Integrating Dynamic Data and Sensors with Semantic 3D City Models in the context of Smart Cities

Kanishk Chaturvedi, Thomas H. Kolbe

Chair of Geoinformatics
Technische Universität München
Germany

3DStadtModelle Workshop
Bonn, Germany
November 8-9, 2016

Smart Cities

- Definition by (Yin et al., 2015)

  A smart city is a system integration of technological infrastructure that relies on advanced data processing with the goals of making city governance more efficient, citizens happier, businesses more prosperous and the environment more sustainable

- Worldwide implementations
  - IBM Smarter Cities
  - Microsoft CityNext
  - Smart Sustainable Districts (Climate-KIC of the European Institute of Innovation & Technology)
- Smart cities concepts mainly focus on ICT concepts
  - Internet of Things (IoT), Sensors, Big Data
- It is highly important to work with models of physical reality
  - Semantic 3D City Models are an important complimentary assets

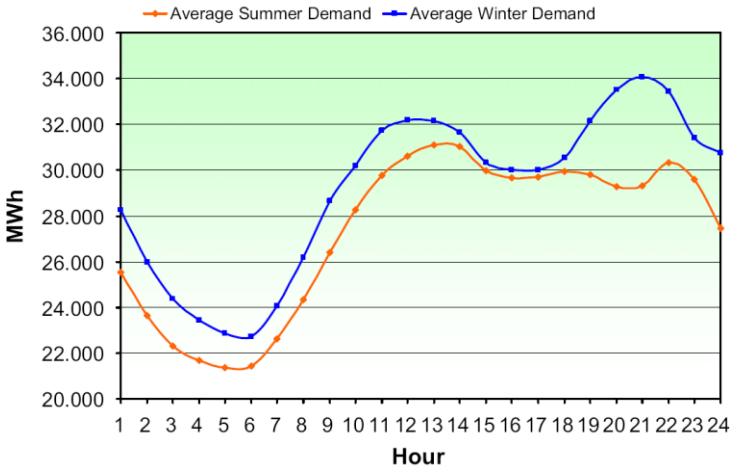
Role of Semantic 3D City Models in Smart Cities

- Semantic 3D City Models
  - Are key for **Urban Information Modelling**
  - Give the spatial context to all the physical entities in the cities
  - Provide a means for interactive and spatio-semantic queries and aggregations

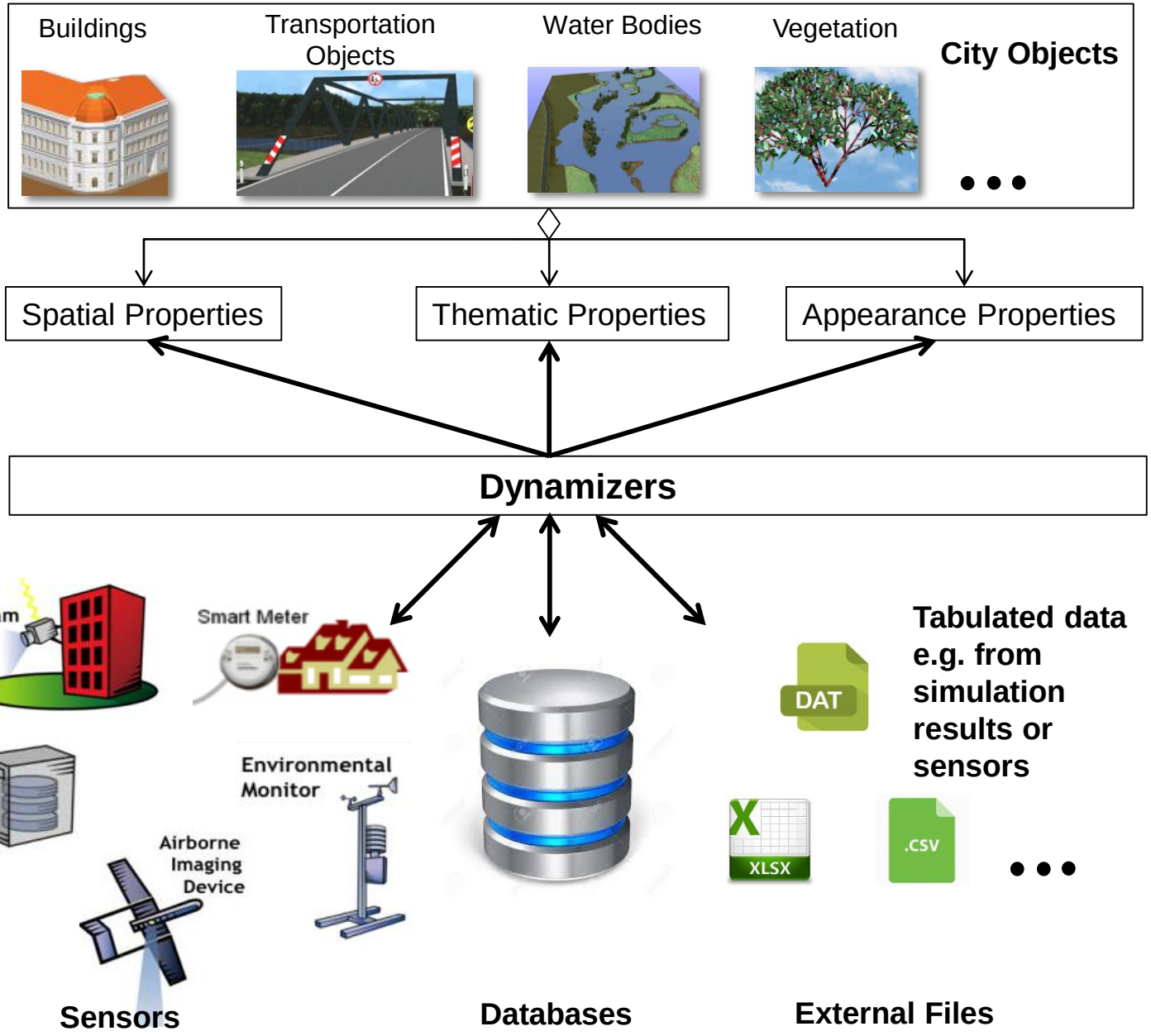
- CityGML (OGC international standard since 2008)
  - **Data model** (UML) + Exchange format (based on GML3)
  - Different thematic areas + Levels of Detail concept (LOD0 - LOD4)
  - 3D geometry, 3D topology, semantics, and appearance
- Semantic 3D city models are mostly static in nature
  - Do not support time-dependent and dynamic properties

