Versioning concept for CityGML 3.0

Minutes of the Face-to-face Meeting of CityGML 3.0 WP06

Thomas H. Kolbe, Kanishk Chaturvedi, Tatjana Kutzner

Chair of Geoinformatics
Technische Universität München

kanishk.chaturvedi@tum.de

6-7 November 2014

Face-to-face Meeting

- November 6-7, 2014 in Munich, Germany

Objective of WP06

- Aim 1: Tighter coupling of semantic 3D city models and simulations by extending the CityGML feature representation to support variations of individual feature properties and associations over time
  - 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
- Aim 2: Extending CityGML by mechanisms for denoting versions of models or model elements as planning alternatives

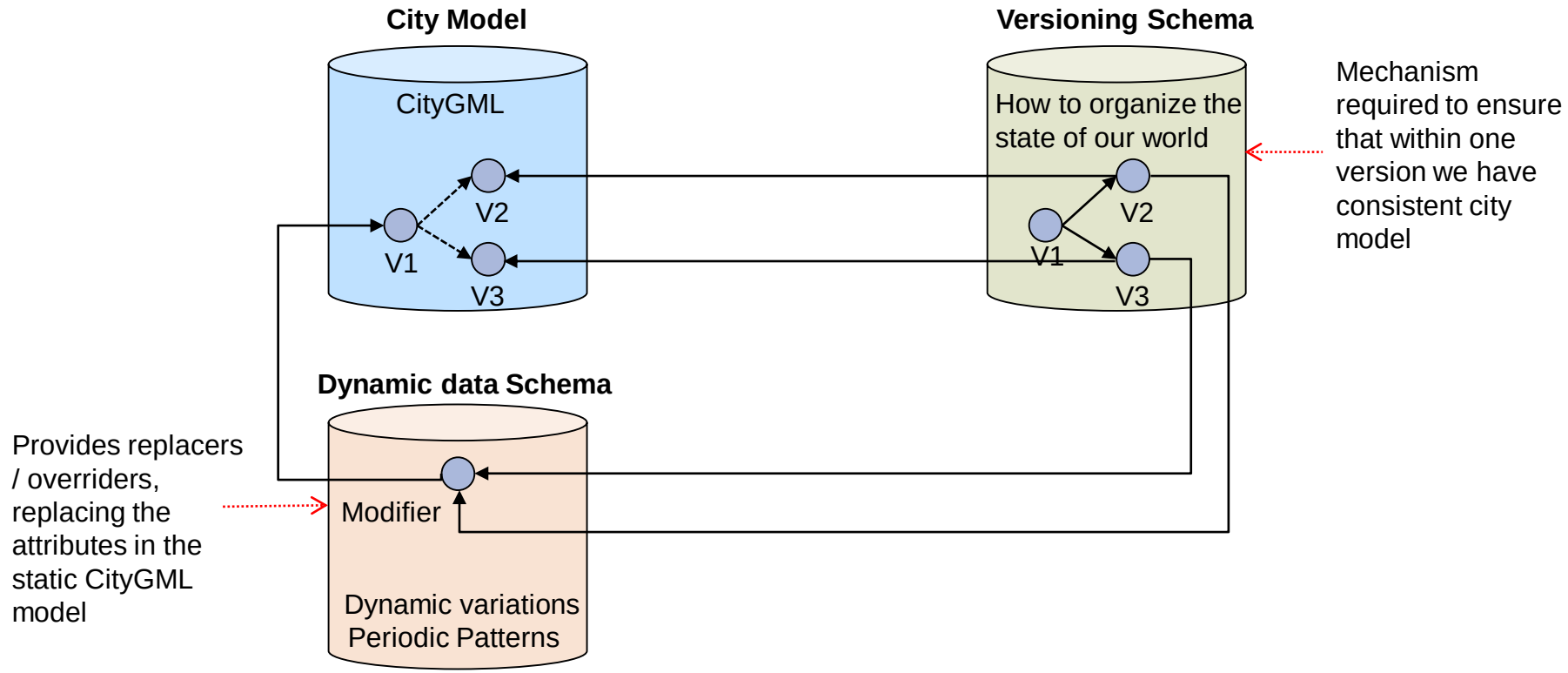
Proposed approach

- To create a mechanism that allows to store time variant or time-depending values separately from the original attributes
  - To develop the versioning schema to organize the state of our world (evolution) in the form of separate versions
  - To develop the dynamic data schema, containing the time-variant or time depending 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.

