Dynamizers - CityGML 3.0

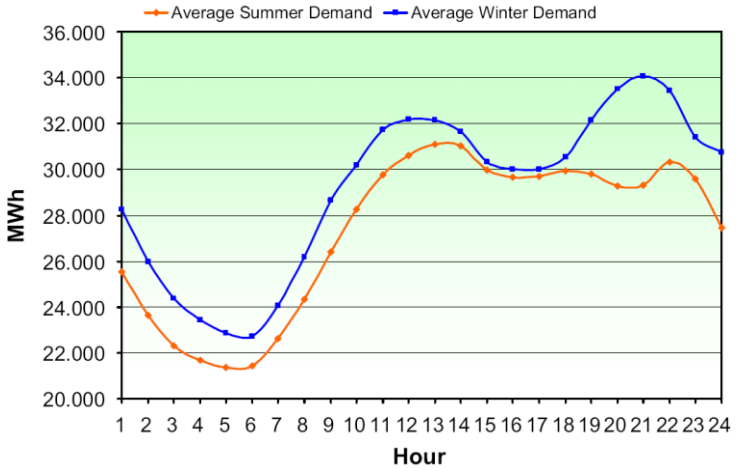
Kanishk Chaturvedi, Thomas H. Kolbe

Chair of Geoinformatics
Technische Universität München

kanishk.chaturvedi@tum.de

Time-varying properties

- Highly dynamic changes
  - Variations of spatial properties: change of a feature's geometry, both in respect to shape and to location (moving objects)
  - Variations of thematic attributes: changes of physical quantities like energy demands, mean temperature, solar irradiation; change of the real property value of a building; change of ownership over time
  - 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*

Source: MOREL M., GESQUIÈRE G., "Managing Temporal Change of Cities with CityGML". In UDMV (2014)

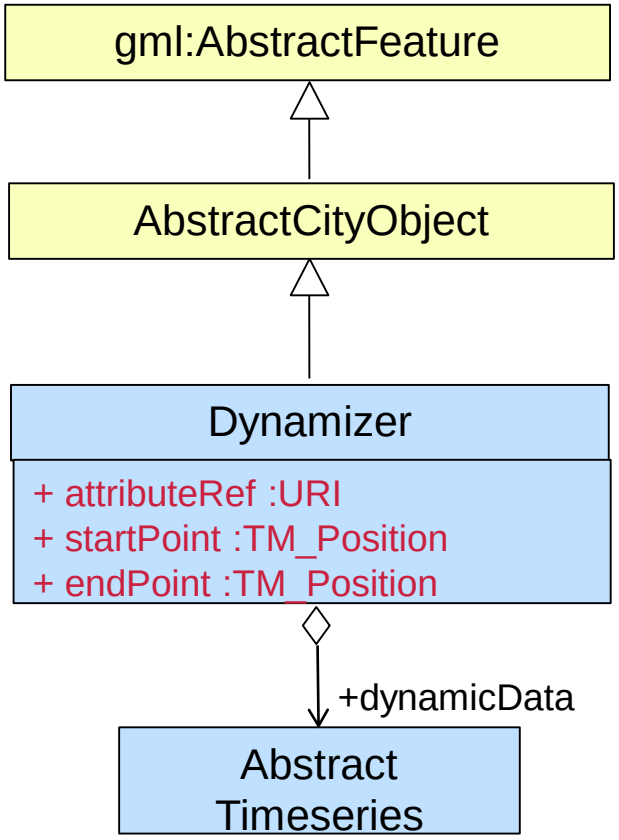
Dynamizers - Proposed approach

- To create a mechanism that allows storing dynamic values separately from original attributes
  - The proposed schema contains dynamic values in special types of features, which would be interpreted as ‘modifiers‘ to the static values of the CityGML feature attributes
  - If an application does not support dynamic data, it simply does not allow/include these special types of features.

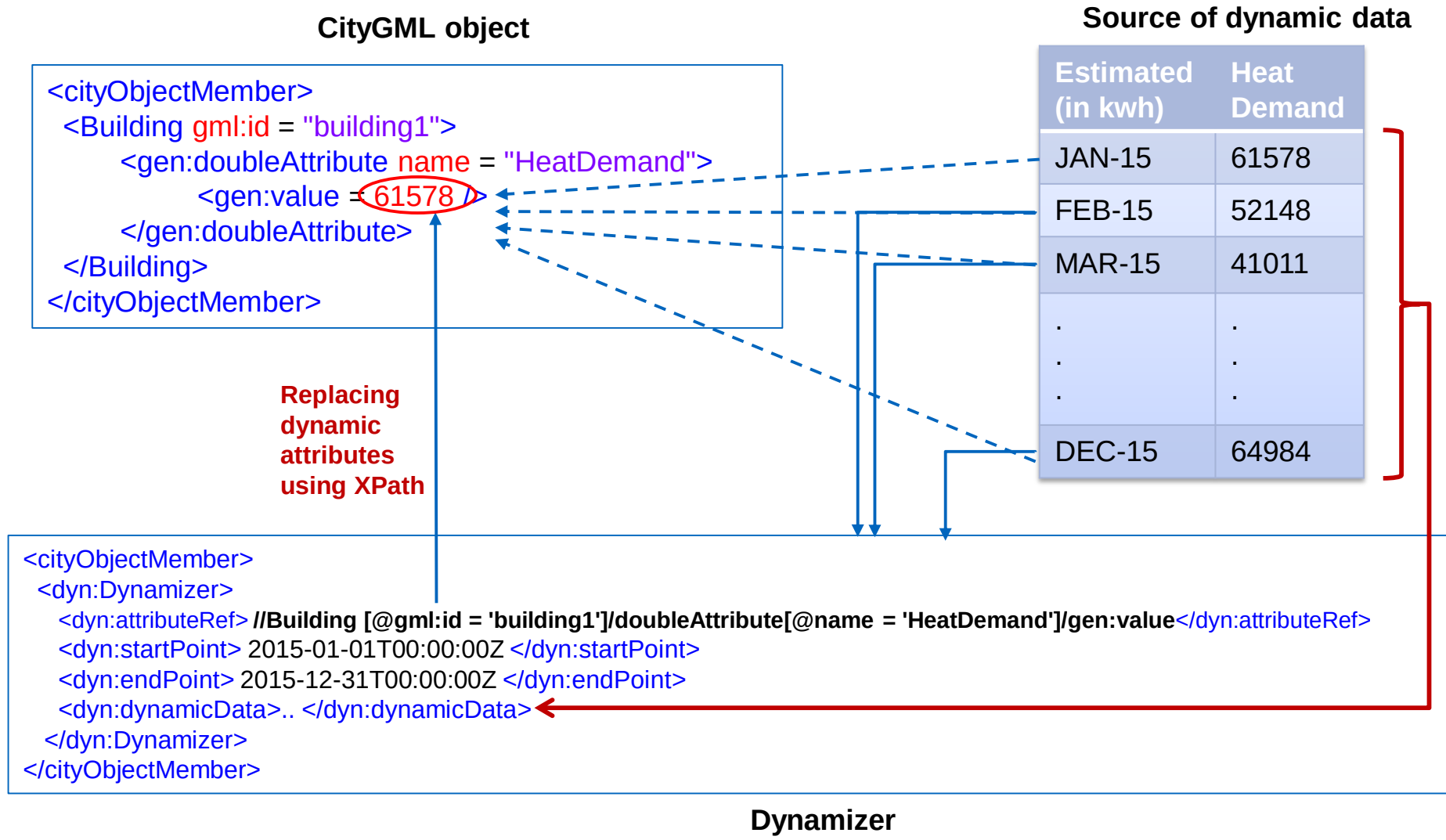
- Advantage: This approach would easily fit into the modularization concept of CityGML.