What are highly dynamic changes?

- Highly dynamic changes
  - Variations of spatial properties: change of a feature's geometry, both in respect to shape and to location (e.g. moving objects)
  - Variations of thematic properties: changes of physical quantities like energy demands, mean temperature, solar irradiation; air quality in streets and buildings
  - Variations of appearance properties: changes of textures or materials of city objects
  - Variations with respect to sensor or real-time data

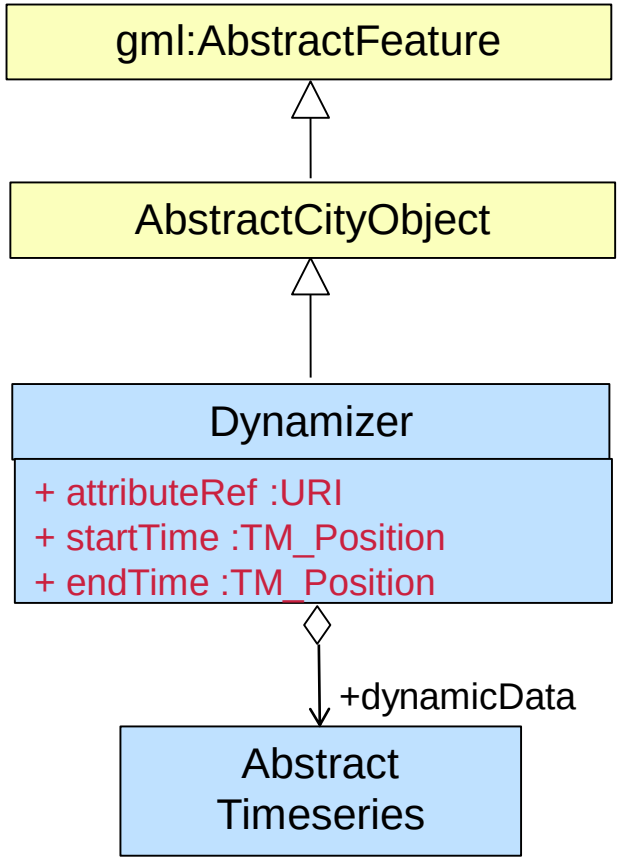


Source:C. García-Ascanio and C. Maté, "Electric power demand forecasting using interval time series: A comparison between VAR and iMLP,"*Energy Policy*


Buildings
Transportation Objects
Water Bodies
Vegetation
City Objects
Spatial Properties
Thematic Properties
Appearance Properties
Dynamizers
Webcam
Smart Meter
Stored Sensor Data
Environmental Monitor
Airborne Imaging Device
DAT
XLSX
.CSV
Tabulated data e.g. from simulation results or sensors
Sensors
Databases
External Files

Dynamizer – New Feature Type

- attributeRef refer to a specific property of a static CityGML feature which value will then be overridden or replaced by the (dynamic) values specified in the ‘Dynamizer’ feature.
- startTime and endTime denote time span for which Dynamizer provides dynamic values
- Dynamizer composes of AbstractTimeseries:
  - Allows represent time-variant values in different and generic ways
  - E.g. Timeseries, Sensor observations etc.



Example Scenario

CityGML object

```
<cityObjectMember>
  <Building gml:id = "building1">
      <gen:doubleAttribute name = "HeatDemand">
          <gen:value = xxx />
      </gen:doubleAttribute>
  </Building>
</cityObjectMember>
```

Source of dynamic data

Estimated Heat Demand (in kwh)	
JAN-15	61578
FEB-15	52148
MAR-15	41011
. . .	. . .
DEC-15	64984

Replacing dynamic attributes using XPath

```
<cityObjectMember>
  <dyn:Dynamizer>
    <dyn:attributeRef> //Building [@gml:id = 'building1']/doubleAttribute[@name = 'HeatDemand']/gen:value</dyn:attributeRef>
    <dyn:startTime> 2015-01-01T00:00:00Z </dyn:startTime>
    <dyn:endTime> 2015-12-31T00:00:00Z </dyn:endTime>
    <dyn:dynamicData>.. </dyn:dynamicData>
  </dyn:Dynamizer>
</cityObjectMember>
```