Conceptual Model

- To develop
  - A versioning schema, which represents the evolution of the city (model) in the form of different versions.
  - A dynamic data schema, where dynamic variations can be stored in special types of features, which would be interpreted as modifiers to the static CityGML model.



Versioning

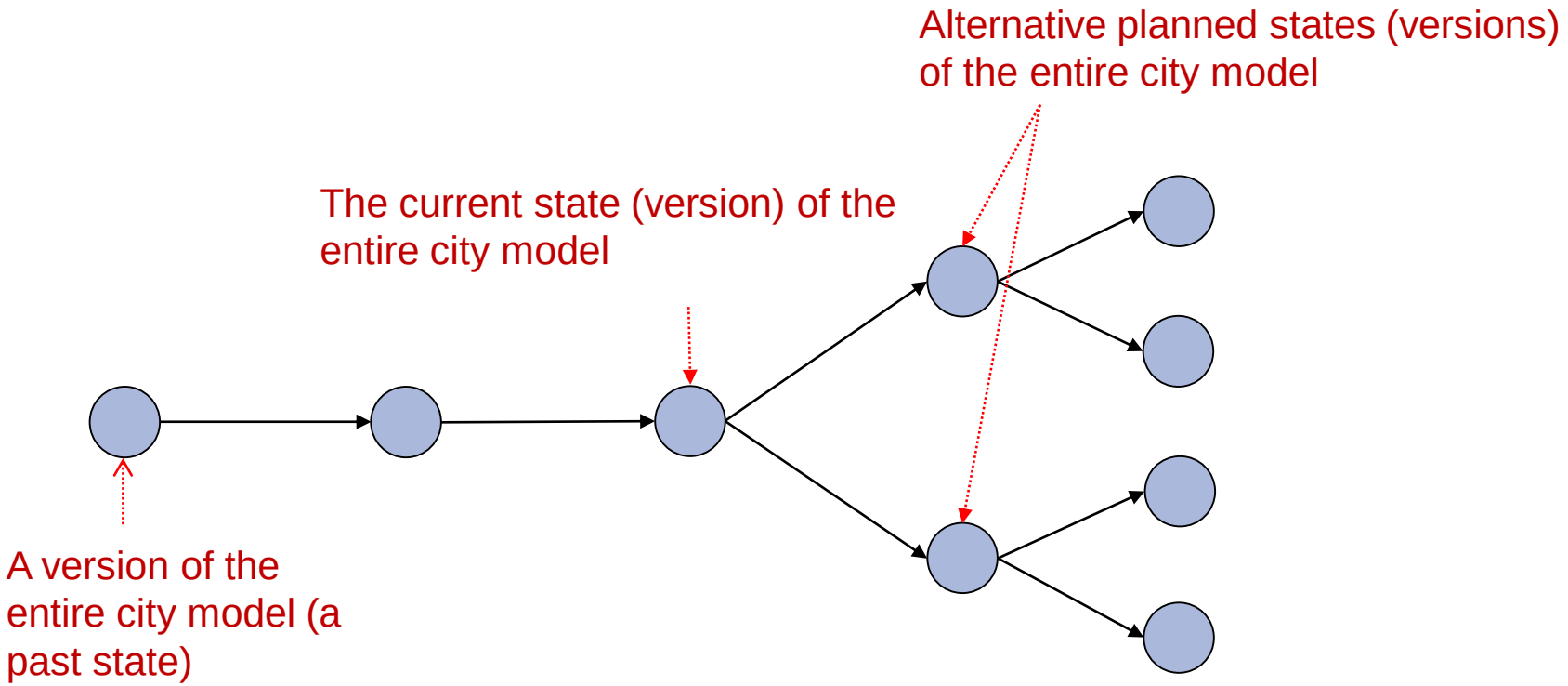
- Allows to identify and organize multiple states of a city model

- Versions may represent the evolution of the city models
  - How did the city look like at a specific point in time?
  - How did the city model look like at a specific point in time?