Dynamizers - Introduction

- Such special types of features are called ‘Dynamizers’.
  - Dynamizers 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.



Example

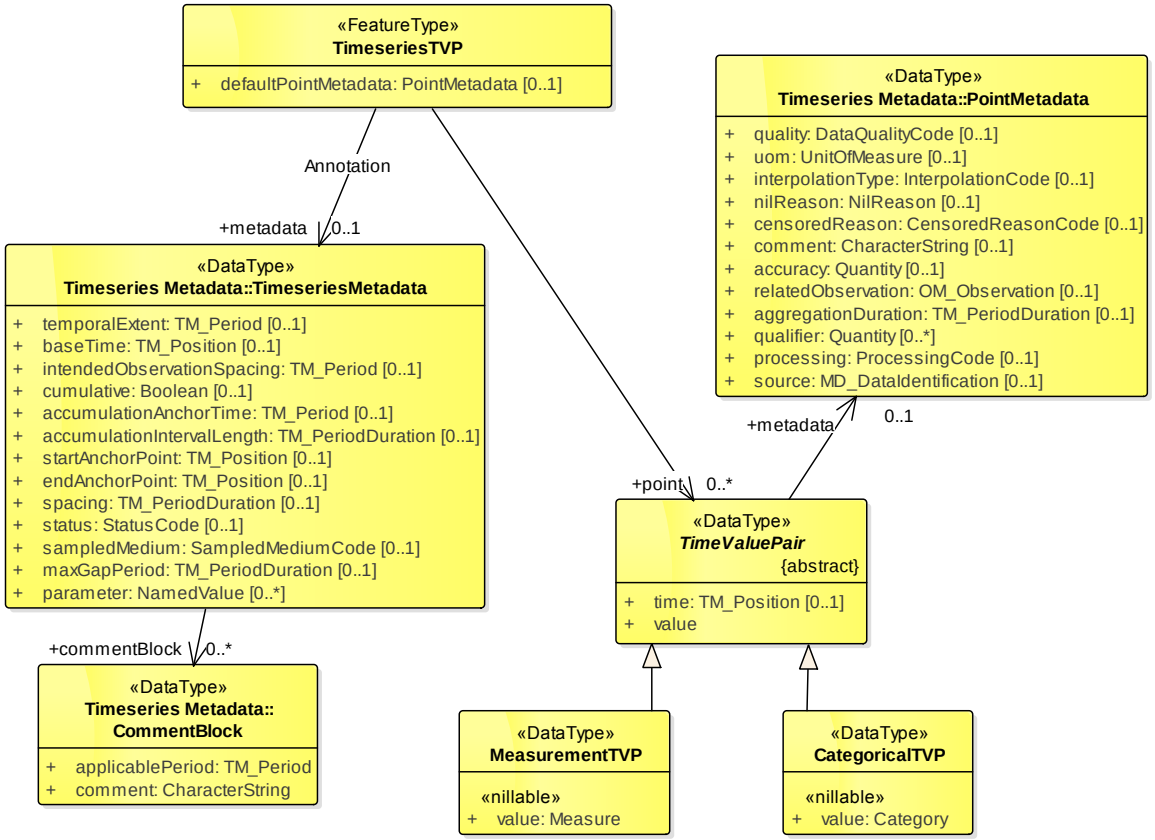

CityGML object
Source of dynamic data
<cityObjectMember>
<Building gml:id = "building1">
<gen:doubleAttribute name = "HeatDemand">
<gen:value = 61578 />
</gen:doubleAttribute>
</Building>
</cityObjectMember>
Estimated (in kwh)
Heat Demand
JAN-15
61578
FEB-15
52148
MAR-15
41011
DEC-15
64984
Replacing dynamic attributes using XPath
<cityObjectMember>
<dyn:Dynamizer>
<dyn:attributeRef> //Building [@gml:id = 'building1']/doubleAttribute[@name = 'HeatDemand']/gen:value</dyn:attributeRef>
<dyn:startPoint> 2015-01-01T00:00:00Z </dyn:startPoint>
<dyn:endPoint> 2015-12-31T00:00:00Z </dyn:endPoint>
<dyn:dynamicData>.. </dyn:dynamicData>
</dyn:Dynamizer>
</cityObjectMember>
Dynamizer

How to specify dynamic values in Dynamizers?

- TimeseriesML 1.0 is a new OGC standard for the representation and exchange of timeseries
  - Extension of the work initially undertaken within OGC WaterML 2.0:Part 1- Timeseries
  - Aim at developing domain-neutral model for the representation and exchange of timeseries data
- Developments
  - OGC 15-043r3: Timeseries Profile of Observations and Measurements
  - OGC 15-042r3: XML encoding that implements the OGC Timeseries Profile of Observations and Measurements

