
  xmlns:tsml="http://www.opengis.net/tsml/1.0"
  xmlns:sos="http://www.opengis.net/sos/2.0"
  xmlns:core="http://www.opengis.net/citygml/2.0"
  xmlns:gen="http://www.opengis.net/citygml/generics/2.0"
  xmlns:gml="http://www.opengis.net/gml" >
  <xsd:import namespace="http://www.opengis.net/tsml/1.0"
    schemaLocation="http://schemas.opengis.net/tsml/1.0/timeseriesML.xsd"/>
  <xsd:import namespace="http://www.opengis.net/sos/2.0"
    schemaLocation="http://schemas.opengis.net/sos/2.0/sosGetObservation.xsd"/>
  <xsd:import namespace="http://www.opengis.net/citygml/2.0"
    schemaLocation="http://schemas.opengis.net/citygml/2.0/cityGMLBase.xsd"/>
  <xsd:import namespace="http://www.opengis.net/citygml/generics/2.0"
    schemaLocation="http://schemas.opengis.net/citygml/generics/2.0/generics.xsd"/>
  <xsd:import namespace="http://www.opengis.net/gml"
    schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
  <xsd:import namespace="http://www.opengis.net/om/2.0"
    schemaLocation="http://schemas.opengis.net/om/2.0/observation.xsd"/>
...
</xsd:schema>
```

DynamizerPropertyType

```
<xsd:element name="dynamizers"
             substitutionGroup="core:_GenericApplicationPropertyOfCityObject"
             type="dyn:DynamizerPropertyType"/>
<!-- ================================================= -->
<xsd:complexType name="DynamizerPropertyType">
  <xsd:sequence minOccurs="0">
    <xsd:element ref="dyn:Dynamizer"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xsd:complexType>
```

DynamizerType, Dynamizer

```
<xsd:element name="Dynamizer" substitutionGroup="core:_CityObject"
             type="dyn:DynamizerType"/>
<!-- ================================================= -->
<xsd:complexType name="DynamizerType">
  <xsd:complexContent>
    <xsd:extension base="core:AbstractCityObjectType">
      <xsd:sequence>
        <xsd:element name="attributeRef" type="xsd:anyURI"/>
        <xsd:element name="startTime" type="gml:TimePositionType"/>
        <xsd:element name="endTime" type="gml:TimePositionType"/>
        <xsd:element minOccurs="0" name="dynamicData"
                     type="dyn:AbstractTimeseriesPropertyType"/>
        <xsd:element minOccurs="0" name="linkToSensor"
                     type="dyn:SensorConnectionPropertyType"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

SensorConnectionType, SensorConnection

```
<xsd:element name="SensorConnection" substitutionGroup="gml:_Object"
             type="dyn:SensorConnectionType"/>
<!-- ================================================= -->
<xsd:complexType name="SensorConnectionType">
  <xsd:sequence>
    <xsd:element name="sensorID" type="xsd:string"/>
    <xsd:element name="serviceType" type="xsd:string"/>
    <xsd:element minOccurs="0" name="linkToObservation" type="xsd:anyURI"/>
    <xsd:element minOccurs="0" name="linkToSensorML" type="xsd:anyURI"/>
    <xsd:element minOccurs="0" name="sensorLocation"
                 type="core:AbstractCityObjectType"/>
  </xsd:sequence>
</xsd:complexType>
<!-- ================================================= -->
<xsd:complexType name="SensorConnectionPropertyType">
  <xsd:sequence>
    <xsd:element ref="dyn:SensorConnection"/>
  </xsd:sequence>
</xsd:complexType>
```

AbstractTimeseriesType, AbstractTimeseries

```
<xsd:element abstract="true" name="AbstractTimeseries"
             substitutionGroup="gml:_Feature"
             type="dyn:AbstractTimeseriesType"/>
<!-- ================================================= -->
<xsd:complexType abstract="true" name="AbstractTimeseriesType">
  <xsd:complexContent>
    <xsd:extension base="gml:AbstractFeatureType">
      <xsd:sequence/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<!-- ================================================= -->
<xsd:complexType name="AbstractTimeseriesPropertyType">
  <xsd:sequence minOccurs="0">
    <xsd:element ref="dyn:AbstractTimeseries"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xsd:complexType>
```

AtomicTimeseriesType, AtomicTimeseries

```
<xsd:element name="AtomicTimeseries" substitutionGroup="dyn:AbstractTimeseries"
             type="dyn:AtomicTimeseriesType"/>
<!-- ================================================= -->
<xsd:complexType name="AtomicTimeseriesType">
  <xsd:complexContent>
    <xsd:extension base="dyn:AbstractTimeseriesType">
      <xsd:sequence>
        <xsd:element minOccurs="0" name="dynamicDataTVP"
                     type="tsml:TimeseriesTVPPropertyType"/>
        <xsd:element minOccurs="0" name="dynamicDataDR"
                     type="tsml:TimeseriesDomainRangePropertyType"/>
        <xsd:element minOccurs="0" name="observationData"
                     type="sos:GetObservationResponsePropertyType"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<!-- ================================================= -->
<xsd:complexType name="AtomicTimeseriesPropertyType">
  <xsd:sequence minOccurs="0">
    <xsd:element ref="dyn:AtomicTimeseries"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xsd:complexType>
```

CompositeTimeseriesType, CompositeTimeseries

```
<xsd:element name="CompositeTimeseries" substitutionGroup="dyn:AbstractTimeseries"
             type="dyn:CompositeTimeseriesType"/>
<!-- ================================================= -->
<xsd:complexType name="CompositeTimeseriesType">
  <xsd:complexContent>
    <xsd:extension base="dyn:AbstractTimeseriesType">
      <xsd:sequence>
        <xsd:element maxOccurs="unbounded" name="component"
                     type="dyn:TimeseriesComponentPropertyType"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<!-- ================================================= -->
<xsd:complexType name="CompositeTimeseriesPropertyType">
  <xsd:sequence minOccurs="0">
    <xsd:element ref="dyn:CompositeTimeseries"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xsd:complexType>
```

TimeseriesComponentType, TimeseriesComponent

```
<xsd:element name="TimeseriesComponent" substitutionGroup="gml:_Feature"
             type="dyn:TimeseriesComponentType"/>
<!-- ================================================= -->
<xsd:complexType name="TimeseriesComponentType">
  <xsd:complexContent>
    <xsd:extension base="gml:AbstractFeatureType">
      <xsd:sequence>
        <xsd:element name="repetitions" type="xsd:positiveInteger"/>
        <xsd:element minOccurs="0" name="additionalGap" type="xsd:duration"/>
        <xsd:element name="timeseries" type="dyn:AbstractTimeseriesPropertyType"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<!-- ================================================= -->
<xsd:complexType name="TimeseriesComponentPropertyType">
  <xsd:sequence minOccurs="0">
    <xsd:element ref="dyn:TimeseriesComponent"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xsd:complexType>
```

Chapter 9. FCP1 Demonstrations

This section provides a brief overview of the components used and the solutions developed for demonstrating the FCP1 scenarios.

9.1. Components development

In this section, the details of the software systems and components have been provided which were used and also modified for the demonstrations of the FCP1 scenarios. The list of the components used is as follows.

9.1.1. Enterprise Architect

Enterprise Architect (EA) [7] is a UML design tool. Since EA supports a model-driven approach for defining geospatial data, the UML data models of the CityGML standard have already been developed using EA. The new dynamizer ADE classes were developed using this software, which were stored using the .eap file name extension.

9.1.2. ShapeChange

ShapeChange[8] is an open-source Java tool which can process the UML models for geographic information and derives the GML application schemas (and other transfer formats) from these UML models. The tool can directly read the UML models defined using EA via the EA Java API. The guidelines and examples for how to derive the XML schemas from the EA file and for how to create the ADEs and to derive the corresponding XML schema using ShapeChange will be documented in [OGC 16-130].

9.1.3. Feature Manipulation Engine (FME)

FME [9] is a software developed by Safe Software and facilitates the transformation of spatial data into a variety of formats, data models, and repositories for transmission to end users. This software is widely used for reading and writing different geospatial data formats including CityGML and CityGML ADEs. For creation of the CityGML instance documents including the Dynamizer ADE, the FME transformer 'XMLTemplater' was used. This transformer takes an XML schema definition file as input and creates sample XML instance documents. However, FME is a proprietary software and requires an appropriate license. There is an open source Java API called 'citygml4j' [10] which can also be used for creating instance CityGML documents along with their ADEs.

9.1.4. 3D City Database

In order to store and manage the CityGML datasets, an open-source geodatabase called '3DCityDB' [11] is used. 3DCityDB stores, represents, and manages the large CityGML datasets on top of a standard spatial relational database management systems (SRDBMS) such as Oracle Spatial and PostgreSQL. It provides a Java front-end application named '3DCityDB Importer/Exporter', which allows for high performance importing and exporting the CityGML datasets with arbitrary file sizes. It also allows exporting the contents in the form of different visualization formats such as KML, COLLADA, and glTF, allowing the 3D objects to be viewed and interactively explored in the web applications. For integration into an OGC Web Service environment, the 3DCityDB contains a Web Feature Service (WFS) interface, using which the CityGML features can be requested in standardized ways. 3DCityDB also allows extending the functionalities in a modular way by the installation of plugins, which add specific abilities to interact with the 3D city database. For instance, by using the Spreadsheet Generator Plugin, arbitrary subsets of the city model data such as generic attributes can be exported in tabular form having the selected attributes from 3D city database instance whether as a CSV file or directly be uploaded as a Google Spreadsheet Document or Google Fusion Table. Furthermore, 3DCityDB also provides a functionality to validate CityGML documents.

9.1.5. 3DCityDB Visualization Clients

For high-performance 3D visualization and interactive exploration of arbitrarily large semantic 3D city models based on the CityGML standard, there are different web-based visualization clients available and used within the pilot. 3DCityDB-Web-Map-Client Pro ([12]) developed by the Chair of Geoinformatics, Technical University of Munich, is a web-based front-end client of 3DCityDB, which not only allows exploring and interacting with large semantic 3D city models, but also provides thematic querying capabilities on the 3D objects. It supports linking the 3D visualization models (KML/glTF) with the cloud-based Google Spreadsheet documents or Google Fusion Table allowing for querying the thematic data of every 3D object. virtualcityMAP [13] is also a web based visualization client developed by virtualcitySYSTEMS GmbH for working with large semantic 3D city models. 3DCityDB-Web-Map-Client [14] is a free and open-source visualization client developed by the Chair of Geoinformatics, Technical University of Munich in cooperation with virtualcitySYSTEMS GmbH. This client provides rich 3D visualization and interactive exploration of arbitrarily large semantic 3D city models based on the CityGML standard. However, it does not support querying capabilities unlike other mentioned visualization clients.

All of the visualization clients use the Cesium [15] virtual globe as their visualization engines. Cesium is an open source JavaScript package supporting the presentation of 3D contents within the web browser where users can dynamically switch between 3D globe visualization and 2D map projection. It utilizes HTML5 and WebGL to provide hardware acceleration and plugin independence and provides cross-platform, cross-browser, and cross-device functionalities.

9.1.6. Solar Potential Analysis Tool

The Solar Potential Analysis Tool [16] is a simulation tool developed by the Chair of Geoinformatics, Technical University of Munich for assessing and estimating solar energy production for the roofs and facades of the 3D building objects in different ways. The simulation tool operates on the semantic 3D city models defined according to the CityGML standard. By combining a transition model, sun position calculation, and an approximation of the sky dome, the solar power from direct, diffuse, and global sunlight irradiation are estimated for individual months and years. The shadowing effects of the surrounding topographic features are considered by applying a ray tracing approach. The Sky View Factor (SVF), a measure indicating the visible fraction of the sky hemisphere, is determined for each surface.

As a result, each building surface is being enriched by its individual irradiation values. These are also aggregated to the building level. Finally, a point cloud is generated from sampling points that have been generated for each building surface for the simulation. Each point is parameterized with the direct, diffuse, and global irradiation values over the different months and can thus, be visualized in different colors according to the respective solar power as shown in figure 7.