- Versions may represent the features as alternatives
  - To represent future planning alternatives / competing designs
  - To represent different interpretations of the past
  - To represent present data of different data qualities (e.g. because of different sources)

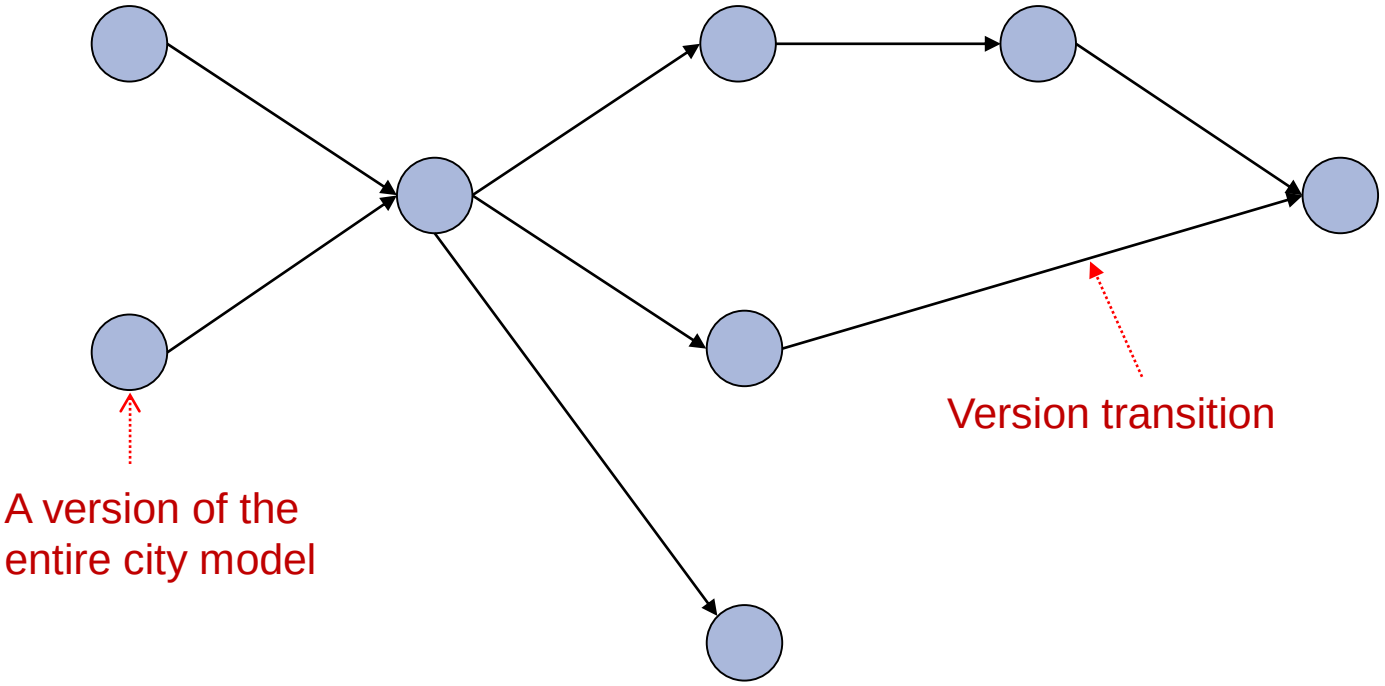
Versioning

- **Example 1**: Multiple representations of the city model can be handled within different versions, which can be modeled as feature types.