TimeseriesML1.0 – Time-Value Pair Encoding

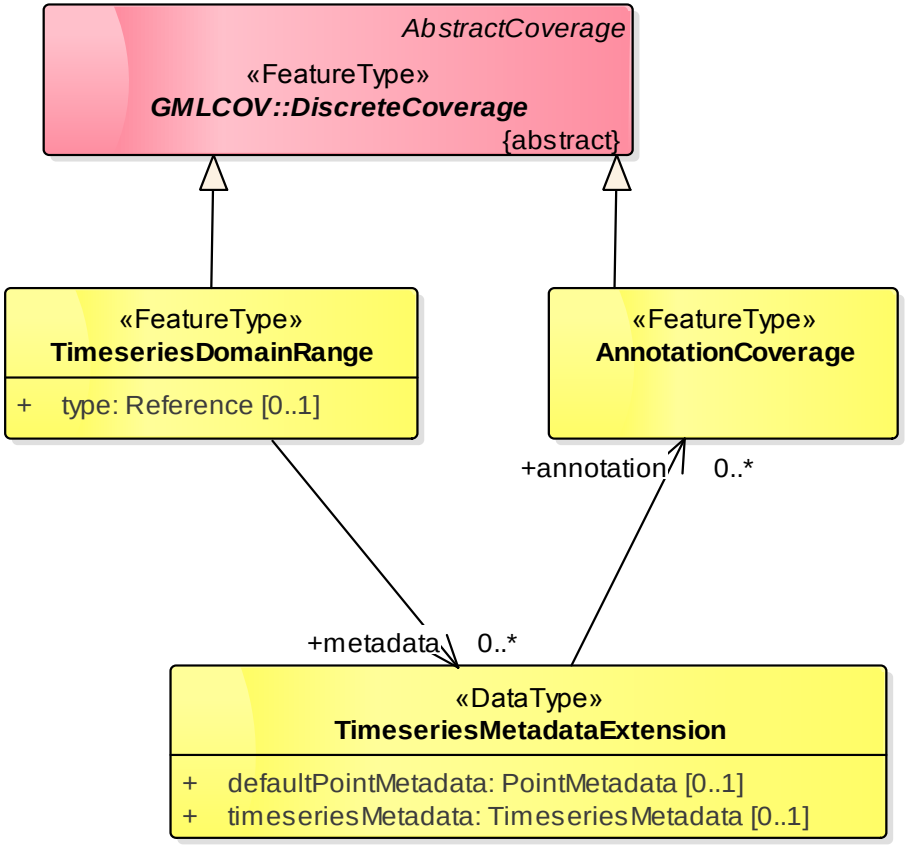
- Representation of a special case of the CV_DiscreteCoverage class from OGC Abstract Specification Topic 6



Source : [OGC 15-043r3 Timeseries Profile of Observations and Measurements]

TimeseriesML 1.0 – Domain-Range Encoding

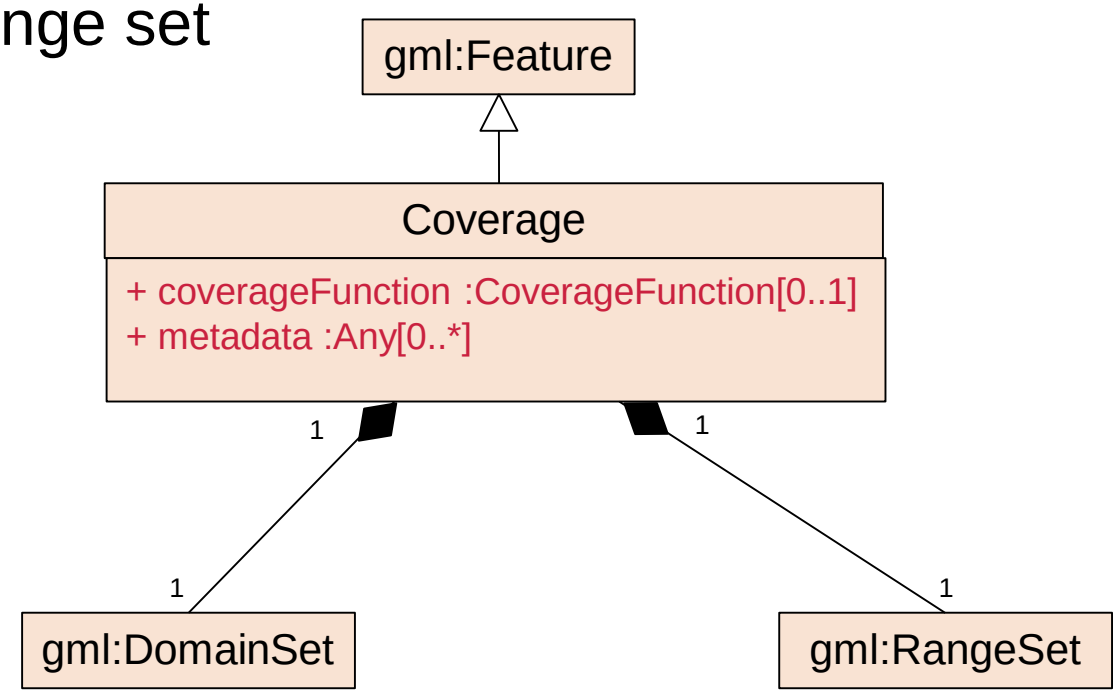
- Extension of OGC Implementation Schema for Coverages (09-146r2)



Source : [OGC 15-043r3 Timeseries Profile of Observations and Measurements]