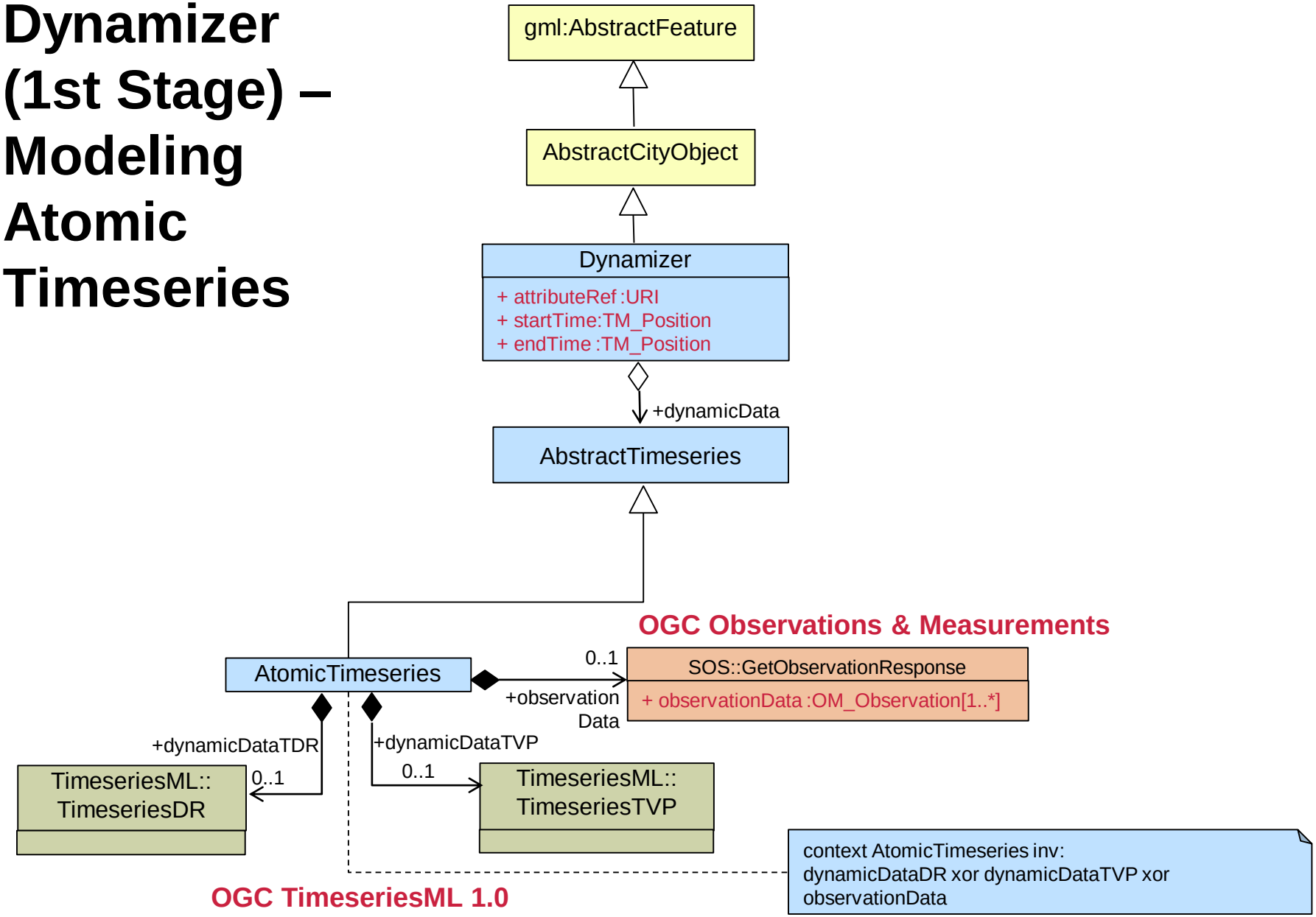
How to model dynamic values?

Dynamizer

How to work with dynamic values?

- **Atomic Timeseries**
  - Tabulation of time/value pairs
    - Thematic properties w.r.t. time
    - Spatial properties w.r.t. time
    - Appearance properties w.r.t. time
  - Different timeseries representations using OGC TimeseriesML 1.0
  - Sensor observations encoded in OGC Observations & Measurements (O&M)
- **Composite Timeseries**
  - Complex and repetitive patterns of time/value pairs
- **Links to Sensors**
  - Explicit links between sensor/observation data and the respective properties of city model objects

Dynamizer (1st Stage) – Modeling Atomic Timeseries



XML Structure – Domain-Range Encoding

```
<cityObjectMember>
  <Building gml:id = "building1">
    <gen:doubleAttribute name = "HeatDemand">
      <gen:value>61578</gen:value>
    </gen:doubleAttribute>
  </Building>
</cityObjectMember>
<cityObjectMember>
  <dyn:Dynamizer gml:id = "HeatDemandTimeseries" >
    <dyn:attributeRef>//Building[@gml:id ='building1']/doubleAttribute[@name = 'HeatDemand']/gen:value </dyn:attributeRef>
    <dyn:startPoint>2016-01-01T00:00:00Z</startPoint>
    <dyn:endPoint>2016-12-01T00:00:00Z</endPoint>
    <dyn:dynamicData>
     <dyn:Timeseries>
      <dyn:dynamicDataDR>
       <tsml:TimeseriesDomainRange gml:id="timeseries">
        <gml:domainSet>
          <tsml:TimePositionList gml:id="temporal_domain">
            <tsml:timePositionList>2016-01-01T00:00:00Z 2016-02-01T00:00:00Z
            2016-03-01T00:00:00Z 2016-04-01T00:00:00Z 2016-05-01T00:00:00Z
            2016-06-01T00:00:00Z 2016-07-01T00:00:00Z 2016-08-01T00:00:00Z
            2016-09-01T00:00:00Z 2016-10-01T00:00:00Z 2016-11-01T00:00:00Z
            2016-12-01T00:00:00Z</tsml:timePositionList>
          </tsml:TimePositionList>
        </gml:domainSet>
        <gml:rangeSet>
          <gml:QuantityList uom="kwh"> 61578 52148 41011 missing 41199 48789 56767
                    66554 76777 67665 missing 66552 </gml:QuantityList>
        </gml:rangeSet>
       </tsml:TimeseriesDomainRange>
      </dyn:dynamicDataDR>
     </dyn:Timeseries>
    </dyn:dynamicData>
  </dyn:Dynamizer>
 <cityObjectMember>
```