Versioning

- **Example 2:** Directed graph allows for confluence and forking



Requirements for versioning of CityGML features (I)

- Each object needs to be assigned a stable object ID (major ID)
  - is supported by GML 3.2.1 through the element “gml:identifier” which can be used for providing globally unique identifiers
- ID extension to distinguish different versions of the same real world entity as “Sub ID” (minor ID)
- Introduce a “separator” symbol which separates the identifier from the version, e.g. Building1020_Version1
  - The concatenated major and minor ID should be used as the gml:id to distinguish the different versions of the same real world object
  - One CityGML instance document can therefore include multiple versions of the same real world object having different gml:id but identical gml:identifier

Requirements for versioning of CityGML features (II)

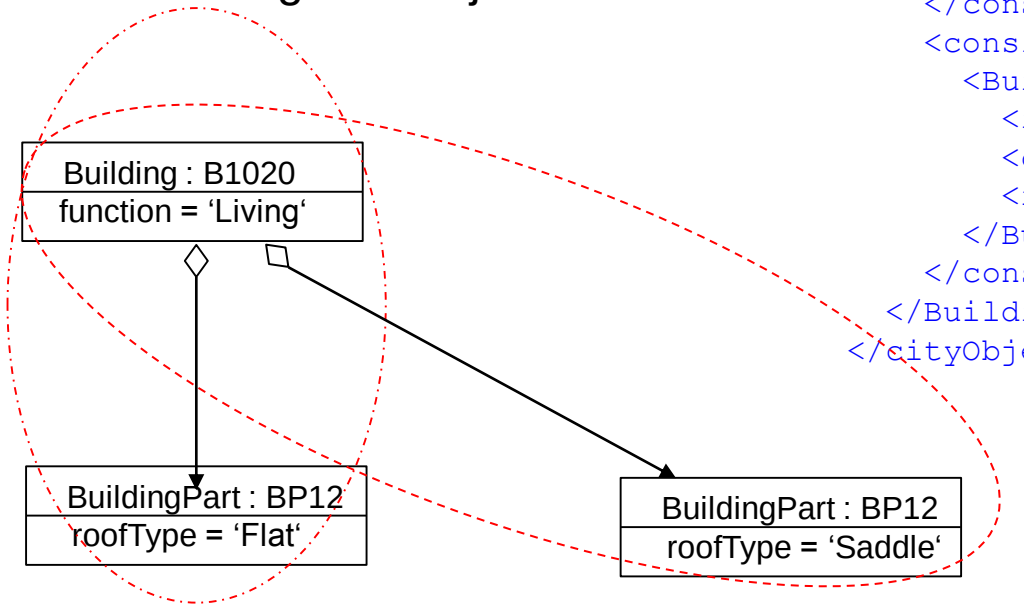
- In order to express a bi-temporal existence model, all city objects shall have the attributes:
  - CreationDate and TerminationDate (which reflect the database transaction time)
  - ValidFrom and ValidTo (which reflect the actual world time)
- This is similar to the INSPIRE data models
- These four attributes
  - can be used to query how the city model or the city itself looked at a specific point in time
  - can be defined as an extension to the CityGML core module
  - may replace yearOfCreation and yearOfDemolition in Building module

Referencing objects and their versions

- The cityobjects can be referenced by
  - A simple XLink to the gml:id of the referenced object, which references a specific version of a real world object
  - An XPath-XLink, which is a general reference to a real world object identified by its major ID, and not taking into account a specific version
    - multiple instances with the same gml:identifier value, but different gml:id may be selected this way
    - it needs to be determined by the application which specific version of the real world object representation should be used
    - the attributes creationDate, terminationDate, validFrom, validTo could be used to choose the appropriate version that was valid at a specific database or real world time respectively

Example 1 – Instance Data

- Representation of two different versions of one real world building
- Roof type of BuildingPart changes from ‘Flat‘ to ‘Saddle‘
- Building Major ID: B1020
- BuildingPart Major ID: BP12




```
<cityObjectMember>
  <Building gml:id="B1020_version1">
    <identifier>B1020</identifier>
    <creationDate>2012-08-02</creationDate>
    <function>Living</function>
    <consistsOfBuildingPart>
      <BuildingPart gml:id="BP12_version1">
        <identifier>BP12</identifier>
        <creationDate>2012-08-02</creationDate>
        <terminationDate>2013-10-19</terminationDate>
        <roofType>Flat</roofType>
      </BuildingPart>
    </consistsOfBuildingPart>
    <consistsOfBuildingPart>
      <BuildingPart gml:id="BP12_version2">
        <identifier>BP12</identifier>
        <creationDate>2013-10-19</creationDate>
        <roofType>Saddle</roofType>
      </BuildingPart>
    </consistsOfBuildingPart>
  </Building>
</cityObjectMember>
```