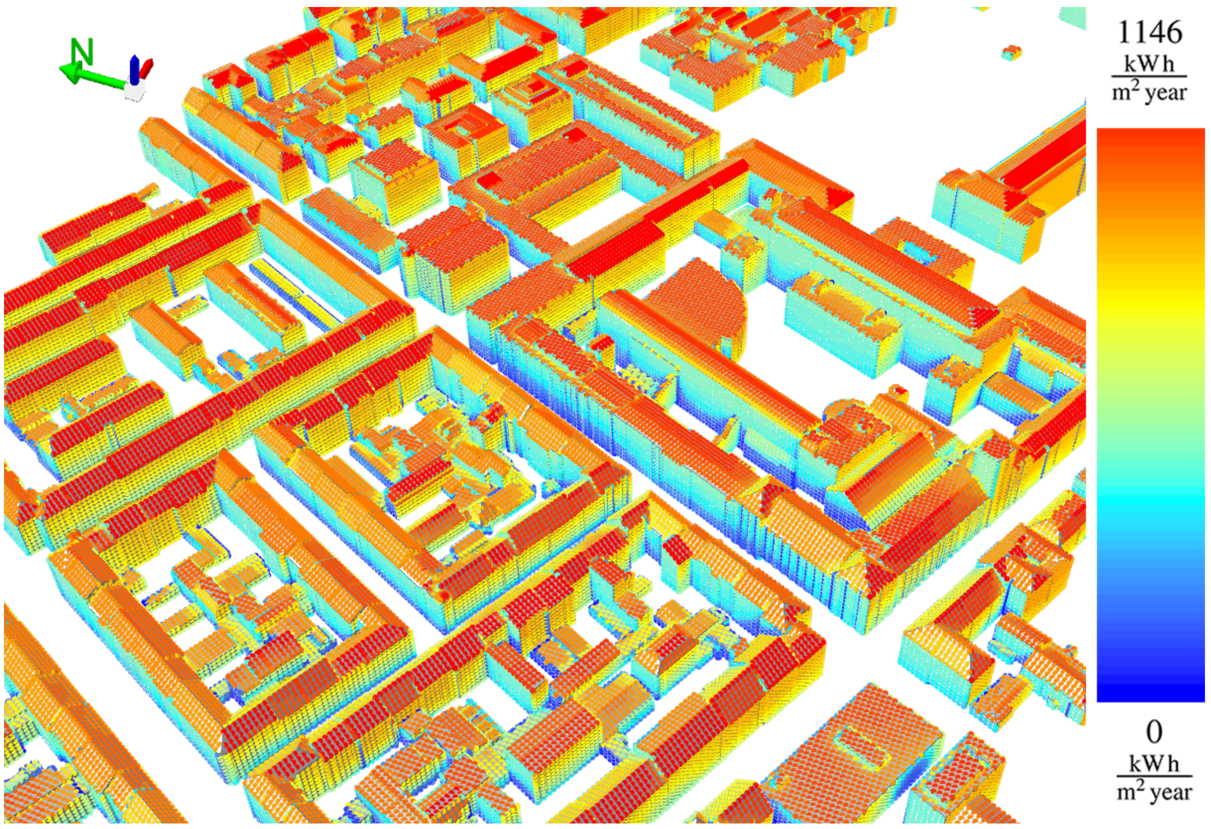


Figure 7. Illustration of yearly global irradiation sum for the building facades. Image taken from ([16])

9.1.7. 52° North Sensor Observation Service Implementation

This is an open source software initiative developed by 52°North GmbH, Germany [17], which allows enabling the realization of Sensor Web infrastructures. It is certified by the Open Geospatial Consortium (OGC) for realization of their Sensor Web Enablement (SWE) suite. The software uses an internal database for storing / managing sensor information on the relational database PostgreSQL/PostGIS, and allows:

- Defining multiple ranges of sensors;
- setting up different sensor services (e.g., Sensor Observation Service); and
- providing extra sensor data upload capabilities, either by using the SOS web interface or by using SOS importers supporting CSV/JSON feeds.

This implementation includes a relational data model, which implements the OGC O&M information model. With the help of the data model, sensor data of different types and multiple instances of sensors are stored, managed, and queried. The implementation also provides a visualization client which not only shows different sensor stations on a map, but also lets the users interact with and query timeseries graphs generated dynamically from the observation properties.

9.2. FCP1 Demonstrations

9.2.1. Integrating sensors with semantic 3D city models

This use case was set up by Ordnance Survey Great Britain and was based on the Royal Borough of Greenwich, London. The main objectives were to (i) provide better services to the citizens of the Royal Borough of Greenwich; (ii) enable departments within the Royal Borough of Greenwich to share and coordinate data more effectively and to be recognized as an example of smarter working practices; and (iii) improve collaboration between London boroughs by creating more interoperable data, content, and insight. The use of interoperable standards creates the opportunity to develop a cross-department platform to collaborate and share data. Furthermore, the potential integration with real-time sensor data (e.g., monitoring humidity, temperature, etc.) within council housing can lead to decisions for matching human needs to the right housing/resources.

Creation of the CityGML datasets

The 3D building objects were created by Ordnance Survey according to the CityGML LoD1 specification. The dataset includes 265,000 building objects, which were generated using the Ordnance Survey MasterMap building footprints. The dataset was further enriched by various thematic properties such as building address, details of the building residents, adult care, and housing stock information. The CityGML documents were validated and stored in the database using 3DCityDB.

Working with sensors

Because of the unavailability of access to the sensors in the Royal Borough of Greenwich, it was discussed and agreed by the pilot sponsors and participants to use a stable sensor-based service in order to demonstrate the capabilities of the dynamizers. For the demonstration purposes, the real-time observations from different weather stations installed in the district Queen Elizabeth Olympic Park, London were used. These weather stations have been set up and operated by 'Intel Labs London' and are being used in the project called Smart Sustainable Districts (SSD) ([18]), which is a Climate-KIC flagship project funded by European Institute of Innovation and Technology (EIT). The weather stations are named after Intel Collaborative Research Institute (ICRI) as ICRI_0001, ICRI_0002, and ICRI_0003 and measure 15 properties in the park including temperature, humidity, wind speed, etc. The observations are recorded every minute. They are accessed from Intel’s platform via a Hypercat registry and encoded using the SenML format. Hypercat and SenML are industry IoT standards that are being developed independent of the OGC SWE standards.

In order to integrate the sensors with the 3D city model, SOS facades were developed for the sensors for retrieving sensor observations and sensor descriptions in a unified interoperable way. For setting up the SOS, the open-source implementation from 52° North (c.f. section 9.1.7) was used.

Integrating sensors with CityGML using the dynamizers

Within the pilot, the OGC Sensor Observation Service (SOS) has been used for linking the weather stations with the CityGML data sets. The SOS supports different requests to retrieve sensor metadata as well as observations, e.g., with the help of the *DescribeSensor* request, sensor metadata (such as sensor names, coordinates, list of observable properties) can be retrieved according to the

SensorML standard. In a similar way using *GetObservation* request, the observations from different properties can be retrieved according to the O&M standard. The direct links to the SOS requests can explicitly be defined within the CityGML datasets using the new *SensorConnection* data type of the Dynamizer ADE (as described in section 7.4). It allows defining the details of the sensor being used such as its unique ID, the type of service, and URLs to retrieve the sensor metadata and observations. Below is an example of a Dynamizer feature defined within a CityGML instance document.

```
<?xml version="1.0" encoding="UTF-8"?>
<core:CityModel xmlns="http://www.citygml.org/citygml/profiles/base/2.0"
 xmlns:core="http://www.opengis.net/citygml/2.0"
 xmlns:bldg="http://www.opengis.net/citygml/building/2.0"
 xmlns:gen="http://www.opengis.net/citygml/generics/2.0"
 xmlns:gml="http://www.opengis.net/gml"
 xmlns:xAL="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0"
 xmlns:dyn="http://www.citygml.org/ade/dynamizer_ade/1.0"
 xmlns:sos="http://www.opengis.net/sos/2.0"
 xmlns:om="http://www.opengis.net/om/2.0"
 xmlns:xlink="http://www.w3.org/1999/xlink"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.opengis.net/citygml/2.0
http://schemas.opengis.net/citygml/2.0/cityGMLBase.xsd
    http://www.opengis.net/citygml/building/2.0
http://schemas.opengis.net/citygml/building/2.0/building.xsd
    http://www.opengis.net/citygml/generics/2.0
http://schemas.opengis.net/citygml/generics/2.0/generics.xsd
    http://www.citygml.org/ade/dynamizer_ade/1.0 CityGML-DynamizerADE.xsd">
<gml:name>Greenwich</gml:name>
<gml:boundedBy>
<gml:Envelope srsName="urn:ogc:def:crs,crs:EPSG:27700,crs:EPSG:5101" srsDimension="3">
<gml:lowerCorner>536741.080000142 170980.299999385 0</gml:lowerCorner>
<gml:upperCorner>548017.10000021 182541.829999454 205.6</gml:upperCorner>
</gml:Envelope>
</gml:boundedBy>
<core:cityObjectMember>
    <bldg:Building gml:id="osgb_7ea64278-a92f-459a-9bea-f905bbcb63e3">
        <gen:stringAttribute name="toid">
            <gen:value>osgb1000041344855</gen:value>
        </gen:stringAttribute>
        <gen:doubleAttribute name="TEMPERATURE">
            <gen:value>19.6</gen:value>
        </gen:doubleAttribute>
        <bldg:lod1Solid>
            ...
        </bldg:lod1Solid>
    </bldg:Building>
</core:cityObjectMember>
<core:cityObjectMember>
    <dyn:Dynamizer gml:id="osgb_7ea64278-a92f-459a-9bea-f905bbcb63e3_Dynamizer">
    <dyn:attributeRef>
```

```
          <!-- Single line XPath Expression -->
          //bldg:Building%5B%40gml:id ='osgb_7ea64278-a92f-459a-9bea-f905bbcb63e3'%5D
          /doubleAttribute%5B%40name = 'TEMPERATURE'%5D
          /gen:value
      </dyn:attributeRef>
      <dyn:startTime>2016-01-01T00:00:00Z</dyn:startTime>
      <dyn:endTime>2017-01-01T00:00:00Z</dyn:endTime>
      <dyn:linkToSensor>
       <dyn:SensorConnection>
        <dyn:sensorID>ICRI_QEOP_0001</dyn:sensorID>
        <dyn:serviceType>SOS 2.0.0</dyn:serviceType>
        <dyn:linkToObservation>
          http://129.187.38.201:8080/52n-sos-webapp-
qeop/service?service=SOS&amp;version=2.0.0&amp;request=GetObservation&amp;featureOfInt
erest=ICRI_QEOP_0001_London&amp;procedure=ICRI_QEOP_0001&amp;observedProperty=Outside_
Temperature&amp;temporalFilter=om:phenomenonTime,2016-07-02T09:00:00Z/2016-07-
02T09:30:00Z
        </dyn:linkToObservation>
        <dyn:linkToSensorML>
          http://129.187.38.201:8080/52n-sos-webapp-
qeop/service?REQUEST=DescribeSensor&amp;SERVICE=SOS&amp;VERSION=2.0.0&amp;PROCEDURE=IC
RI_QEOP_0001&amp;procedureDescriptionFormat=http://www.opengis.net/sensorML/1.0.1
        </dyn:linkToSensorML>
        <dyn:sensorLocation xlink:href="#osgb_7ea64278-a92f-459a-9bea-
f905bbcb63e3"></dyn:sensorLocation>
       </dyn:SensorConnection>
     </dyn:linkToSensor>
    </dyn:Dynamizer>
</core:cityObjectMember>
```

As mentioned in the above example, the building object has a generic attribute TEMPERATURE, which is dynamic in nature and whose value should be retrieved and overridden by the values retrieved according to the Sensor observation Service. The new Dynamizer feature type has been defined for this building object to link real-time sensor observations. The *attributeRef* includes an XPath Expression which refers to the generic attribute TEMPERATURE. The valid XPath expression is:

```
//bldg:Building[@gml:id ='osgb_7ea64278-a92f-459a-9bea-f905bbcb63e3']
/doubleAttribute[@name = 'TEMPERATURE']
/gen:value
```

However, there are some characters which could cause the XML parser to misunderstand the resulting data. Hence, it is necessary to escape such control characters so that the parser can interpret them correctly as data, and not confuse them for markup. For this purpose, the following escaped strings were used for special characters in the CityGML instance document.

Table 2. List of escaped characters

Character	Escaped String
[	%5B
]	%5D
@	%40
&	& amp;

Furthermore, *startTime* and *endTime* are absolute time points of one complete year denoting the time span for which the dynamizer provides dynamic values. The details of the sensor connection are specified using the data type *SensorConnection*. It includes the unique id of the sensor (named as "ICRI_QEOP_0001"), type of the service (SOS version 2.0.0), and links to the sensor description and observations. According to SOS 2.0.0, the sensor observations can be retrieved using the *GetObservation* request. The request can be made using a URL with the specified parameters:

```
http://129.187.38.201:8080/52n-sos-webapp-qeop/service?
service=SOS&version=2.0.0&request=GetObservation&
featureOfInterest=ICRI_QEOP_0001_London&
procedure=ICRI_QEOP_0001&
observedProperty=Outside_Temperature&
temporalFilter=om:phenomenonTime,2016-07-02T09:00:00Z/2016-07-02T09:30:00Z
```

The above mentioned request will return the response of observations encoded according to the OGC O&M standard for the specified parameters.