CityGML Building

Overriding using XPath

Absolute Time Points

Domain-Range Encoding (Absolute Time Points, can also be irregular time points)

XML Structure – TimeseriesML 1.0 TVP Encoding

```
<cityObjectMember>
  <dyn:Dynamizer gml:id = "HeatDemandTimeseries" >
    <dyn:attributeRef>//Building[@gml:id ='building1']/doubleAttribute[@name = 'HeatDemand']/gen:value </dyn:attributeRef>
    <dyn:startPoint>2016-01-01T00:00:00Z</startPoint>
    <dyn:endPoint>2016-12-01T00:00:00Z</endPoint>
    <dyn:dynamicData>
     <dyn:Timeseries>
      <dyn:dynamicDataTVP>
       <tsml:TimeseriesTVP gml:id="tsml.measurementimeseries.heatdemand">
        <tsml:metadata>
         <tsml:TimeseriesMetadata>
          <tsml:baseTime>2016-01-01T00:30:00.000+12:00</tsml:baseTime>
          <tsml:spacing>PT30M</tsml:spacing>
         </tsml:TimeseriesMetadata>
        </tsml:metadata>
        <tsml:point>
         <tsml:MeasurementTVP>
           <tsml:value>39.97</tsml:value>
         </tsml:MeasurementTVP>
        </tsml:point>
        <tsml:point>
         <tsml:MeasurementTVP>
           <tsml:value>40.12</tsml:value>
         </tsml:MeasurementTVP>
        </tsml:point>
        <tsml:point>
         <tsml:MeasurementTVP>
           <tsml:value>40.02</tsml:value>
         </tsml:MeasurementTVP>
        </tsml:point>
           ............
       </tsml:TimeseriesTVP>
     <dyn:dynamicDataTVP>
    </dyn:Timeseries>
   </dyn:dynamicData>
  </dyn:Dynamizer>
</cityObjectMember>
```

Spacing of 30 minutes

Metadata

Time-Value Pair Encoding
(Relative Time Points, equi-distant/ regular)

XML Structure – Observation encoded in O&M

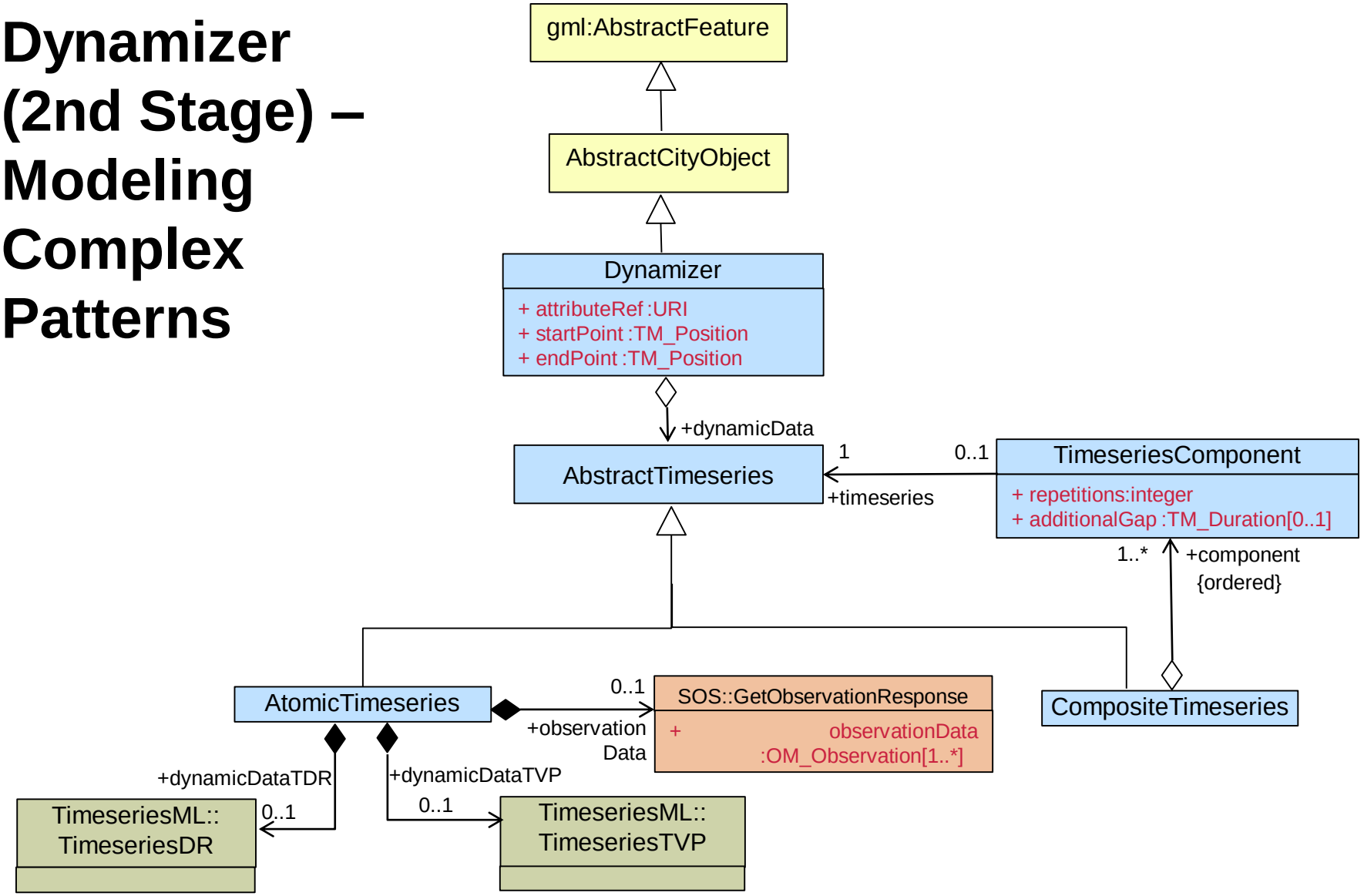
```
<cityObjectMember>
  <dyn:Dynamizer gml:id = "SOSResponse" >
    <dyn:attributeRef> . . . . </attributeRef>
    <dyn:startTime>2015-01-01T00:00:00Z</startTime>
    <dyn:endTime>2015-12-01T00:00:00Z</endTime>
    <dyn:dynamicData>
     <dyn:AtomicTimeseries>
      <dyn:observationData>
       <sos:GetObservationResponse>
         <sos:observationData>
          <om:OM_Observation gml:id="o_1">
            <om:type xlink:href="OM_Measurement"/>
            <om:phenomenonTime>
              <gml:TimeInstant gml:id="phenomenonTime_1">
                <gml:timePosition>2015-11-10T09:00:18.000Z</gml:timePosition>
              </gml:TimeInstant>
            </om:phenomenonTime>
            <om:resultTime xlink:href="#phenomenonTime_1"/>
            <om:procedure xlink:href="Solar_PV_Panel"/>
            <om:observedProperty xlink:href="HeatDemand"/>
            <om:featureOfInterest xlink:href="#building1_roofSurface1"/>
            <om:result>27.7</om:result>
           </om:OM_Observation>
         </sos:observationData>
          . . . . . .
          . . . . . .
       </sos:GetObservationResponse>
    . . . . .
</cityObjectMember>
```