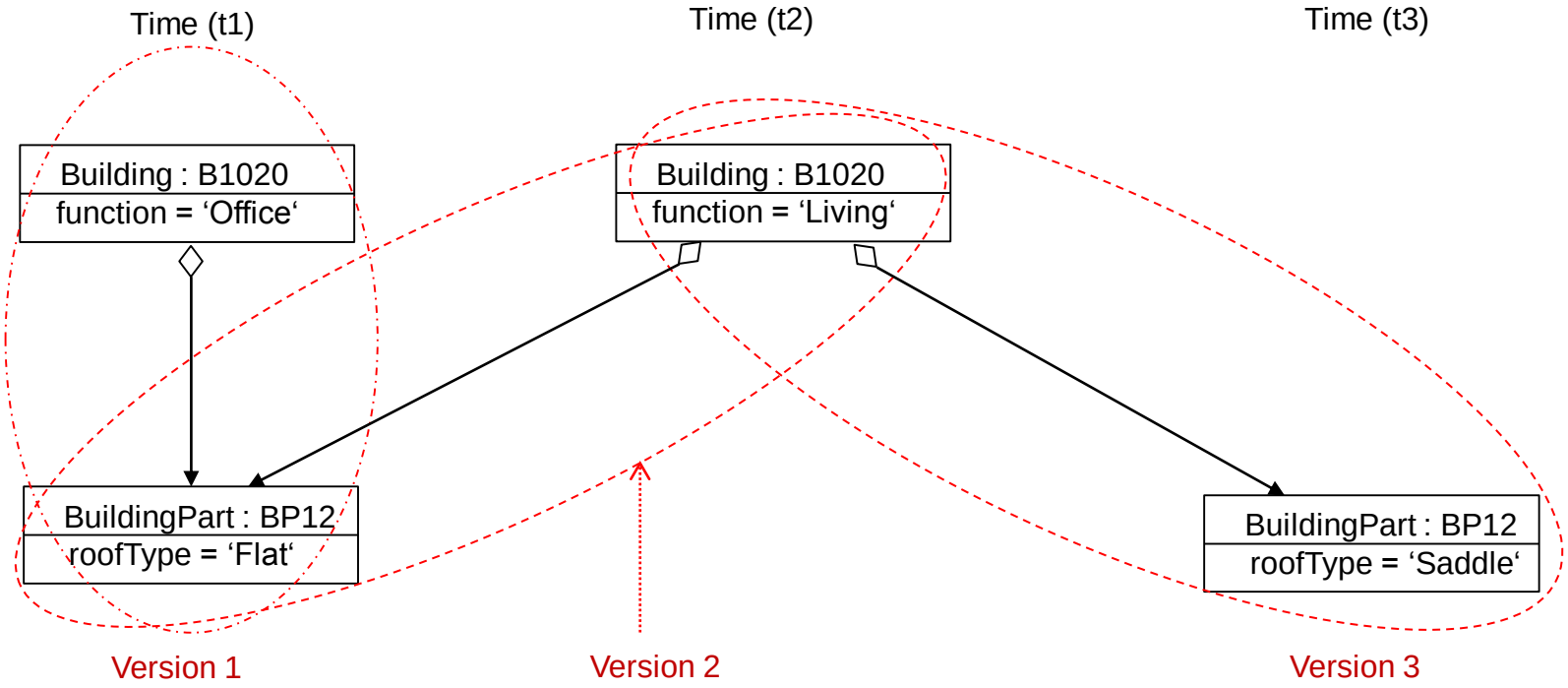
Example 1 – Instance Data (Alternative Representation)