In the same way, the sensor description and its metadata can be retrieved using the *DescribeSensor* request. The request can be made using a URL, which is as follows:

```
http://129.187.38.201:8080/52n-sos-webapp-qeop/service?
REQUEST=DescribeSensor&SERVICE=SOS&VERSION=2.0.0&
PROCEDURE=ICRI_QEOP_0001&
procedureDescriptionFormat=http://www.opengis.net/sensorML/1.0.1
```

The above mentioned request will return the response of sensor description in the OGC SensorML standard.

Visualization of the scenario

For visualization of this scenario, the 3DCityDB Web-map-client Pro (c.f. section 9.1.5) was used. This web client serves as a user interface to the end users to explore the 3D buildings of the Royal Borough of Greenwich in interactive ways. The thematic information of each building can be visualized simply by clicking on the building. The web client does require the users to log in by a valid Google credential in order to retrieve the building attributes from Google Fusion Table [19]. The client also provides functionalities to query the attributes.

Screenshots

Following are the screenshots taken from the 3DCityDB Web-map-client Pro, which was set up for demonstrating the scenario for integrating sensor data with CityGML data and interacting with real-time sensor observations.

Figures 8 and 9 show different panels of the 3DCityDB Web-map-client Pro, allowing users to visualize and interact with 3D building geometries as well as thematic attributes of the buildings. As shown in figure 9, the GML ID is a unique ID of the building which is based on the TOID provided by Ordnance Survey Mastermap. The sensor-related attributes have been defined based on the dynamizer class. Sensor_ID is a unique ID for a specific sensor. ServiceType shows the type of service being used by the sensor (SOS 2.0 in this case). Further, LinkToSensorML is the request URL for the SOS DescribeSensor operation, which returns the sensor descriptions and metadata encoded in the SensorML format. LinkToObservation is the request URL for the SOS GetObservation operation, which returns the sensor observations encoded in the OGC O&M format. LinkToSensorClient is the URL for the 52° North sensor visualization client, which shows the sensor observations in the form of timeseries graphs as shown in figure 10.

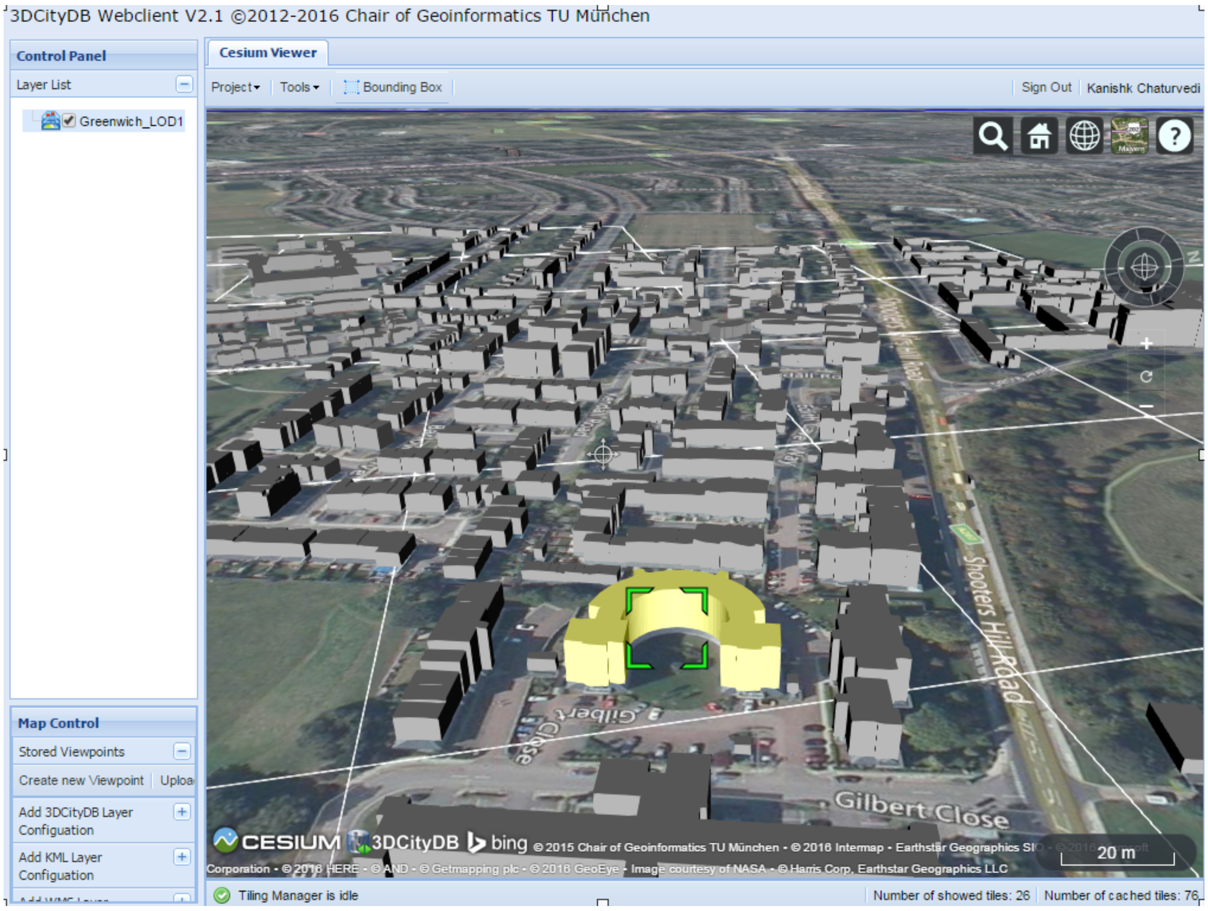
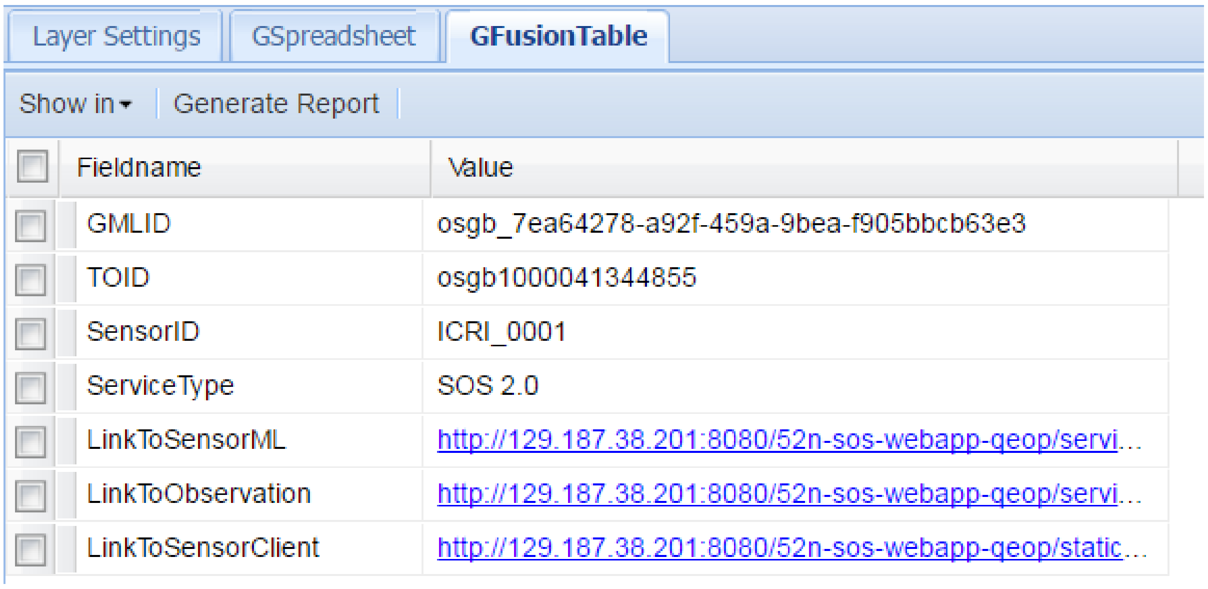


Figure 8. Screenshot 1: Visualization of and interaction with 3D building geometries



Fieldname	Value
GMLID	osgb_7ea64278-a92f-459a-9bea-f905bbcb63e3
TOID	osgb1000041344855
SensorID	ICRI_0001
ServiceType	SOS 2.0
LinkToSensorML	http://129.187.38.201:8080/52n-sos-webapp-qeop/servi...
LinkToObservation	http://129.187.38.201:8080/52n-sos-webapp-qeop/servi...
LinkToSensorClient	http://129.187.38.201:8080/52n-sos-webapp-qeop/static...

Figure 9. Screenshot 2: Thematic attributes of the building including description and links for sensor based services.

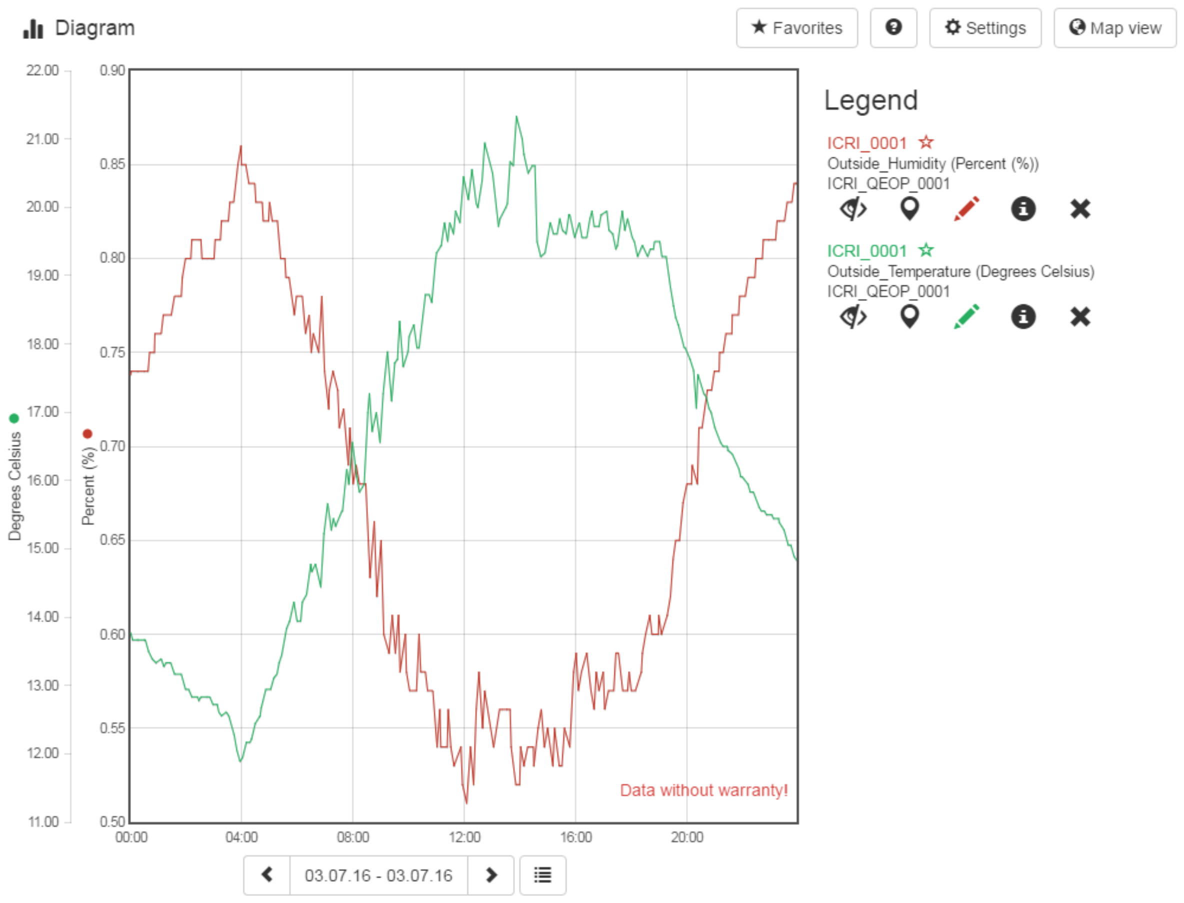


Figure 10. Screenshot 3: Timeseries graph visualization of real-time sensor observations (screenshot taken from 52° North SOS Client). The graph shows the outside temperature and humidity retrieved from the station ICRI_0001 on the 3rd of July 2016 overlain within the same view.