Dynamizer

Observations encoded in O&M

← Other observations

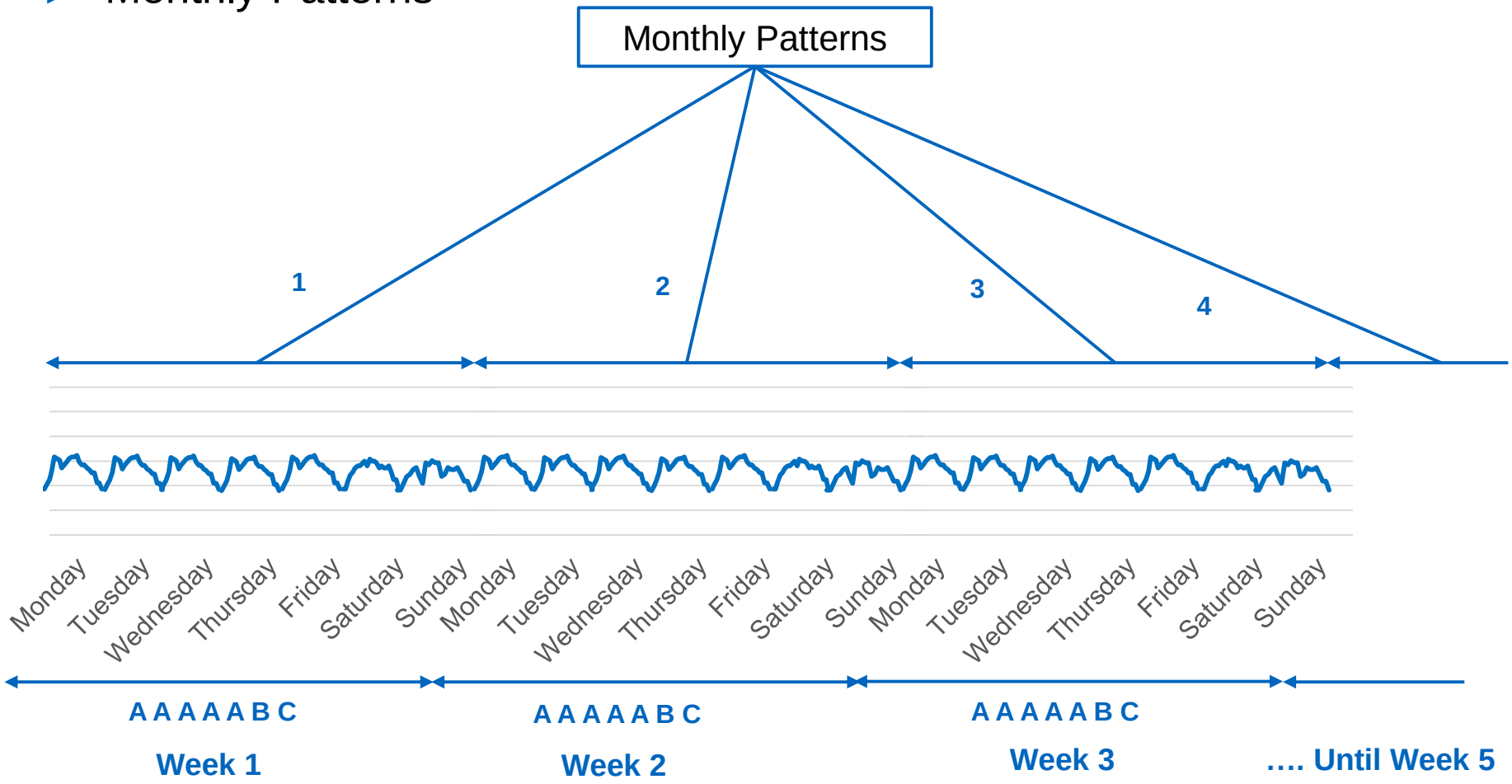
Dynamizer (2nd Stage) – Modeling Complex Patterns