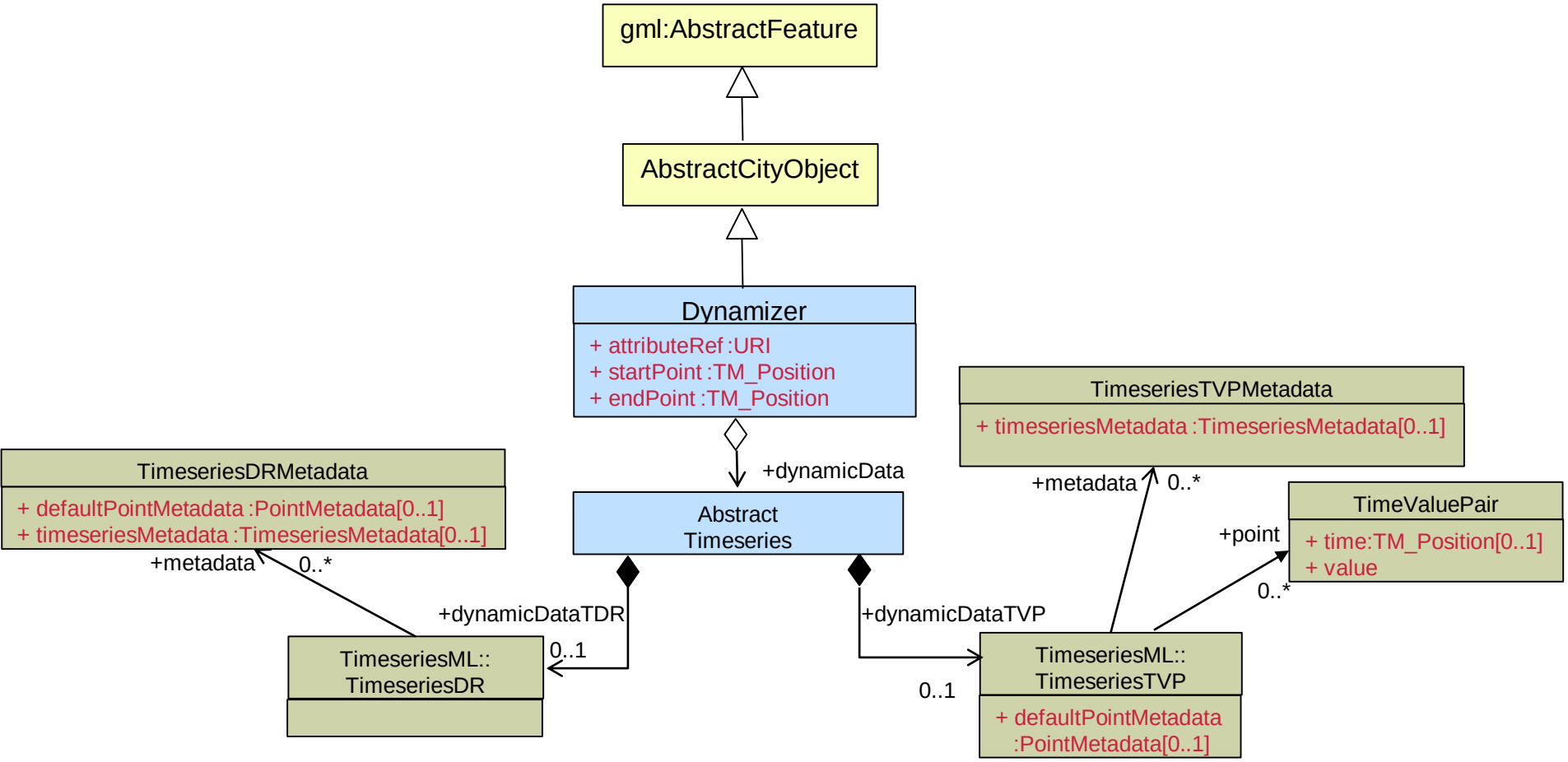
GML Implementation of ISO 19123 - Coverages

- Domain Set (Spatio-temporal values)
- Range Set (attribute values)
- Coverage function, according to which spatio-temporal values from domain set can be mapped to attribute values in the range set



Source : [OGC 09-146 GMLCOV GML Application Schema - Coverages]

Dynamizers (1st Stage) – 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:dynamicDataTDR>
      <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:dynamicDataTDR>
  </dyn:dynamizer>
 <cityObjectMember>
```

CityGML Building

Overriding using XPath

Absolute Time Points

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

Alternative Representation: Time-Value Pair 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:dynamicDataTVP>
     <tsml:TimeseriesTVP gml:id="tsml.measurementimeseries.heatdemand">
       <tsml:point>
        <tsml:MeasurementTVP>
          <tsml:time>2016-01-01T00:00:00Z</tsml:time>
          <tsml:value>39.97</tsml:value>
        </tsml:MeasurementTVP>
       </tsml:point>
       <tsml:point>
        <tsml:MeasurementTVP>
          <tsml:time>2016-01-01T01:00:00Z</tsml:time>
          <tsml:value>40.12</tsml:value>
        </tsml:MeasurementTVP>
       </tsml:point>
       <tsml:point>
        <tsml:MeasurementTVP>
          <tsml:time>2016-01-01T02:00:00Z</tsml:time>
          <tsml:value>40.02</tsml:value>
        </tsml:MeasurementTVP>
       </tsml:point>
          ............
     </tsml:TimeseriesTVP>
    <dyn:dynamicDataTVP>
</dyn:Dynamizer>
</cityObjectMember>
```

Time-Value Pair Encoding (Absolute Time Points, can also be irregular time points)

Relative Time

- Previous examples show absolute time points to be represented in timeseries
- How can we represent relative time points?
- TimeseriesML 1.0
  - Well-defined set of Metadata
  - baseTime – absolute time points (considered as start points)
  - Spacing – time duration, used for calculating regular spacing
- Timeseries feature support both absolute and relative time points
  - however, with a limitation: the start point of each timeseries must be given by an absolute time point
    (→ causes problems in our case; we need to find out, if we can specify a “local“, i.e. relative time reference system)

Handling Relative Time: 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: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:Dynamizer>
</cityObjectMember>
```

Metadata

Spacing of 30 minutes

Currently, it supports absolute time point.

Mechanism required for the support of relative/local time reference system

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

Handling Relative Time: DR 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:dynamicDataTDR>
     <tsml:TimeseriesDomainRange 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>
      <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:dynamicDataTDR>
 </dyn:Dynamizer>
</cityObjectMember>
```

Spacing of 30 minutes

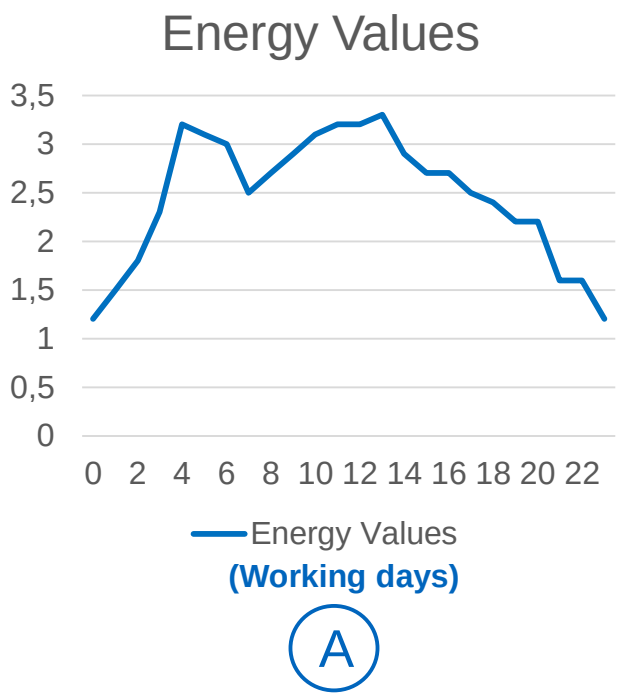
Metadata

DR Encoding (Relative Time Points, equi-distant or regular)