9.2.2. Integrating time-dependent properties with semantic 3D city models

This use case was set up by IGN and virtualcitySYSTEMS GmbH and was based in the commune of Bruz, located 11 km southwest of Rennes in Brittany, France and a part of Rennes Metropole. The main objectives were to provide better services to the citizens and the energy planners by performing a sophisticated solar potential analysis. The use case aimed at answering questions, such as (i) *How many buildings have a solar irradiation of more than a specific value (e.g. 5000 MWh)?*, (ii) *Which buildings are well suited for installing solar panels?*, and *(iii) How does the irradiation for a building varies through the year?*.

Creation of the CityGML datasets

The buildings with wall and roof surfaces were created in accordance with the IGN REF3DNAT specification based on the CityGML standard. The dataset includes approximately 5500 building objects in the LoD2 specification. The LoD2 building objects have thematically-differentiated boundary surfaces (roof and wall surfaces).

Performing the solar potential analysis

The CityGML dataset was further enriched by solar irradiation values computed by the Solar Potential Analysis tool (c.f. section 9.1.6). The simulation tool estimates the solar power from direct, diffuse, and global sunlight irradiation for individual months of the year. Below is a CityGML file generated by the Solar Potential Analysis tool enriching the city objects with the solar irradiation values.

```
<?xml version="1.0" encoding="UTF-8"?>
<core:CityModel xmlns="http://www.citygml.org/citygml/profiles/base/2.0"
 xmlns:core="http://www.opengis.net/citygml/2.0"
 xmlns:bldg="http://www.opengis.net/citygml/building/2.0"
 xmlns:gen="http://www.opengis.net/citygml/generics/2.0"
 xmlns:gml="http://www.opengis.net/gml"
 xmlns:xAL="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0"
 xmlns:dyn="http://www.citygml.org/ade/dynamizer_ade/1.0"
 xmlns:sos="http://www.opengis.net/sos/2.0"
 xmlns:om="http://www.opengis.net/om/2.0"
 xmlns:xlink="http://www.w3.org/1999/xlink"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.opengis.net/citygml/2.0
http://schemas.opengis.net/citygml/2.0/cityGMLBase.xsd
 http://www.opengis.net/citygml/building/2.0
http://schemas.opengis.net/citygml/building/2.0/building.xsd
 http://www.opengis.net/citygml/generics/2.0
http://schemas.opengis.net/citygml/generics/2.0/generics.xsd">
<gml:name>Bruz</gml:name>
<gml:boundedBy>
    <gml:Envelope srsDimension="3"
srsName="urn:ogc:def:crs,crs:EPSG:6.12:3948,crs:EPSG:6.12:5720">
      <gml:lowerCorner>1343348.875000 7208788.500000 12.183561</gml:lowerCorner>
      <gml:upperCorner>1348673.750000 7218523.000000 125.694313</gml:upperCorner>
    </gml:Envelope>
 </gml:boundedBy>
<core:cityObjectMember>
    <bldg:Building gml:id="BU_66ac0df9-cba5-40ac-a935-742917fb4e98">
        ...
        <bldg:boundedBy>
            <bldg:WallSurface gml:id="UUID_23d7f89d-e222-4ebb-980e-affac4721f77">
                ....
                <gen:doubleAttribute name="directRadMonth_01">
                    <gen:value>146.607727</gen:value>
                </gen:doubleAttribute>
                <gen:doubleAttribute name="directRadMonth_02">
                    <gen:value>231.137695</gen:value>
                </gen:doubleAttribute>
                <gen:doubleAttribute name="directRadMonth_03">
                    <gen:value>333.088562</gen:value>
                </gen:doubleAttribute>
                ...

            </bldg:WallSurface>
       </bldg:boundedBy>
    </bldg:Building>
 </core:cityObjectMember>
</core:CityModel>
```

As shown, the solar power from direct, diffuse, and global sunlight irradiation were estimated for individual months and stored as individual generic attributes for the building surfaces. However, these monthly solar irradiation values can now be represented within the CityGML documents using the dynamizers, which can be used for visualizations and simulations. An example illustration below shows the CityGML dynamizers representing these monthly solar irradiation values according to the OGC TimeseriesML 1.0 standard.

```
<?xml version="1.0" encoding="UTF-8"?>
<core:CityModel xmlns="http://www.citygml.org/citygml/profiles/base/2.0"
 xmlns:core="http://www.opengis.net/citygml/2.0"
 xmlns:bldg="http://www.opengis.net/citygml/building/2.0"
 xmlns:gen="http://www.opengis.net/citygml/generics/2.0"
 xmlns:gml="http://www.opengis.net/gml"
 xmlns:xAL="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0"
 xmlns:dyn="http://www.citygml.org/ade/dynamizer_ade/1.0"
 xmlns:sos="http://www.opengis.net/sos/2.0"
 xmlns:om="http://www.opengis.net/om/2.0"
 xmlns:xlink="http://www.w3.org/1999/xlink"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.opengis.net/citygml/2.0
http://schemas.opengis.net/citygml/2.0/cityGMLBase.xsd
 http://www.opengis.net/citygml/building/2.0
http://schemas.opengis.net/citygml/building/2.0/building.xsd
 http://www.opengis.net/citygml/generics/2.0
http://schemas.opengis.net/citygml/generics/2.0/generics.xsd
 http://www.citygml.org/ade/dynamizer_ade/1.0 CityGML-DynamizerADE.xsd">
<gml:name>Bruz</gml:name>
<gml:boundedBy>
    <gml:Envelope srsDimension="3"
srsName="urn:ogc:def:crs,crs:EPSG:6.12:3948,crs:EPSG:6.12:5720">
      <gml:lowerCorner>1343348.875000 7208788.500000 12.183561</gml:lowerCorner>
      <gml:upperCorner>1348673.750000 7218523.000000 125.694313</gml:upperCorner>
    </gml:Envelope>
 </gml:boundedBy>
<core:cityObjectMember>
    <bldg:Building gml:id="BU_66ac0df9-cba5-40ac-a935-742917fb4e98">
        ...
        <bldg:boundedBy>
            <bldg:WallSurface gml:id="UUID_23d7f89d-e222-4ebb-980e-affac4721f77">
                ....
                <gen:doubleAttribute name="directRadMonth">
                    <gen:value>146.607727</gen:value>
                </gen:doubleAttribute>
                ...
            </bldg:WallSurface>
       </bldg:boundedBy>
    </bldg:Building>
 </core:cityObjectMember>
 <core:cityObjectMember>
    <dyn:Dynamizer gml:id="UUID_23d7f89d-e222-4ebb-980e-
affac4721f77_directRad_Dynamizer">
```

```
        <dyn:attributeRef>
            <!-- Single line XPath Expression -->
            //bldg:WallSurface%5B%40gml:id='UUID_33f88cd5-89cf-44ea-a6eb-
050524157035'%5D
            /doubleAttribute%5B%40name='directRadMonth'%5D
            /gen:value
        </dyn:attributeRef>
        <dyn:startTime frame="#ISO-8601">
            2015-01-01T00:00:00Z
        </dyn:startTime>
        <dyn:endTime frame="#ISO-8601">
            2016-01-01T00:00:00Z
        </dyn:endTime>
        <dyn:dynamicData>
            <dyn:AtomicTimeseries>
                <dyn:dynamicDataTVP>
                    <tsml:TimeseriesTVP gml:id="TS1">
                        <tsml:point>
                            <tsml:MeasurementTVP>
                                <tsml:time>2015-01</tsml:time>
                                <tsml:value>146.607727</tsml:value>
                            </tsml:MeasurementTVP>
                        </tsml:point>
                        <tsml:point>
                            <tsml:MeasurementTVP>
                                <tsml:time>2015-02</tsml:time>
                                <tsml:value>231.137695</tsml:value>
                            </tsml:MeasurementTVP>
                        </tsml:point>
                        <tsml:point>
                            <tsml:MeasurementTVP>
                                <tsml:time>2015-03</tsml:time>
                                <tsml:value>333.088562</tsml:value>
                            </tsml:MeasurementTVP>
                        </tsml:point>
                        ....
                    </tsml:TimeseriesTVP>
                </dyn:dynamicDataTVP>
            </dyn:AtomicTimeseries>
        </dyn:dynamicData>
    </dyn:Dynamizer>
 </core:cityObjectMember>
</core:CityModel>
```

As shown in the above example, the wall surface of the CityGML building object contains only one generic attribute *directRadMonth* for handling monthly direct irradiation values. This generic attribute is referred by the Dynamizer feature using the XPath expression:

```
//bldg:WallSurface[@gml:id='UUID_33f88cd5-89cf-44ea-a6eb-050524157035']
/doubleAttribute[@name='directRadMonth']
/gen:value
```

Further, the Dynamizer contains the AtomicTimeseries, which represents the time/value pairs of monthly direct irradiation values for the wall surface.

Visualization of the scenario

For visualization, a virtualcityMap based web application [20] was developed and set up by virtualcitySYSTEMS GmbH and was used for the demonstrations. It allows exploring the 3D buildings and solar irradiation values in interactive ways.

Screenshots

Following are the screenshots taken from the virtualcityMAP web application. Figures 11 and 12 show solar irradiation values of the building roof and wall surfaces defined according to specific color ramps. For comparing the solar irradiation values, figure 11 shows the values from the month February and figure 12 shows the values from the month August. Figure 13 illustrates all the values of the solar potential results encoded in a single timeseries graph. Such timeseries graphs can be retrieved and used for further simulations with the help of the dynamizers. Figure 14 shows the point clouds in different colors which were generated from the sampling points for each building surface for the simulation.

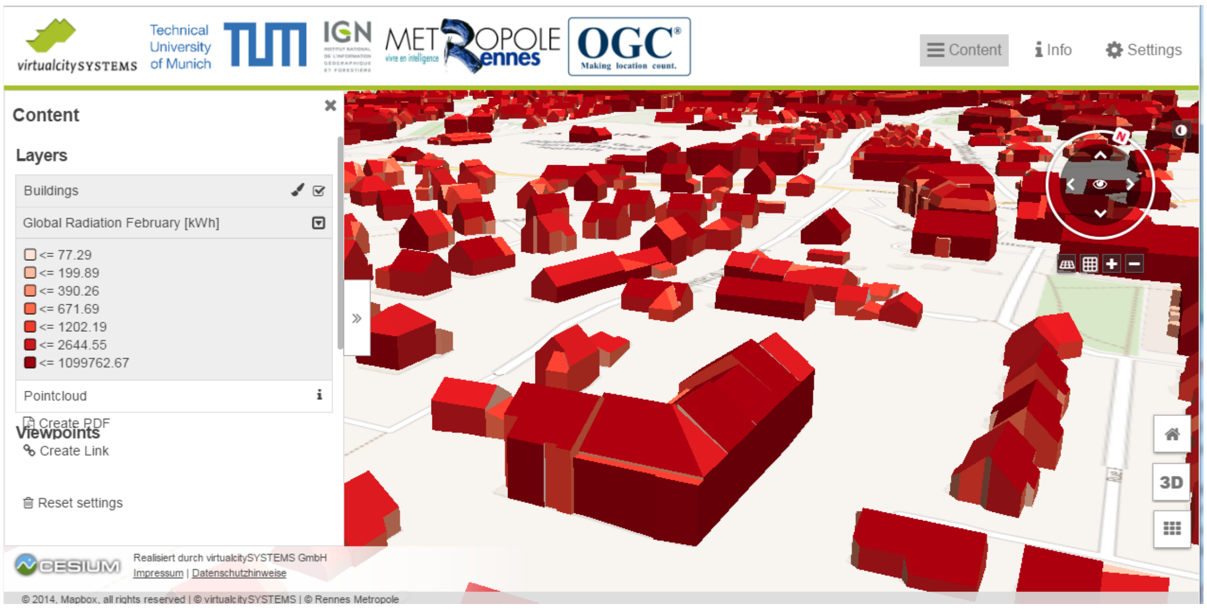

virtualcitySYSTEMS
Technical University of Munich
IGN
METROPOLE Rennes
OGC Making location count.
Content
Info
Settings
Content
Layers
Buildings
Global Radiation February [kWh]
<= 77.29
<= 199.89
<= 390.26
<= 671.69
<= 1202.19
<= 2644.55
<= 1099762.67
Pointcloud
Create PDF
Viewpoints
Create Link
Reset settings
3D
CESIUM
Realisiert durch virtualcitySYSTEMS GmbH
Impressum | Datenschutzhinweise
© 2014, Mapbox, all rights reserved | © virtualcitySYSTEMS | © Rennes Metropole