- Semantically equivalent to the instance document from previous slide

```
<cityObjectMember>
   <Building gml:id="B1020_version1">
      <identifier>B1020</identifier>
      <creationDate>2012-08-02</creationDate>
      <consistsOfBuildingPart>
         <BuildingPart xlink:href="//identifier[text()='BP12']"/>
      <consistsOfBuildingPart>
      <function>Living</function>
   </Building>
</cityObjectMember>
<cityObjectMember>
   <BuildingPart gml:id="BP12_version1">
      <identifier>BP12</identifier>
      <creationDate>2012-08-02</creationDate>
      <terminationDate>2013-10-19</terminationDate>
      <roofType>Flat</roofType>
   </BuildingPart>
</cityObjectMember>
<cityObjectMember>
   <BuildingPart gml:id="BP12_version2">
      <identifier>BP12</identifier>
      <creationDate>2013-10-19</creationDate>
      <roofType>Saddle</roofType>
   </BuildingPart>
</cityObjectMember>
```

Example 2

- Update 1: Building function changes from ‘Office‘ to ‘Living‘
- Update 2: BuildingPart changes from ‘Flat‘ to ‘Saddle‘.



Example 2 – Instance Data

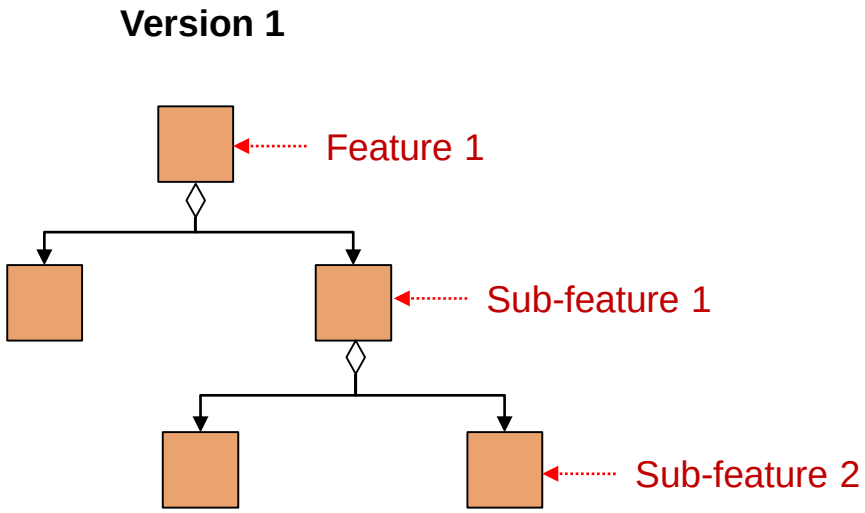
```
<cityObjectMember>
     <Building gml:id="B1020_t1">
            <identifier>B1020</identifier>
            <consistsOfBuildingPart>
                  <BuildingPart xlink:href="#BP12_t1"/>
            </consistsOfBuildingPart>
            <function>Office</function>
      </Building>
</cityObjectMember>
<cityObjectMember>
      <Building gml:id="B1020_t2">
            <identifier>B1020</identifier>
            <consistsOfBuildingPart>
                  <BuildingPart xlink:href="#BP12_t1"/>
            </consistsOfBuildingPart>
            <function>Living</function>
      </Building>
 </cityObjectMember>
 <cityObjectMember>
      <BuildingPart gml:id="BP12_t1">
            <identifier>BP12</identifier>
            <roofType>Flat</roofType>
      </BuildingPart>
 </cityObjectMember>
 <cityObjectMember>
      <BuildingPart gml:id="BP12_t3">
            <identifier>BP12</identifier>
            <roofType>Saddle</roofType>
      </BuildingPart>
 </cityObjectMember>
 <cityObjectMember>
      <Building gml:id="B1020_t3">
            <identifier>B1020</identifier>
            <consistsOfBuildingPart>
                  <BuildingPart xlink:href="#BP12_t3"/>
            </consistsOfBuildingPart>
            <function>Living</function>
      </Building>
</cityObjectMember>
```