Composite Timeseries - Supporting patterns

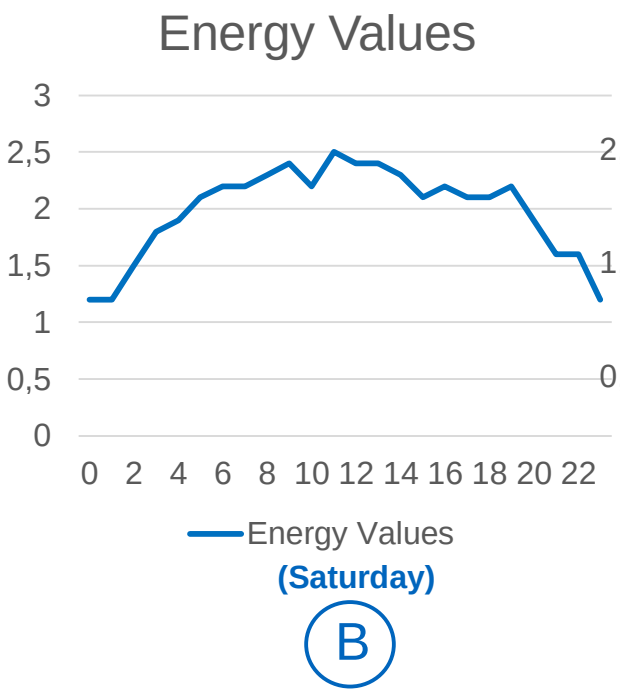
Weekly Patterns

Repeat 5 times



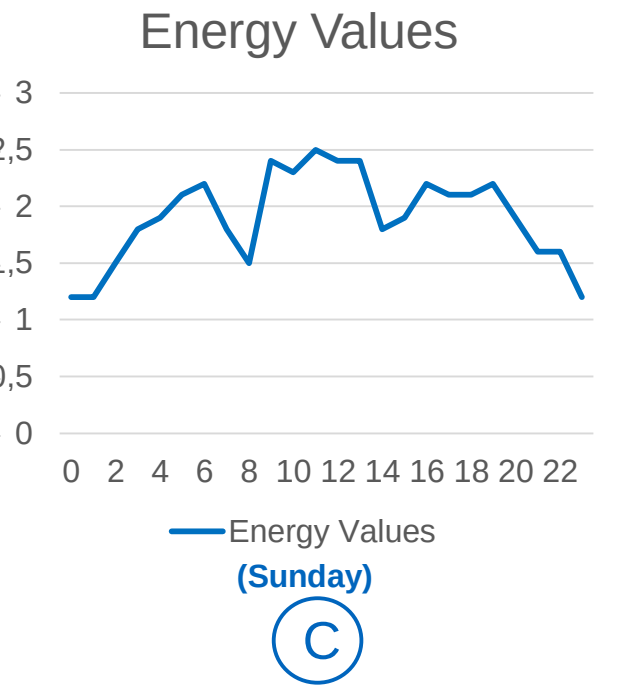
(Working days)

A



(Saturday)

B

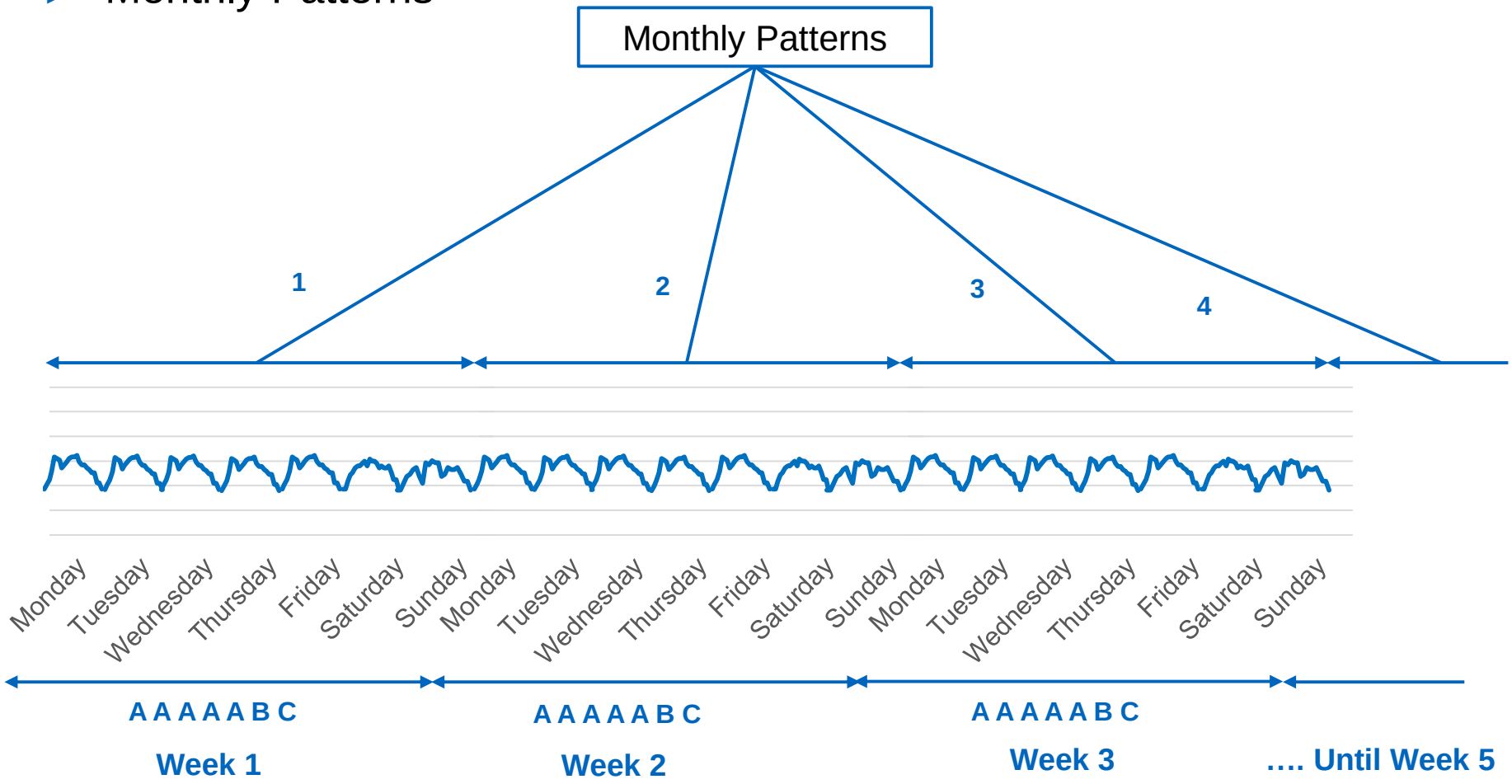


(Sunday)