Further classes of TimeseriesML have been omitted here for better readability

Complex Composite Timeseries

- Monthly Patterns



Dynamic Data from Sensors

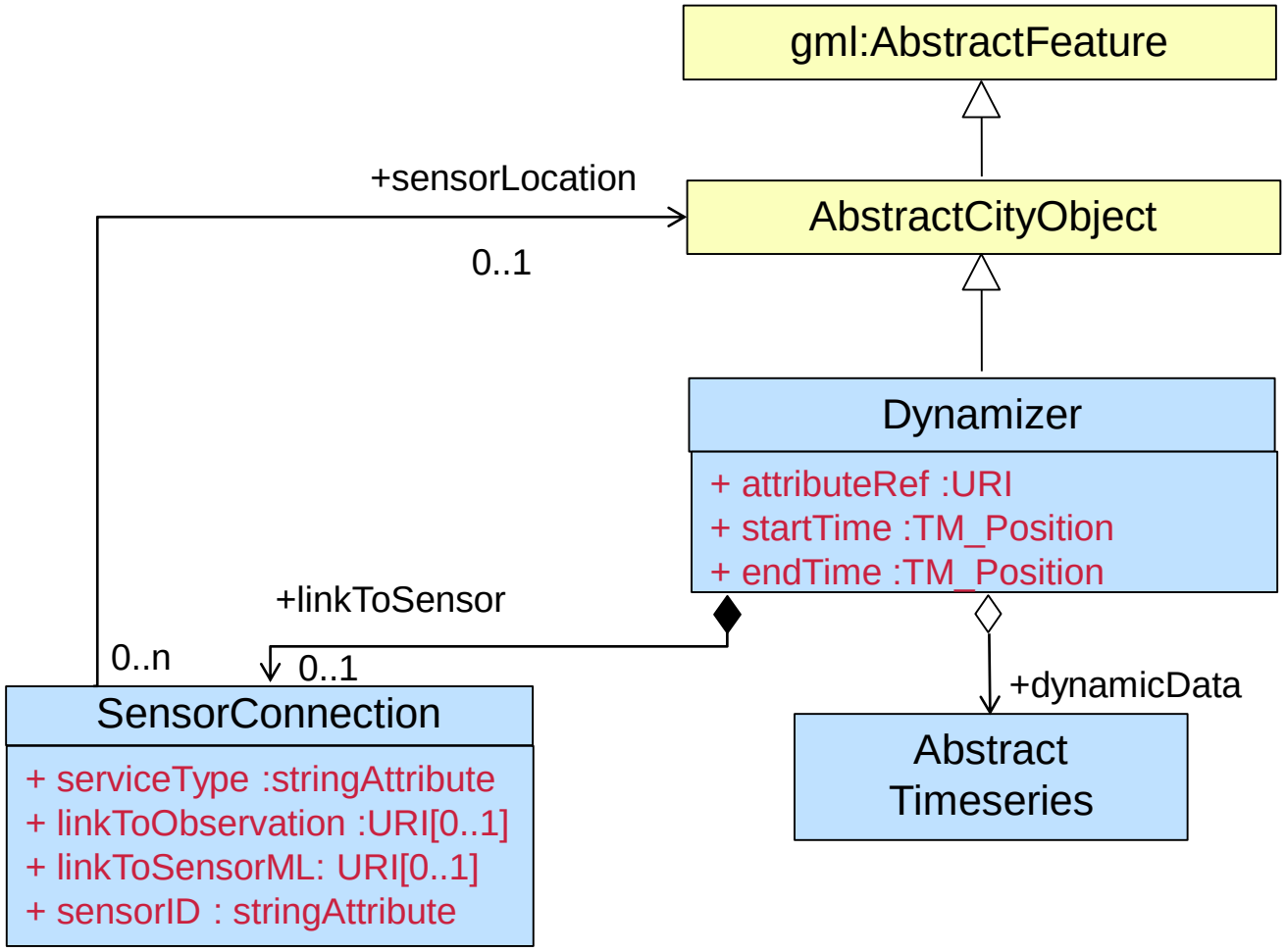
- An important source of dynamic data may be sensor services.
- Two popular standards
  - **OGC Sensor Observation Services (SOS)**
    - Open standard, part of OGC Sensor Web Enablement (SWE)
    - Allows querying real-time sensor data and sensor data timeseries.
    - Observation responses are encoded in O&M standard
  - **OGC SensorThings API**
    - Very lightweight standard to interconnect the Internet of Things devices, data and applications over the web
    - Built on OGC SWE and O&M standards
    - Provides REST services and compact data encodings in JSON format