Figure 11. Screenshot 1: Solar irradiation values for the month February

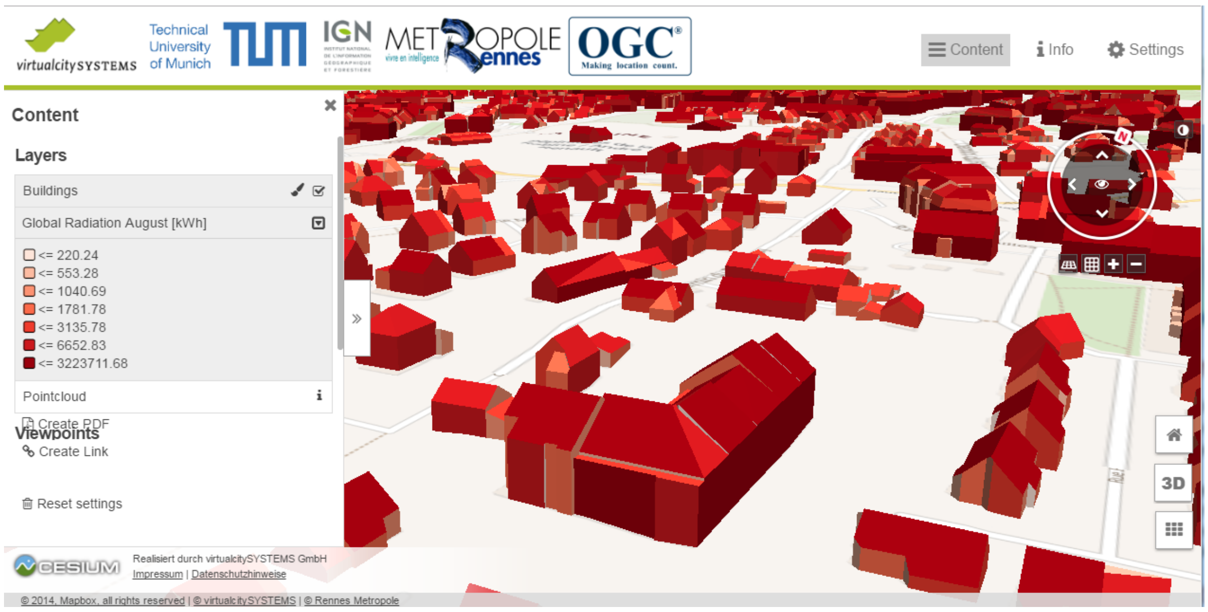
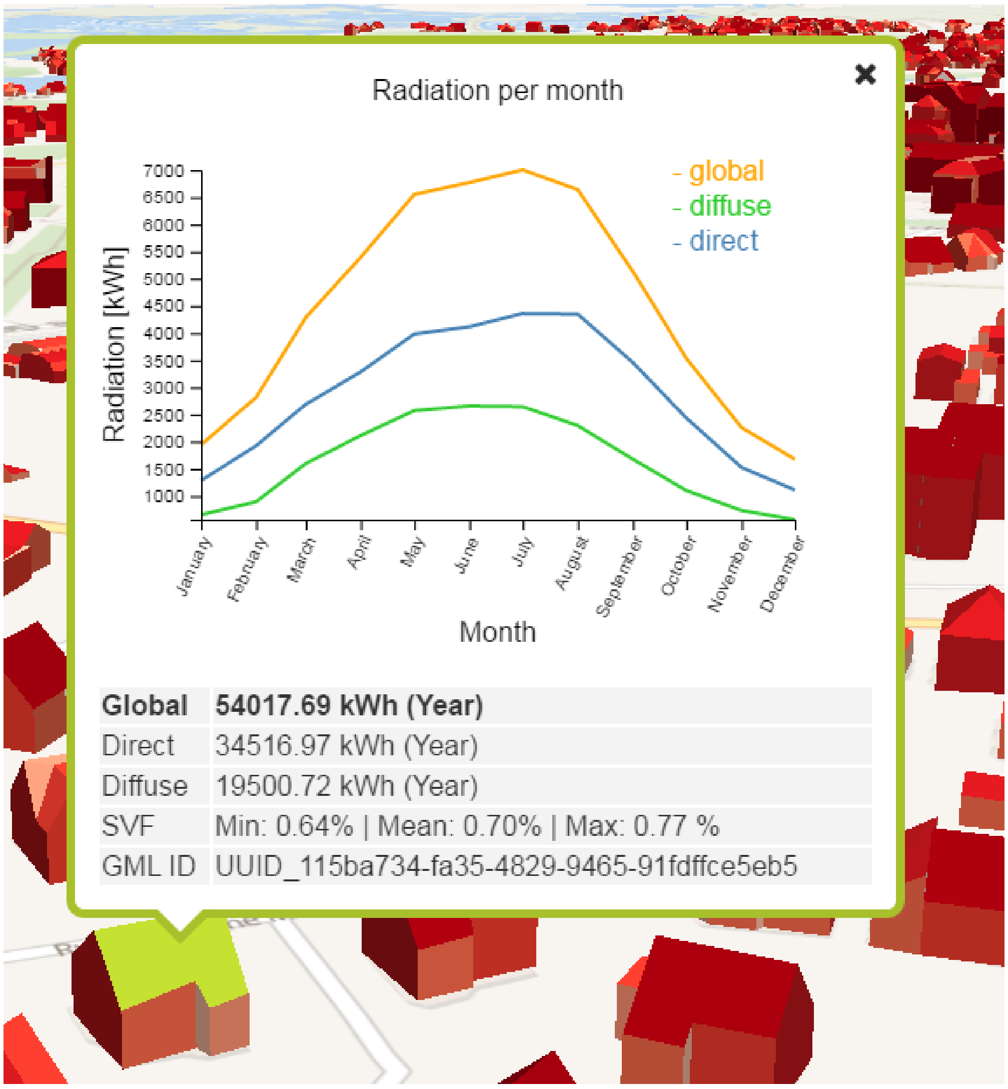

virtualcitySYSTEMS
Technical University of Munich
Content
Info
Settings
Content
Layers
Buildings
Global Radiation August [kWh]
<= 220.24
<= 553.28
<= 1040.69
<= 1781.78
<= 3135.78
<= 6652.83
<= 3223711.68
Pointcloud
Create PDF
Viewpoints
Create Link
Reset settings
3D
Realisiert durch virtualcitySYSTEMS GmbH
Impressum | Datenschutzhinweise
© 2014, Mapbox, all rights reserved | © virtualcitySYSTEMS | © Rennes Metropole

Figure 12. Screenshot 2: Solar irradiation values for the month August



Global	**54017.69 kWh (Year)**
Direct	34516.97 kWh (Year)
Diffuse	19500.72 kWh (Year)
SVF	Min: 0.64% | Mean: 0.70% | Max: 0.77 %
GML ID	UUID_115ba734-fa35-4829-9465-91fdffce5eb5

Figure 13. Screenshot 3: All the values of solar potential results encoded in a single timeseries graph

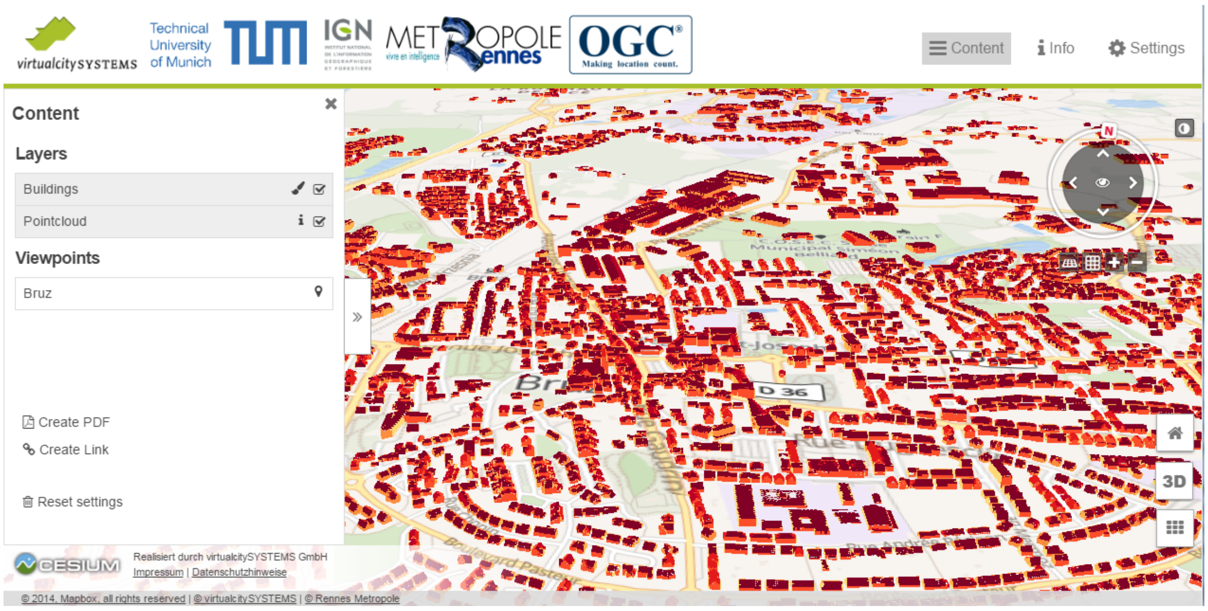

virtualcitySYSTEMS
Technical University of Munich
Content
Info
Settings
Content
Layers
Buildings
Pointcloud
Viewpoints
Bruz
Create PDF
Create Link
Reset settings
Realisiert durch virtualcitySYSTEMS GmbH
Impressum | Datenschutzhinweise
© 2014, Mapbox, all rights reserved | © virtualcitySYSTEMS | © Rennes Metropole
3D

Figure 14. Screenshot 4: Point clouds visualization

Chapter 10. Conclusions and future work

10.1. Conclusions

The Future City Pilot Phase 1 successfully demonstrates how the use of international standards such as CityGML and IFC together can provide stakeholders with information, knowledge, and insight which enhances financial, environmental, and social outcomes for citizens living in cities. During the pilot, three scenarios were set up based on real-world requirements put forward by the pilot sponsors and their solutions were developed and demonstrated by the pilot participants.

This Engineering Report (ER) focuses on the scenario requiring the support of real-time sensor observations and other time-dependent properties within the CityGML standard. It highlights the conceptual and implementation details of a new concept 'Dynamizer', which has been modeled as an Application Domain Extension (ADE) of the CityGML standard. Dynamizers allow extending static 3D city models by supporting variations of individual feature properties and associations over time. They provide a data structure to represent dynamic values in different and generic ways. Such dynamic values may be given as 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 the thematic, geometry, and appearance properties of the city objects.

As a part of the pilot, Dynamizers were successfully modeled and implemented for scenarios which allowed for the integration of sensors as well as other time-dependent properties within the CityGML standard. This report explains the steps taken for implementing the Dynamizer ADE. The conceptual UML model and its components were developed and explained in detail. The XML Schema definition file was derived from the UML model. Based on the XML schema definition file, the valid CityGML instance documents were created and used within the pilot and demonstrated with the help of the visualization clients.

10.2. Future Work

10.2.1. CityGML 3.0

The dynamizer concept has been implemented as an 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). However, this concept is intended to become a 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 UK Ordnance Survey Mastermap or the German cadaster standard ALKIS.

10.2.2. Supporting different time-dependent properties with Dynamizers

As mentioned in Chapter 6, the dynamizers are capable of supporting time-dependent variations of the spatial, thematic, as well as appearance properties of a city object. Within FCP1, the dynamizers were successfully implemented to support variations in the thematic properties of the objects, such as solar irradiation values of a roof of a building. However, in future, the dynamizers will be implemented to support such variations in the spatial and appearance properties, for example, variations in flood depth values of water bodies over a period of time, moving objects in case of traffic simulations, and representation of changes in properties using different textures.

10.2.3. Dynamizer ADE support in databases

The 3D City Database (3DCityDB) is a powerful tool for storing, representing, and managing large CityGML datasets on top of a standard spatial relational database such as Oracle Spatial and PostgreSQL. However, the current version of the 3DCityDB does not support ADEs, due to which the dynamizer ADE features could not be stored within the database in the FCP1.

There is ongoing work at the Chair of Geoinformatics, Technical University of Munich for providing an automated way for dynamically extending the 3DCityDB to support storage and management of the CityGML models with ADEs. The initial work has already been successfully implemented, tested, and evaluated based on a number of different CityGML ADEs like Energy ADE, UtilityNetwork ADE, and Dynamizer ADE [21]. However, currently the timeseries values (such as Atomic Timeseries) are stored as Large Objects (CLOB) within the database, which is not suitable for further analysis. The Dynamizer ADE allows representing timeseries data in standardized ways (e.g., according to OGC TimeseriesML 1.0 and OGC O&M). It will be investigated how these time-dependent values can be more efficiently managed and stored within the 3DCityDB.