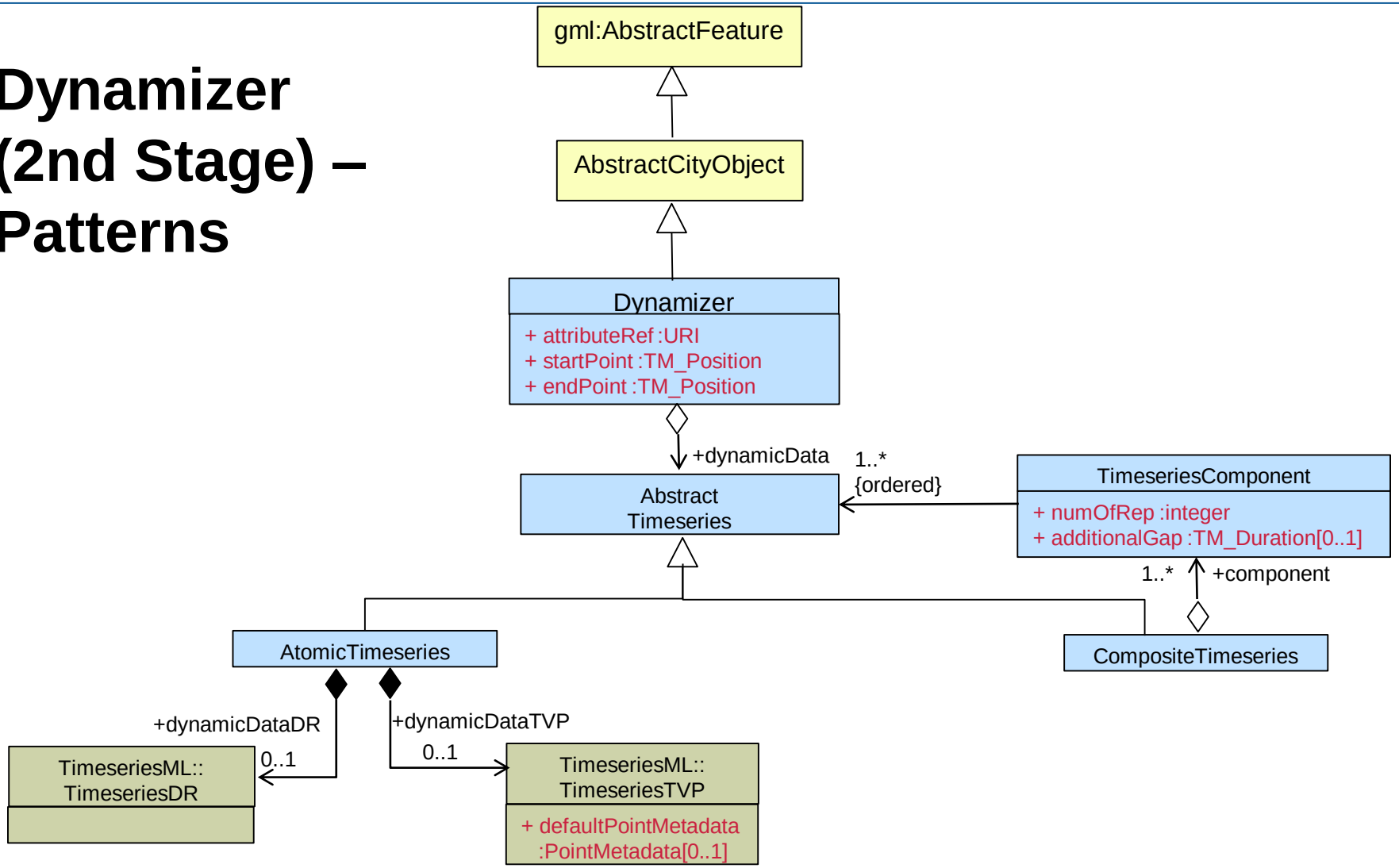
C

Complex Composite Timeseries

- Monthly Patterns



Dynamizer (2nd Stage) – Patterns



Further classes of TimeseriesML have been omitted here for better visibility

Handling Patterns
TVP Encoding

```
<cityObjectMember>
  <dyn:Dynamizer gml:id = "WeeklyPatterns" >
    <dyn:attributeRef>. . . .  </dyn:attributeRef>
    <dyn:startPoint>2016-01-01T00:00:00Z</startPoint>
    <dyn:endPoint>2016-12-01T00:00:00Z</endPoint>
    <dyn:dynamicdata>
     <dyn:CompositeTimeseries>
      <dyn:component>
       <dyn:TimeseriesComponent gml:id="Weekdays">
         <dyn:numberOfRepetitions>5</dyn:numberOfRepetitions>
           <dyn:AtomicTimeseries>
             <tsml:TimeseriesTVP>
               <tsml:metadata>
                 <tsml:TimeseriesMetadata>
                   <tsml:baseTime>2016-01-01T00:30:00.000+12:00</tsml:baseTime>
                   <tsml:spacing>PT1H</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>
              ............
       </dyn:TimeseriesComponent>
       <dyn:TimeseriesComponent gml:id="Saturdays">
         <dyn:numberOfRepetitions>1</dyn:numberOfRepetitions>
           <dyn:AtomicTimeseries>
              <dyn:dynamicDataTVP>
                <tsml:TimeseriesTVP>
                  <tsml:metadata>
                       ............
                  </tsml:metadata>
                  <tsml:point>
                    <tsml:MeasurementTVP>
                         <tsml:value>39.97</tsml:value>
                    </tsml:MeasurementTVP>
                  </tsml:point>
          ............
       </dyn:TimeseriesComponent>
     </dyn:CompositeTimeseries>
```