Source : http://www.opengeospatial.org/ogc/markets-technologies/swe
Source : http://www.sensorup.com/

Integrating Sensors and Observations

- **By including sensor observations within Dynamizers**
  - Sensor observations are typically encoded in O&M format.
  - Dynamizers provide explicit support of O&M as Atomic Timeseries,
    - Hence, the result of an SOS GetObservation request can directly be embedded (i.e. stored inline) within the Dynamizer
  - Storage is an issue for highly frequent observations
- **By linking Dynamizers with Sensors**
  - Such links basically mean that a specific dynamic value for a city object property is measured by a specific sensor (service)
  - Dynamizers represent these direct links to sensors and observations utilizing different requests.
    - in case of OGC SOS: DescribeSensor and GetObservation
  - In order to get the dynamic data, requests to the sensor services must be performed

Dynamizers (3rd stage) – Direct links to sensors and observations



Example for a Sensor Connection

Sensor (PV Panel)

building1_roofSurface1

building1

```
<cityObjectMember>
  <dyn:Dynamizer gml:id = "PV_Power_Timeseries" >
    <dyn:attributeRef>//RoofSurface[@gml:id ='building1_roofSurface1']
                      /doubleAttribute[@name =  'PV_Power']
                      /gen:value </dyn:attributeRef>
    <dyn:startTime>2016-01-01T00:00:00Z</startTime>
    <dyn:endTime>2016-12-01T00:00:00Z</endTime>
    <dyn:linkToSensor>
     <dyn:SensorConnection>
      <dyn:sensorID>. . . </dyn:sensorID>
      <dyn:serviceType>. . . </dyn:serviceType>
      <dyn:linkToObservation>. . . </dyn:linkToObservation>
      <dyn:linkToSensorML>. . . </dyn:linkToSensorML>
      <dyn:sensorLocation xlink:href="#building1_roofSurface1"/>
     </dyn:SensorConnection>
    </dyn:linkToSensor>
  </dyn:Dynamizer>
<cityObjectMember>
```