10.2.4. Dynamizer support in visualization clients

Within the pilot, different visualization clients were used and modified to represent dynamic data. For representing real-time sensor observations, the 52° North SOS Client was used (as demonstrated in section 9.2.1). Similarly, as shown in section 9.2.2, the virtualcityMap was set up by virtualcitySYSTEMS GmbH for assessing and visualizing the estimations of solar energy production for the roofs and facades of buildings in different ways (e.g., visualization of solar irradiation values using different color ramps on the facades of building surfaces as well as using timeseries graphs).

There is an ongoing work at the Chair of Geoinformatcs, Technical University of Munich for investigating how dynamizers can directly be interpreted by such visualization clients.

10.2.5. Other sensors and IoT standards

There are many different types of sensors and IoT devices, measuring different qualities/quantities. They may belong to different stakeholders with different rights and interests. They may also belong to different platforms, which can be open or proprietary. In order to integrate diverse sensors and IoT devices with city information models within one operational framework, interoperability plays an important role in ensuring different components from different vendors can work together. The OGC already provides the SWE standard suite for realizing interoperable sensor web infrastructures. Within the OGC SWE standards suite, sensor descriptions are encoded in the SensorML format and sensor observations in the O&M format. The web services such as Sensor Observation Service and SensorThings API allow retrieval of sensor descriptions and observations using different requests.

Within the FCP1, the Sensor Observation Service was successfully set up, used, and integrated with

CityGML using dynamizers. In the future, it will be interesting to extend support to other sensor and IoT based APIs like the OGC Sensor Things API[OGC 15-078r6] , OpenSensorHub [22], and FIWARE [23].

Acknowledgements

The work has been partly funded by the OGC and the sponsors of the FCP1. We thank the OGC and all the FCP1 sponsors for supporting this work. For the use cases, the CityGML datasets were provided by Ordnance Survey Great Britain and Institut National de l’Information Géographique et Forestière (IGN), France. We thank virtualcitySYSTEMS GmbH, Germany for setting up the virtualcityMAP web application for one of the demonstrations. We also acknowledge the Climate-KIC of the European Institute of Innovation and Technology (EIT). The real-time sensor observations used in the FCP1 demonstration were retrieved from the project Smart Sustainable Districts funded by the Climate-KIC.

Appendix A: Flood Inundation Modeling with 3D city models

Human, natural, and physical systems interact in space and time and digital systems in cities will become increasingly diverse and numerous, with many actors. Cities therefore need a neutral, open platform, and standards for communicating spatial and temporal data. The Open Geospatial Consortium (OGC®) Future Cities Pilot 1 (FCP1) shows how cities can begin to adapt. The goal of this pilot project is to help cities around the world benefit from modern standards for geospatial technologies. The European-based pilot demonstrates and enhances the ability of cities to use different interoperable geospatial technologies to deliver better quality of life and civic initiatives in order to improve the resilience of society.

With the theme of climate change, the resilience to the risks of flooding is becoming more and more imminent. The management of the urban or peri-urban territory is characterized by complexity and requires the collaboration of all the actors to make the right decisions when managing the risks of flooding based on data of various origins created using different software. This risk management requires setting up tools and methods for the proper coordination of the actions of the various players. Flood mitigation and crisis management, as well as recovery, involve communities at different levels. Collaboration between government departments, local authorities and public and private actors (citizens, businesses, etc.) is therefore necessary to protect people and properties.

Cities need to take into account urban planning and management, the implementation of flood levees and protection structures, and address the ecological functionality of rivers and urban wetland areas. These efforts must be complemented by numerical model simulations of floods across 2-D floodplains.

For a more advanced analysis and visualization of the numerical simulations as well as to support the actors in the management of the risks of flooding and mitigation, we proposed a flood inundation scenario for the cities of Rennes and Greenwich, in collaboration with flood experts at Rennes Métropole and the University of Bristol, respectively. The purpose of the “inundation scenario” of the FCP1 pilot project is to transform the mapping of the results of flood modeling into CityGML format in order to add such simulations to existing 3D City Model databases (figure 15). This will enable cities to manage urban and environmental data more efficiently, interoperably and sustainably.

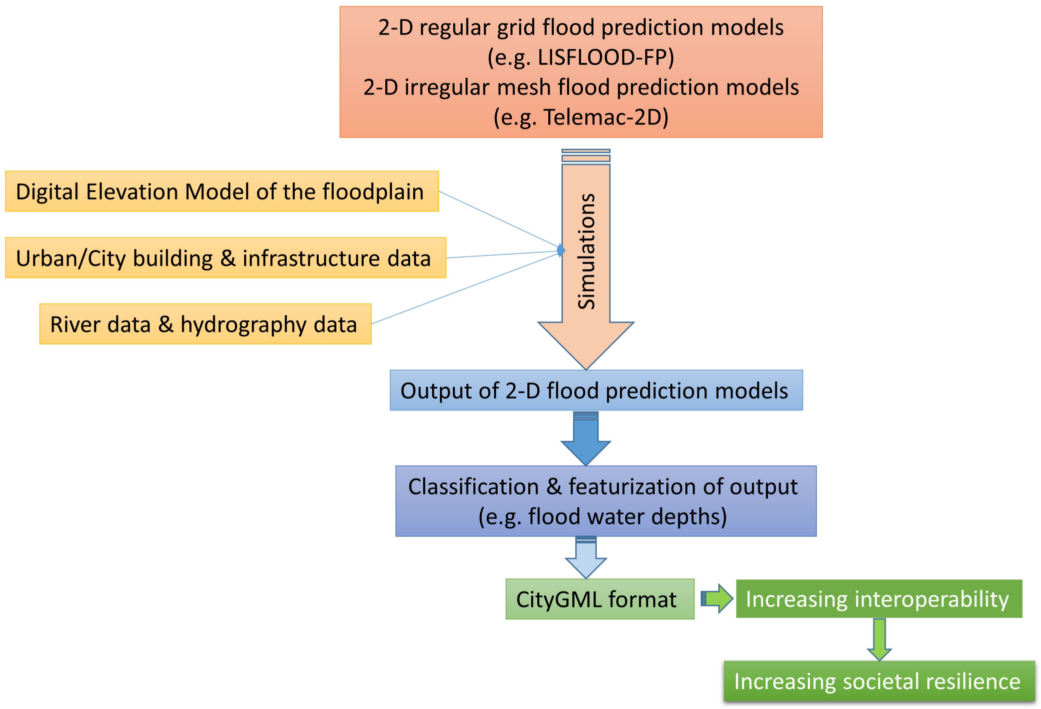

2-D regular grid flood prediction models (e.g. LISFLOOD-FP)
2-D irregular mesh flood prediction models (e.g. Telemac-2D)
Digital Elevation Model of the floodplain
Urban/City building & infrastructure data
River data & hydrography data
Simulations
Output of 2-D flood prediction models
Classification & featurization of output (e.g. flood water depths)
CityGML format
Increasing interoperability
Increasing societal resilience

Figure 15. Workflow schematic of the FCP1 pilot demonstration of the "inundation scenario" .

For the demonstration of the Rennes flood scenario in CityGML, we used the TELEMAC-2D software which solves the Saint-Venant equations in two dimensions [24]. Typically, results from this state-of-the-art model are, at each point of the resolution mesh, the height of water and the mean velocity in the vertical, and are commonly presented in a Geographic Information System (GIS) as points X, Y, and Z. For Rennes, we used an existing simulation of a flood of a return time of 1:100 years (provided by Artelia Engineering). The transformation from "traditional" flood model outputs to CityGML was achieved by first creating a triangular irregular network (TIN) with the X, Y and Z points to produce a high-resolution raster.

For the Greenwich "scenario", we used an existing 1:200-year flood return period simulated by the 2-D LISFLOOD-FP flood model of the University of Bristol [25]. This model solves for the hydrodynamic forcing terms of the full Saint-Venant equations, except for convection, on a regular grid and decoupled on the computational faces of the water flux. For the Greenwich use case, this model was employed to simulate an "undefended" flood scenario of the River Thames. Given that for this regular-grid model, the typical outputs are provided in the form of an ASCII raster grid, transformation to a TIN was not necessary.

However, since the CityGML format requires input data as a "feature," the raster of flood heights for both scenarios was first converted into a vector using the classification scheme as shown in figure 16 and figure 17.

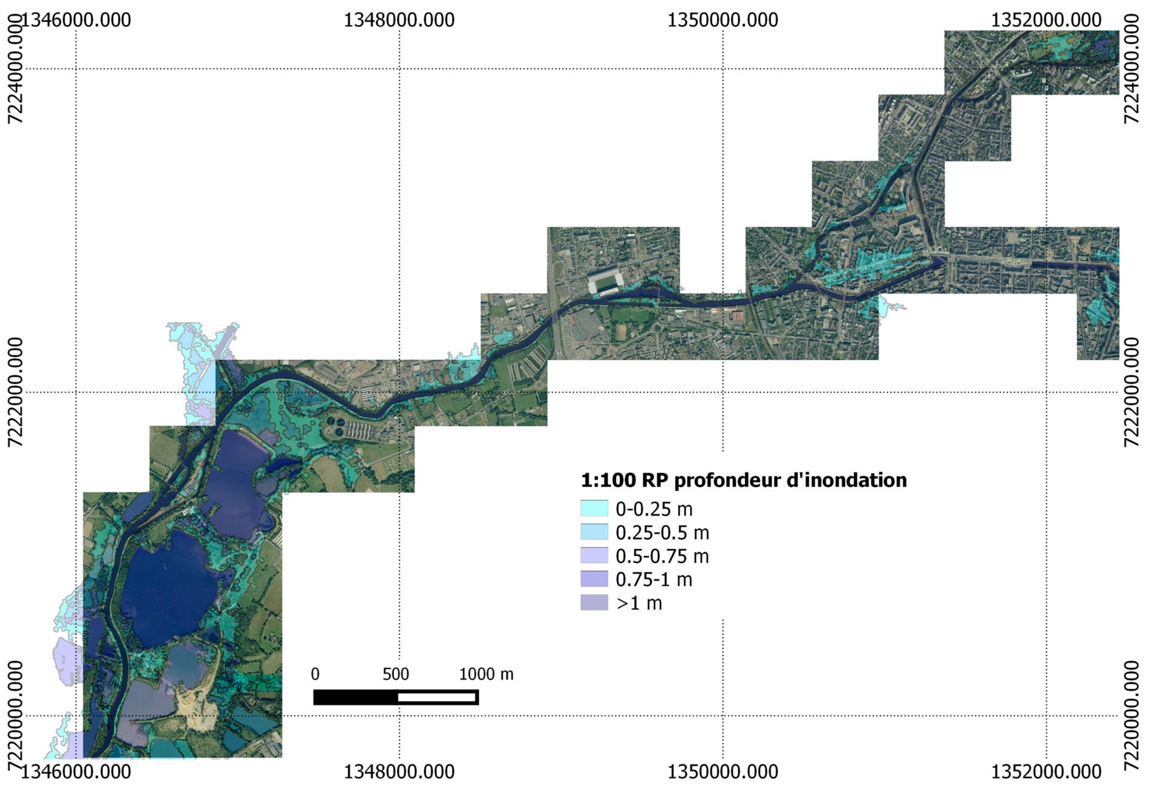


Figure 16. Flood depth classification of a "typical" GIS vector format (before transformation to CityGML) for the Rennes scenario.