Spacing of 1 Hour

Timeseries for weekdays

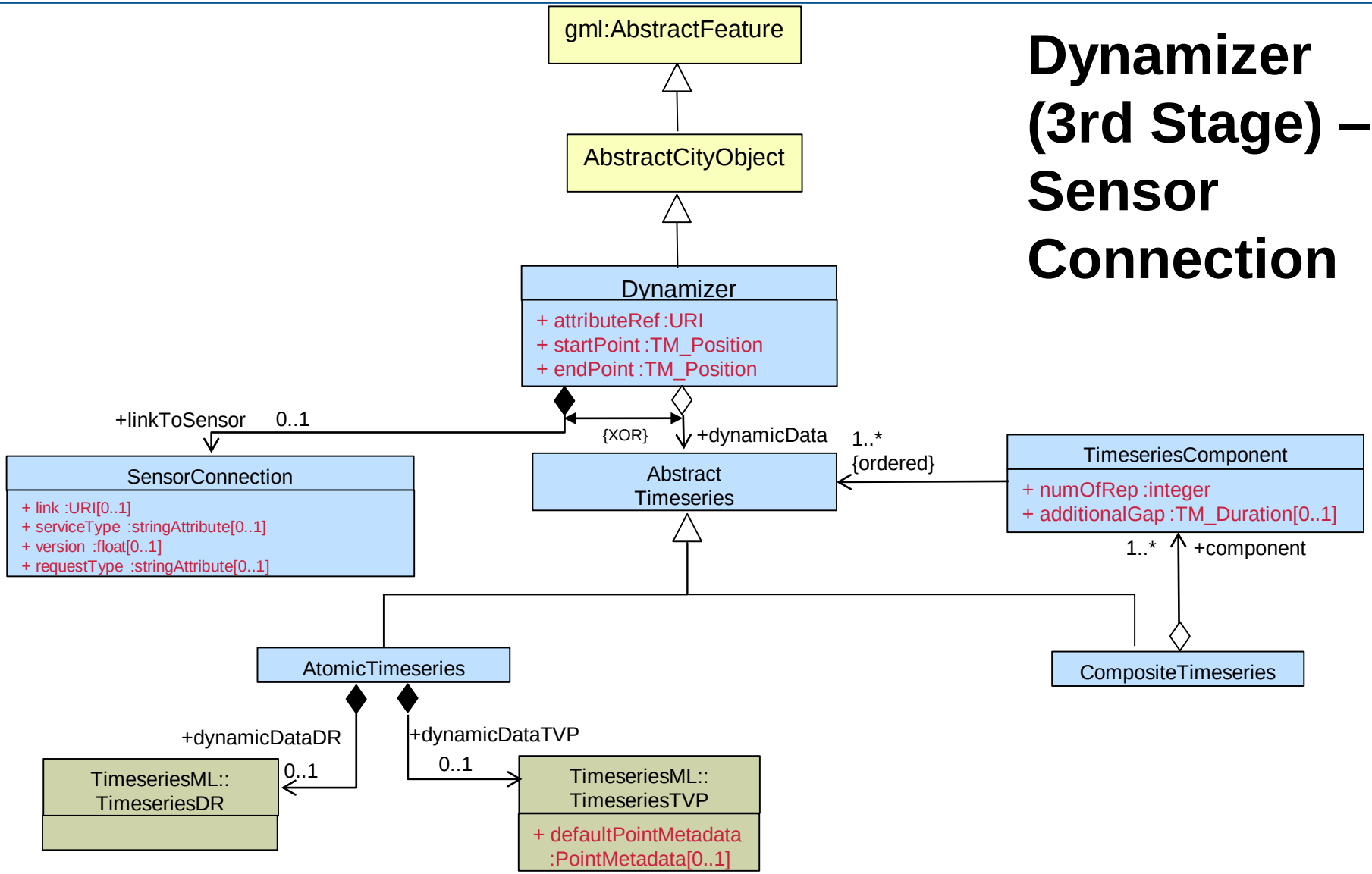
Timeseries for Saturdays

Modeling Sensors Observations

- Important source of dynamic data may also be sensor services.
- Two popular standards
  - OGC Sensor Observation Services (SOS)
    - Open standard and is a 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

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

Dynamizer (3rd Stage) – Sensor Connection



Further classes of TimeseriesML have been omitted here for better visibility

Link to Sensor Observation Services

- Query: Get Observation for a sensor for a specific property (temperature in this example) between a given time period http://129.187.38.201:8080/52n-sos-webapp/service?service=SOS&version=2.0.0&request=GetObservation&featureOfInterest=DHT22_Sensor_Munich&procedure=DHT22_Sensor&observedProperty=Temperature_DHT22&temporalFilter=om:phenomenonTime,2015-11-10T09:00:00Z/2015-11-10T12:00:00Z

- Structure of the request is
  - http://129.187.38.201:8080/52n-sos-webapp/service(SOS instance)
  - REQUEST=GetObservation (SOS Request parameter)
  - SERVICE=SOS&VERSION=2.0.0 (Service of the request)
  - PROCEDURE=DHT22_Sensor (Procedure of the sensor)
  - temporalFiler= 2015-11-10T09:00:00Z/2015-11-10T12:00:00Z

Enriching with Sensor data

- Sensor responses are usually encoded in O&M Format
- Example SOS Response Format