Unique Sensor ID

SOS or SensorThings API

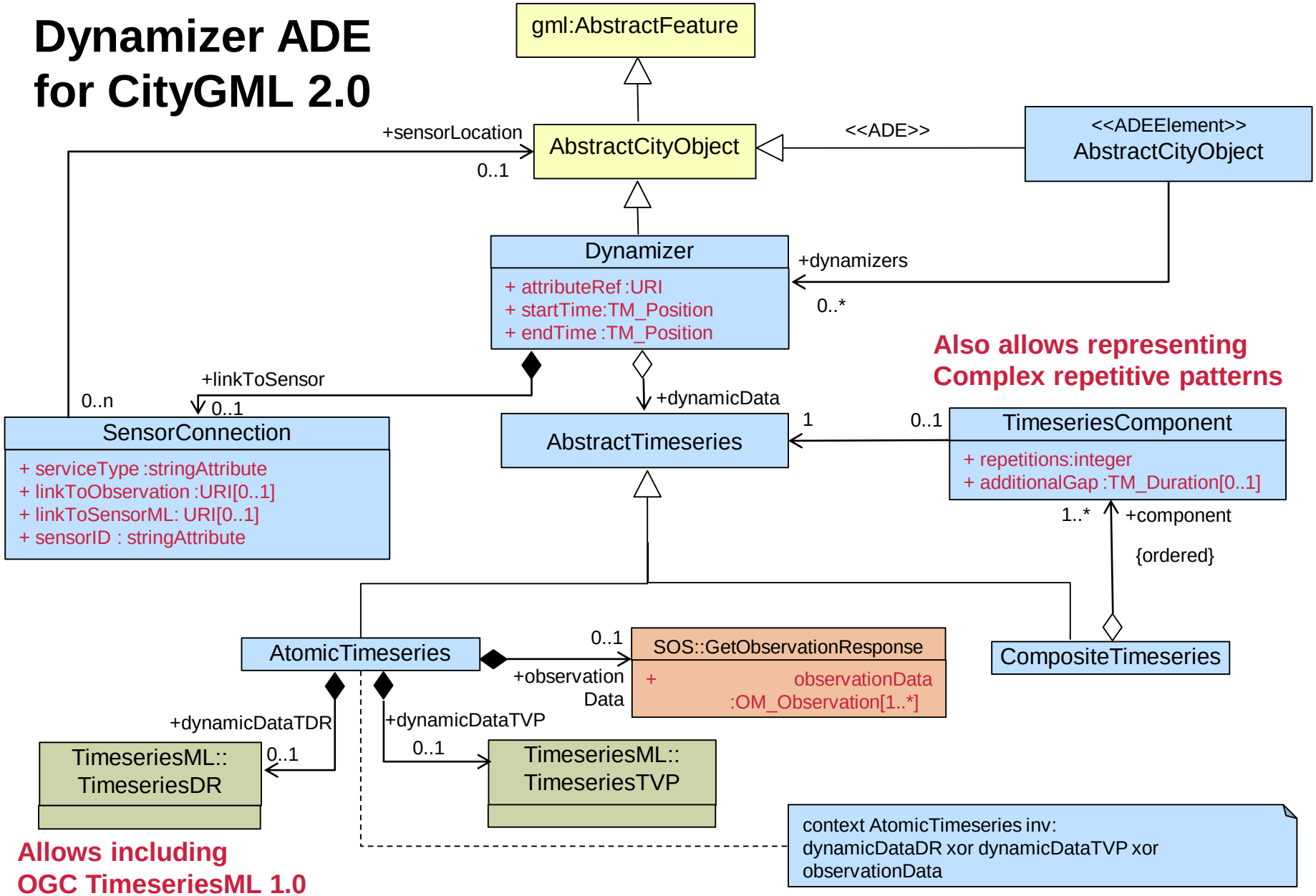
SOS GetObservation

SOS DescribeSensor

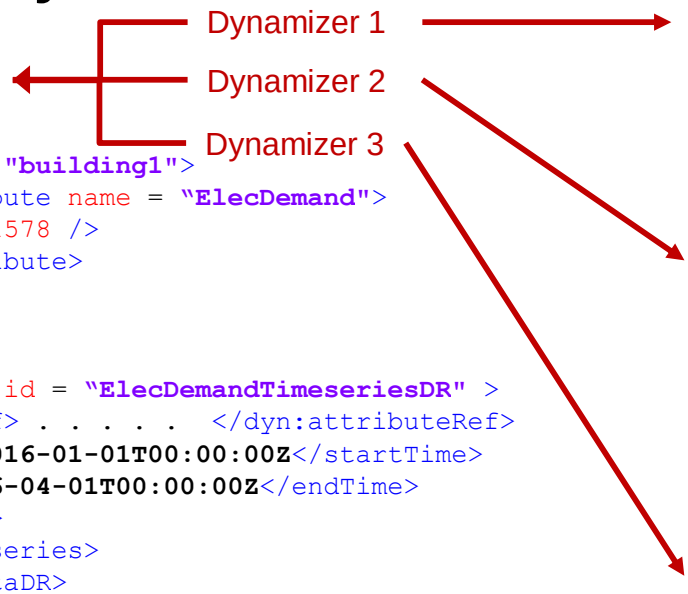
Link to CityGML Object

Image source : http://www.royalgreengas.com/index.php/photovoltaic/residential-buildings

Dynamizer ADE for CityGML 2.0



Combining Dynamic data from different sources



Timeline
Jan-15
Feb-15
Mar-15
Apr-15
May-15
Jun-15
Jul-15
Aug-15
Sep-15
Oct-15
Nov-15
Dec-15

```
<cityObjectMember>
  <Building gml:id = "building1">
    <gen:doubleAttribute name = "ElecDemand">
      <gen:value = 61578 />
    </gen:doubleAttribute>
  </Building>
</cityObjectMember>
<cityObjectMember>
  <dyn:Dynamizer gml:id = "ElecDemandTimeseriesDR" >
    <dyn:attributeRef> . . . . . </dyn:attributeRef>
    <dyn:startTime>2016-01-01T00:00:00Z</startTime>
    <dyn:endTime>2016-04-01T00:00:00Z</endTime>
    <dyn:dynamicData>
     <dyn:AtomicTimeseries>
      <dyn:dynamicDataDR>
       . . . . . .
      </dyn:dynamicDataDR>
     </dyn:AtomicTimeseries>
    </dyn:dynamicData>
  </dyn:Dynamizer>
</cityObjectMember>
<cityObjectMember>
  <dyn:Dynamizer gml:id = "ElecDemandTimeseriesTVP" >
    <dyn:attributeRef> . . . . . </dyn:attributeRef>
    <dyn:startTime>2016-04-01T00:00:00Z</startTime>
    <dyn:endTime>2016-08-01T00:00:00Z</endTime>
    <dyn:dynamicData>
     <dyn:AtomicTimeseries>
      <dyn:dynamicDataDR>
       . . . . . .
      </dyn:dynamicDataDR>
     </dyn:AtomicTimeseries>
    </dyn:dynamicData>
  </dyn:Dynamizer>
</cityObjectMember>
```

```
<cityObjectMember>
    <dyn:Dynamizer gml:id = "SensorObservation" >
      <dyn:attributeRef> . . . . . </dyn:attributeRef>
      <dyn:startTime>2016-08-01T00:00:00Z</startTime>
      <dyn:endTime>2016-12-31T00:00:00Z</endTime>
      <dyn:dynamicData>
       <dyn:AtomicTimeseries>
        <dyn:observationData>
         . . . . . .
        </dyn:observationData>
       </dyn:AtomicTimeseries>
      </dyn:dynamicData>
    </dyn:Dynamizer>
</cityObjectMember>
```

Summary

- Dynamizers enhance static city models by dynamic property values
  - It references a specific attribute (e.g. geometry, semantics or appearance) of an object
  - Overrides the static values of the referenced object attribute by dynamic property values
- Establish explicit links to sensors
  - Linking sensor observations and descriptions
  - Providing location of sensors as city objects
- Dynamizers support multiple dynamic representations
  - OGC TimeseriesML1.0, OGC O&M
- Support complex patterns based on statistics and general rules

Current and Future Work

- Already published in 3DGeoInfo 2016
- OGC Future Cities Pilot Phase 1
  - Dynamizers are being implemented and tested as an Application Domain Extension (ADE) for CityGML 2.0
- CityGML 3.0
  - Dynamizer is intended to become an integral part of next version of CityGML (version 3.0)
- Modeling links to sensors that comply to other standards
  - E.g., OGC SensorThings, Hypercat/SenML, FIWARE
- Dynamizer support in Databases and Visualization Clients
  - Extension of existing CityGML database 3DCityDB
  - Extension of 3DCityDB Web Map visualization client