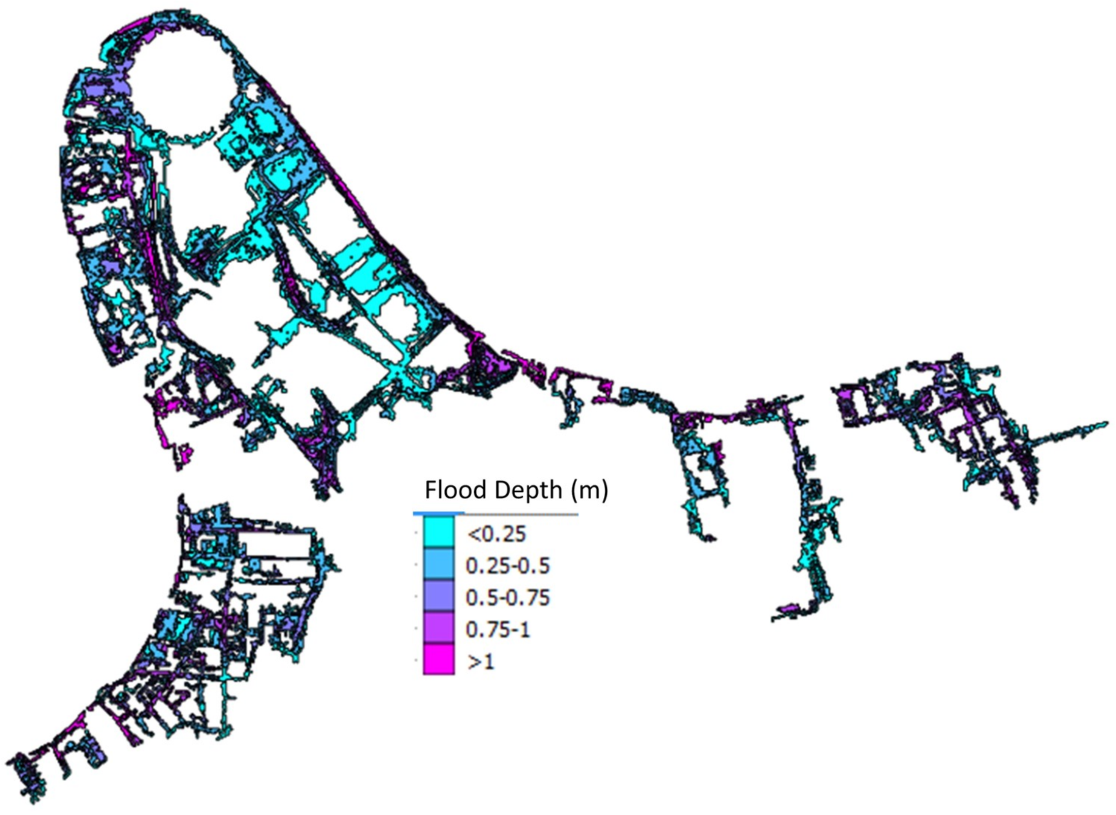


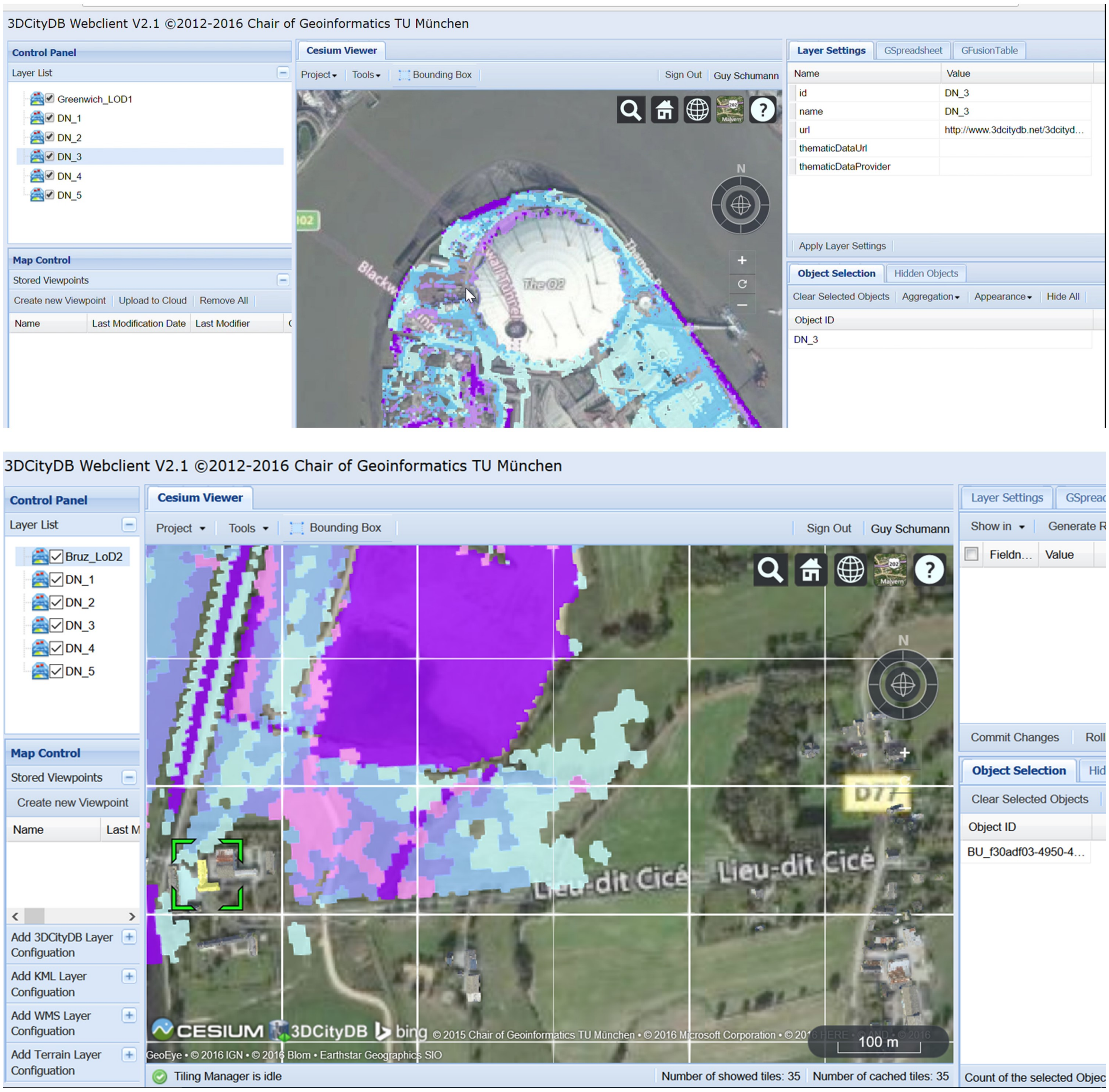
Figure 17. Flood depth classification of a "typical" GIS vector format (before transformation to CityGML) for the Greenwich scenario.

Below are the specifications of a possible/suggested CityGML encoding for flood depths (provided by TUM) and figure 18 shows the application of both the Rennes and Greenwich inundation scenarios in the interactive 3D City Model of the TUM.

CityGML formatting for one polygon patch sample of the Rennes “flood scenario”

```
<?xml version="1.0" encoding="UTF-8"?>
<core:CityModel xmlns:app="http://www.opengis.net/citygml/appearance/2.0"
xmlns:luse="http://www.opengis.net/citygml/landuse/2.0"
xmlns:wtr="http://www.opengis.net/citygml/waterbody/2.0"
xmlns:xAL="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0"
xmlns:core="http://www.opengis.net/citygml/2.0"
xmlns:smil20="http://www.w3.org/2001/SMIL20/"
xmlns:grp="http://www.opengis.net/citygml/cityobjectgroup/2.0"
xmlns:bldg="http://www.opengis.net/citygml/building/2.0"
xmlns:dem="http://www.opengis.net/citygml/relief/2.0"
xmlns:gen="http://www.opengis.net/citygml/generics/2.0"
xmlns:tex="http://www.opengis.net/citygml/texturedsurface/2.0"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:tun="http://www.opengis.net/citygml/tunnel/2.0"
xmlns:frn="http://www.opengis.net/citygml/cityfurniture/2.0"
xmlns:tran="http://www.opengis.net/citygml/transportation/2.0"
xmlns:gml="http://www.opengis.net/gml"
xmlns:bridge="http://www.opengis.net/citygml/bridge/2.0"
xmlns:pbase="http://www.opengis.net/citygml/profiles/base/2.0"
xmlns:smil20lang="http://www.w3.org/2001/SMIL20/Language"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:veg="http://www.opengis.net/citygml/vegetation/2.0"
xmlns:sch="http://www.ascc.net/xml/schematron">
    <gml:boundedBy>
        <gml:Envelope srsDimension="3"
srsName="urn:ogc:def:crs,crs:EPSG:6.12:3948,crs:EPSG:6.12:5720">
            <gml:lowerCorner>1344064.2188 7216529.4531 0.125</gml:lowerCorner>
            <gml:upperCorner>1357029.2188 7227149.4531 1.125</gml:upperCorner>
        </gml:Envelope>
    </gml:boundedBy>
    <core:cityObjectMember>
        <wtr:WaterBody gml:id="DN_1">
            <gen:doubleAttribute name="Height">
                <gen:value>0.125</gen:value>
            </gen:doubleAttribute>
            <wtr:lod1MultiSurface>
                <gml:MultiSurface srsDimension="3">
                    <gml:surfaceMember>
                        <gml:Polygon>
                            <gml:exterior>
                                <gml:LinearRing>
                                    <gml:posList>1350664.2188 7223059.4531 0.125
1350699.2188 7223059.4531 0.125 1350699.2188 7223054.4531 0.125 1350724.2188
7223054.4531 0.125 1350724.2188 7223049.4531 0.125 1350734.2188 7223049.4531 0.125
1350734.2188 7223059.4531 0.125 1350739.2188 7223059.4531 0.125 1350739.2188
```

```
7223054.4531 0.125 1350744.2188 7223054.4531 0.125 1350744.2188 7223044.4531 0.125
1350749.2188 7223044.4531 0.125 1350749.2188 7223039.4531 0.125 1350744.2188
7223039.4531 0.125 1350734.2188 7223039.4531 0.125 1350734.2188 7223034.4531 0.125
1350729.2188 7223034.4531 0.125 1350714.2188 7223034.4531 0.125 1350714.2188
7223029.4531 0.125 1350709.2188 7223029.4531 0.125 1350694.2188 7223029.4531 0.125
1350694.2188 7223024.4531 0.125 1350689.2188 7223024.4531 0.125 1350679.2188
7223024.4531 0.125 1350679.2188 7223019.4531 0.125 1350674.2188 7223019.4531 0.125
1350669.2188 7223019.4531 0.125 1350669.2188 7223014.4531 0.125 1350664.2188
7223014.4531 0.125 1350664.2188 7223009.4531 0.125 1350659.2188 7223009.4531 0.125
1350654.2188 7223009.4531 0.125 1350654.2188 7223004.4531 0.125 1350649.2188
7223004.4531 0.125 1350649.2188 7222999.4531 0.125 1350644.2188 7222999.4531 0.125
1350639.2188 7222999.4531 0.125 1350639.2188 7222994.4531 0.125 1350634.2188
7222994.4531 0.125 1350634.2188 7222989.4531 0.125 1350629.2188 7222989.4531 0.125
1350629.2188 7222984.4531 0.125 1350624.2188 7222984.4531 0.125 1350624.2188
7222994.4531 0.125 1350629.2188 7222994.4531 0.125 1350629.2188 7222999.4531 0.125
1350624.2188 7222999.4531 0.125 1350624.2188 7223004.4531 0.125 1350629.2188
7223004.4531 0.125 1350634.2188 7223004.4531 0.125 1350634.2188 7223029.4531 0.125
1350639.2188 7223029.4531 0.125 1350639.2188 7223039.4531 0.125 1350644.2188
7223039.4531 0.125 1350644.2188 7223044.4531 0.125 1350649.2188 7223044.4531 0.125
1350649.2188 7223039.4531 0.125 1350654.2188 7223039.4531 0.125 1350654.2188
7223044.4531 0.125 1350659.2188 7223044.4531 0.125 1350659.2188 7223054.4531 0.125
1350664.2188 7223054.4531 0.125 1350664.2188 7223059.4531 0.125</gml:posList>
                                </gml:LinearRing>
                            </gml:exterior>
                            <gml:interior>
                        </gml:Polygon>
                    </gml:surfaceMember>
                </gml:MultiSurface>
            </wtr:lod1MultiSurface>
        </wtr:WaterBody>
    </core:cityObjectMember>
</core:CityModel>
```



Name	Value
id	DN_3
name	DN_3
url	http://www.3dcitydb.net/3dcityd…
thematicDataUrl	
thematicDataProvider	

Object ID
DN_3

Object ID
BU_f30adf03-4950-4…

Figure 18. Illustration of CityGML-based flood depth simulations in a 3D City Model environment for Greenwich (top) and Rennes (bottom).

Furthermore, in order to present and process simulations of the TELEMAC-2D in CityGML for the Rennes project (as shown in figure 19), Rennes Métropole and IGN have made available several types of geospatial data, including a high-resolution numerical terrain model, river bathymetric data, details of 3-D infrastructures (flood levees, residential, industrial and commercial buildings), and aerial photographs of the city. All of these geospatial data comply with OGC standards and protocols, thus allowing more efficient interoperability.