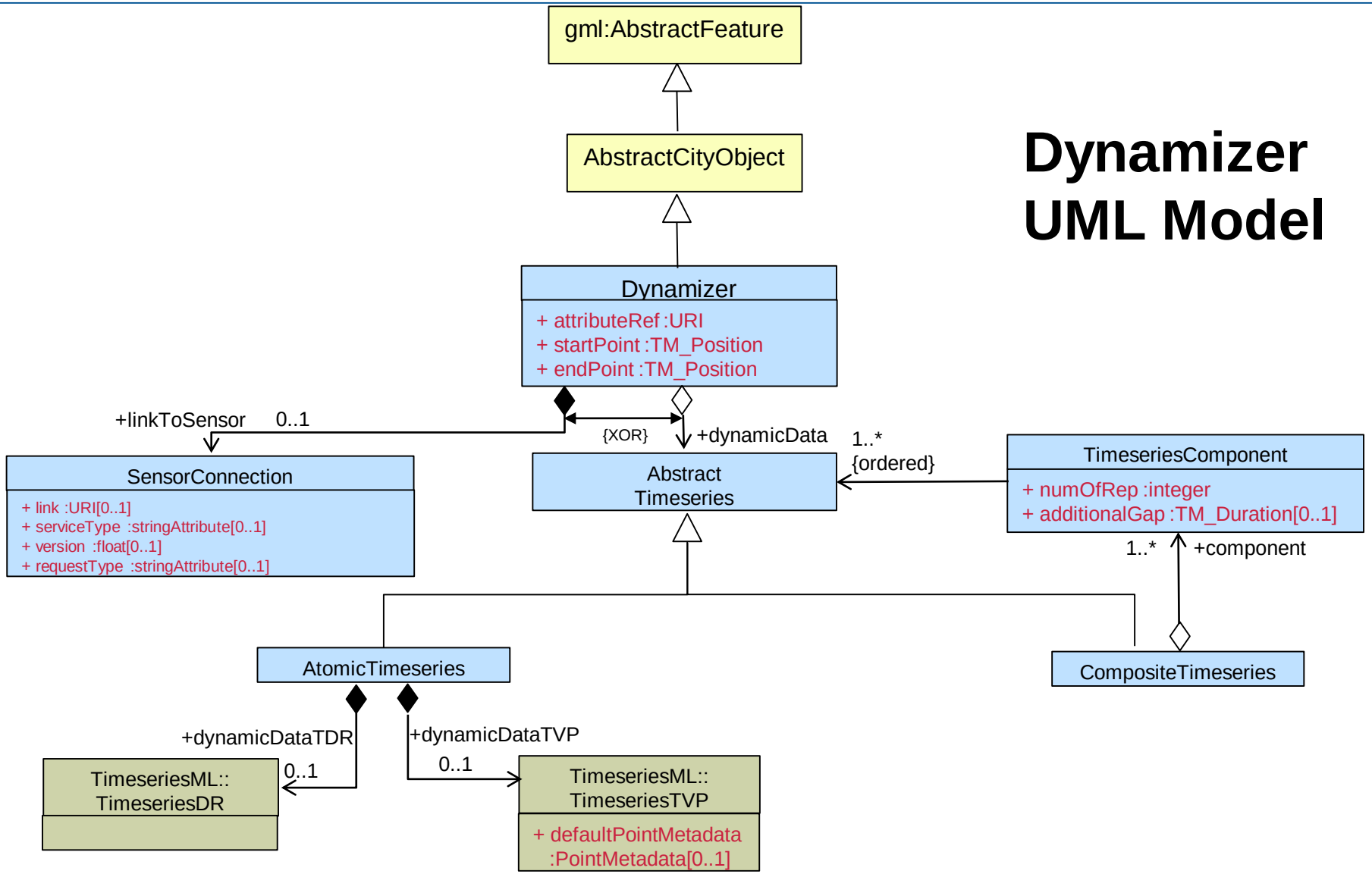
```
<sos:observationData>
  <om:OM_Observation gml:id="o_24581">
    <om:type xlink:href="http://www.opengis.net/def/observationType/OGC-OM/2.0/OM_Measurement"/>
    <om:phenomenonTime>
      <gml:TimeInstant gml:id="phenomenonTime_24581">
        <gml:timePosition>2015-11-10T09:00:18.000Z</gml:timePosition>
      </gml:TimeInstant>
    </om:phenomenonTime>
    <om:resultTime xlink:href="#phenomenonTime_24581"/>
    <om:procedure xlink:href="DHT22_Sensor"/>
    <om:observedProperty xlink:href="Temperature_DHT22"/>
    <om:featureOfInterest xlink:href="DHT22_Sensor_Munich" xlink:title="DHT22_Sensor_Munich"/>
    <om:result>27.7</om:result>
  </om:OM_Observation>
</sos:observationData>
```

- Key-Value Pair
  - om_phenomenonTime, om:result
  - om_resultTime, om:result

Link to Sensor Observation Services

```
<cityObjectMember>
  <dyn:Dynamizer gml:id = „LinkingSensor" >
    <dyn:attributeRef> . . . . </attributeRef>
    <dyn:startPoint>2015-01-01T00:00:00Z</startPoint>
    <dyn:endPoint>2015-12-01T00:00:00Z</endPoint>
    <dyn:linkToSensor>
       <dyn:SensorConnection>
         <dyn:link xlink:href = "http://129.187.38.201:8080/52n-sos-
webapp/service?service=SOS&version=2.0.0&request=GetObservation&featureOfInterest=DHT22_Sensor_Munich&procedure=
DHT22_Sensor&observedProperty=Temperature_DHT22&temporalFilter=om:phenomenonTime,2016-01-01T09:00:00Z/2016-01-
01T12:00:00Z"/>
        </dyn:SensorConnection>
     <dyn:linkToSensor>
  </dyn:Dynamizer>
</cityObjectMember>
```

Dynamizer UML Model



Further classes of TimeseriesML have been omitted here for better visibility

Key benefits of the modified Dynamizer ADE

- Supports multiple dynamic representations
  - Timeseries encoded in Time-value Pair
  - Timeseries encoded in Domain-Range
  - Absolute and relative time
  - Linking external sensor services
- Mappings of missing or multiple attribute values utilizing interpolation and aggregation methods
- Supporting complex patterns based on statistics and general rules
- Future Work
  - Mapping of OGC SOS response within Dynamizers
    - Treats om:phenomenonTime, om:result as key value pair of TimeseriesTVP encoding
    - Allows modeling patterns based on Sensor observations
  - Modeling response of OGC SensorThings

Key benefits of the modified Dynamizer ADE

- Allows defining multiple dynamizers for the same CityGML feature attributes for non-overlapping time periods

```
<cityObjectMember>
  <Building gml:id = "building1">
    <gen:doubleAttribute name = "HeatDemand">
      <gen:value = 61578 />
    </gen:doubleAttribute>
  </Building>
</cityObjectMember>
<cityObjectMember>
  <dyn:Dynamizer gml:id = "HeatDemandTimeseriesTDR" >
    <dyn:startPoint>2016-01-01T00:00:00Z</startPoint>
    <dyn:endPoint>2016-04-01T00:00:00Z</endPoint>
    <dyn:dynamicDataTDR>
       . . . . . .
    </dyn:dynamicDataTDR>
    <attributeRef> . . . . .  </attributeRef>
   </dyn:dynamizer>
<cityObjectMember>
<cityObjectMember>
  <dyn:Dynamizer gml:id = "HeatDemandTimeseriesTVP" >
    <dyn:startPoint>2016-04-01T00:00:00Z</startPoint>
    <dyn:endPoint>2016-08-01T00:00:00Z</endPoint>
    <dyn:dynamicDataTVP>
       . . . . . .
    </dyn:dynamicDataTVP>
    <attributeRef> . . . . .  </attributeRef>
   </dyn:dynamizer>
<cityObjectMember>
```

```
<cityObjectMember>
    <dyn:Dynamizer gml:id = „LinkToSensor" >
      <dyn:startPoint>2016-08-01T00:00:00Z</startPoint
      <dyn:endPoint>2016-12-01T00:00:00Z</endPoint>
      <dyn:LinkToSensor>
         . . . . . .
      </dyn:LinkToSensor>
      <dyn:attributeRef> . . . . .  </attributeRef>
     </dyn:dynamizer>
```