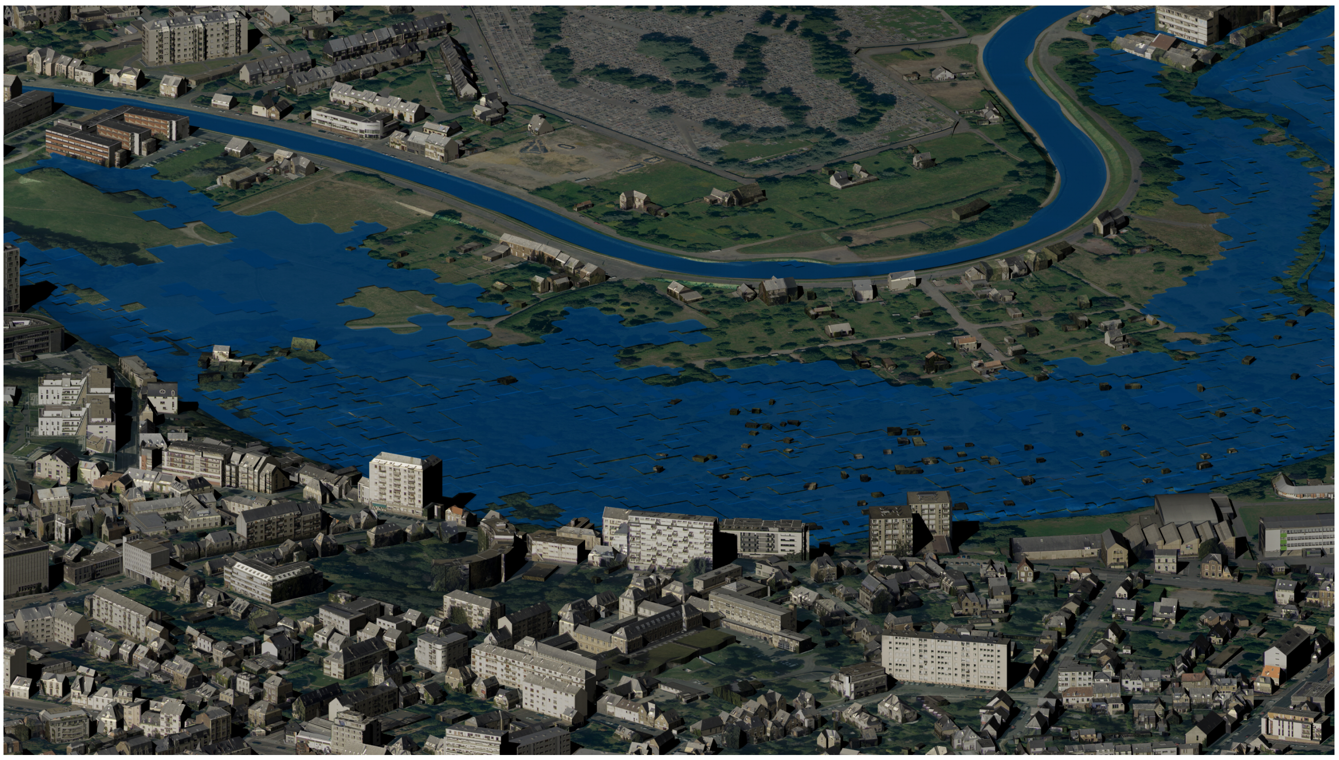


Figure 19. Visualization of flooding in Rennes using the proposed CityGML encoding draped over 3D terrain and building data (provided by Rennes Métropole/IGN).

Finally, figure 20 illustrates a possible wiring diagram scenario using OGC standards of how the CityGML encoding within a 3D City Model could benefit society and more specifically flood disaster response assistance, particularly in urban areas where most assets at risk of flooding are located.

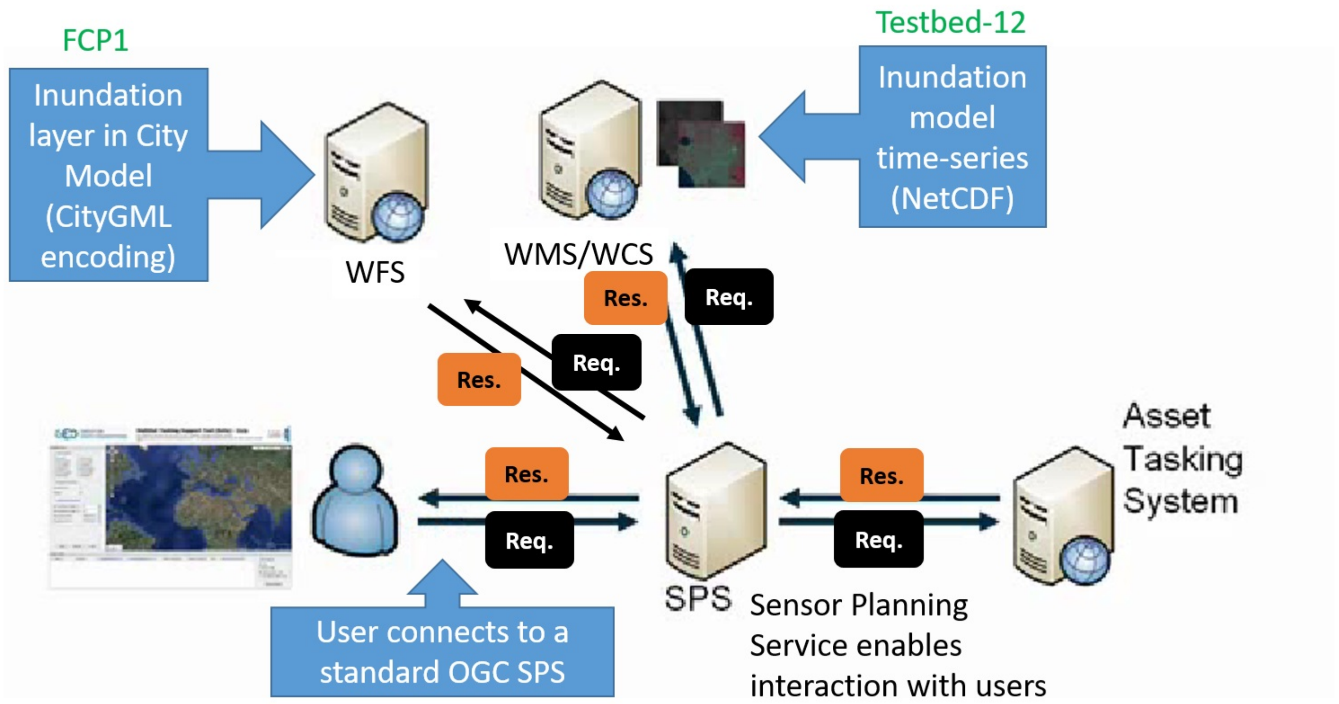

FCP1
Inundation layer in City Model (CityGML encoding)
Testbed-12
Inundation model time-series (NetCDF)
WFS
WMS/WCS
Res.
Req.
Res.
Req.
Asset Tasking System
Res.
Req.
Res.
Req.
SPS
Sensor Planning Service enables interaction with users
User connects to a standard OGC SPS

Figure 20. OGC-based testbed wiring diagram of the usage of a possible "flood scenario" in CityGML. Note that this example specifically aims to link FCP1 to Testbed-12 scenarios.

Appendix B: Revision History

Table 3. Revision History

Date	Release	Editor	Primary clauses modified	Descriptions
June 8, 2016	0.1	K. Chaturvedi	all	initial version
August 23, 2016	0.2	K. Chaturvedi	all	draft engineering report
January 20, 2017	0.3	K. Chaturvedi	all	draft engineering report capturing key results
March 20, 2017	0.4	K. Chaturvedi	all	Addition of appendix to cover flood inundation scenario, additions in Future Work section
May 16, 2017	0.5	K. Chaturvedi	all	Improvements suggested by the reviewers

Appendix C: Bibliography

[1] IFC: Industry Foundation Classes (IFC) Overview Summary, http://www.buildingsmart-tech.org/specifications/ifc-overview/ifc-overview-summary.

[2] Chaturvedi, K., Smyth, C.S., Gesquière, G., Kutzner, T., Kolbe, T.H.: Managing Versions and History Within Semantic 3D City Models for the Next Generation of CityGML. In: Abdul-Rahman, A. (ed.) Advances in 3D Geoinformation, pp. 191–206. Springer International Publishing, Cham (2017), https://mediatum.ub.tum.de/node?id=1276238.

[3] Chaturvedi, K.,Kolbe, T.H.: Integrating Dynamic data and Sensors with Semantic 3D City Models in the context of Smart Cities. In: ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. IV-2-W1vol. , pp. 31–38. Copernicus GmbH (2016), https://mediatum.ub.tum.de/node?id=1276240.

[4] Chaturvedi, K.,Kolbe, T.H.: Dynamizers - Modeling and Implementing Dynamic Properties for Semantic 3D City Models. In: Filip Biljecki, Vincent Tourre (eds.) Eurographics Workshop on Urban Data Modelling and Visualisation. The Eurographics Association (2015), https://diglib.eg.org/handle/10.2312/udmv20151348.

[5] W3C Recommendation: XML Path Language (XPath) 2.0 (Second Edition), http://www.w3.org/TR/xpath20.

[6] Kutzner, T.: Geospatial Data Modelling and Model-driven Transformation of Geospatial Data based on UML Profiles, pp. 75-111, PhD Thesis, Ingenieurfakultät Bau Geo Umwelt, Technische Universität München (2016), https://mediatum.ub.tum.de/node?id=1341432.

[7] EA: Enterprise Architect - UML Design Tools and UML CASE tools for software development, http://www.sparxsystems.com/products/ea.

[8] ShapeChange: ShapeChange - Processing application schemas for geographic information, http://shapechange.net.

[9] FME: Safe Software FME Integrate Data, Applications, Web Services, http://www.safe.com.

[10] citygml4j: citygml4j - The Open Source Java API for CityGML, https://github.com/citygml4j.

[11] 3DCityDB: 3D City Database (3DCityDB) Homepage, http://www.3dcitydb.org/3dcitydb/3dcitydbhomepage.

[12] Chaturvedi, K., Yao, Z., Kolbe, T.H.: Web-based Exploration of and Interaction with Large and Deeply Structured Semantic 3D City Models using HTML5 and WebGL. In: Kersten, T.P. (ed.) Bridging Scales - Skalenübergreifende Nah- und Fernerkundungsmethoden, 35. Wissenschaftlich-Technische Jahrestagung der DGPF, 24. Deutsche Gesellschaft für Photogrammetrie, Fernerkundung und Geoinformation e.V, Köln (2015), https://mediatum.ub.tum.de/node?id=1245285.

[13] virtualcityMAP: virtualcityMAP, http://www.virtualcitysystems.de/en/products/virtualcitymap.

[14] 3DCityDB-Web-Map: Cesium-based 3D viewer and JavaScript API for the 3D City Database, https://github.com/3dcitydb/3dcitydb-web-map.

[15] Cesium: Cesium - WebGL Virtual Globe and Map Engine, https://cesiumjs.org.

[16] Willenborg, B., Sindram, M., Kolbe, T. H.: Applications of 3D City Models for a better understanding of the Built Environment: (accepted). In Martin Behnisch, Gotthard Meinel (Eds.): Trends in Spatial Analysis and Modelling. Berlin, Heidelberg: Springer (Geotechnologies and the Environment) (2017), https://mediatum.ub.tum.de/node?id=1348882.

[17] 52°North SWE: 52N Sensor Web Community - Sensor Observation Service, http://52north.org/communities/sensorweb/sos.

[18] SSD: Smart Sustainable Districts, http://www.climate-kic.org/programmes/smart-sustainable-districts.

[19] Fusion Tables: Google Fusion Tables Help, https://support.google.com/fusiontables.

[20] virtualcityMAP: virtualcityMAP, http://www.virtualcitysystems.de/en/products/virtualcitymap.

[21] Yao, Z., Kolbe, T.H.: Dynamically Extending Spatial Databases to support CityGML Application Domain Extensions using Graph Transformations. In: Kersten, T.P. (Ed.), Kulturelles Erbe erfassen und bewahren - Von der Dokumentation zum virtuellen Rundgang, 37. Wissenschaftlich-Technische Jahrestagung der DGPF. Deutsche Gesellschaft für Photogrammetrie, Fernerkundung und Geoinformation e.V, Würzburg, pp. 316–331 (2017), http://www.dgpf.de/src/tagung/jt2017/proceedings/proceedings/papers/30_DGPF2017_Yao_Kolbe.pdf.

[22] OpenSensorHub: OpenSensorHub – Software for building smarter sensor networks, https://opensensorhub.org.

[23] FIWARE: FIWARE, https://www.fiware.org.

[24] Ata, R., Goeury, C., Hervouet, J.M.: TELEMAC MODELLING SYSTEM: 2D hydrodynamics TELEMAC-2D Software Release 7.0 User Manual. EDF-R&D, France (2014), http://www.opentelemac.org/downloads/MANUALS/TELEMAC-2D/telemac-2d_user_manual_en_v7p0.pdf

[25] Fewtrell, T.J., Bates, P.D., de Wit, A., Asselman, N., Sayers, P.B.: Comparison of varying complexity numerical models for the prediction of flood inundation in Greenwich, UK. In: FLOODrisk 2008, 30 September - 2 October 2008, Keble College, Oxford, UK (2008).