Example 2 – Instance Data (Alternative Representation)

```
<cityObjectMember>
    <Building gml:id="B1020_t1">
        <identifier>B1020</identifier>
        <consistsOfBuildingPart>
            <BuildingPart xlink:href="//identifier[text()='BP12']"/>
        </consistsOfBuildingPart>
        <creationDate>2012-08-02</creationDate>
        <terminationDate>2013-10-09</terminationDate>
        <function>Office</function>
    </Building>
</cityObjectMember>
<cityObjectMember>
    <Building gml:id="B1020_t2">
        <identifier>B1020</identifier>
        <consistsOfBuildingPart>
            <BuildingPart xlink:href="//identifier[text()='BP12']"/>
        </consistsOfBuildingPart>
        <creationDate>2013-10-09</creationDate>
        <function>Living</function>
    </Building>
</cityObjectMember>
<cityObjectMember>
    <BuildingPart gml:id="BP12_t1">
        <identifier>BP12</identifier>
        <creationDate>2012-08-02</creationDate>
        <terminationDate>2014-06-03</terminationDate>
        <roofType>Flat</roofType>
    </BuildingPart>
</cityObjectMember>
<cityObjectMember>
    <BuildingPart gml:id="BP12_t3">
        <identifier>BP12</identifier>
        <creationDate>2014-06-03</creationDate>
        <roofType>Saddle</roofType>
    </BuildingPart>
</cityObjectMember>
```

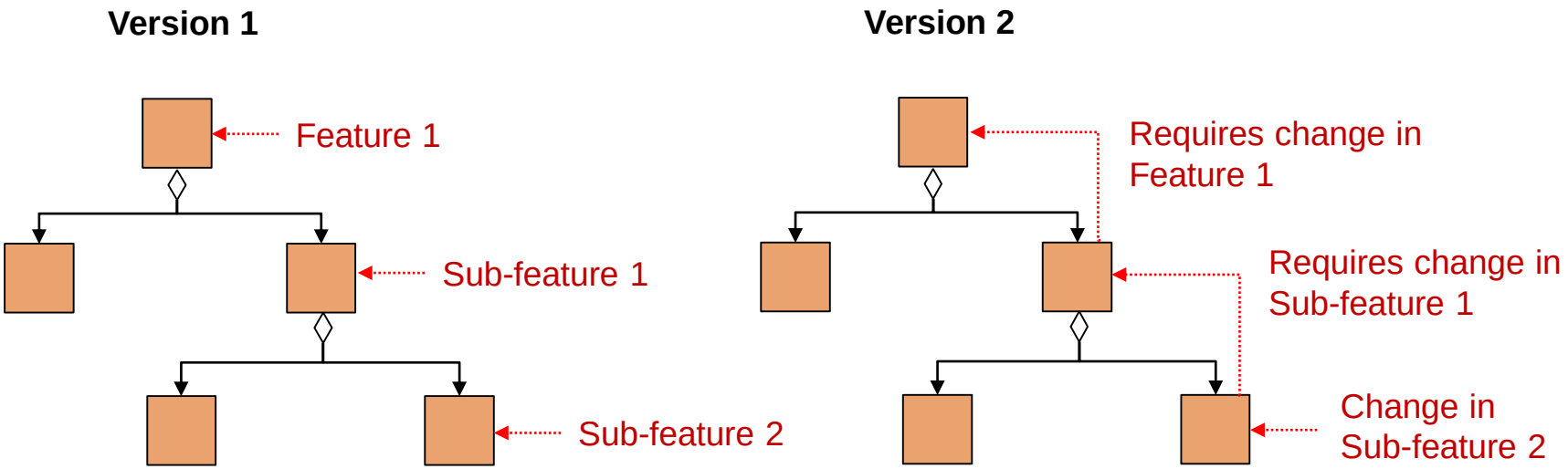
Versioning of aggregated features

- In CityGML, features can have aggregated sub-features
- For example, a Building feature can have _BoundarySurface features (e.g., WallSurface), which may further consist of sub-features such as Window or Door.



Versioning of aggregated features

- However, in order to create a new version including a change in any of the sub-features, the model would require changing all the parent features in the aggregation levels above.
  - reason: the aggregate object points to the part; if the part is replaced by a new version with a new gml:id, the pointer in the aggregate object also will have to be updated, creating a new object version also for the aggregate object (and so on)



- This issue can be resolved by referencing the Major ID attribute

Representation of the versions

- Versions need to be represented as objects with attributes
  - ID (Major ID)
  - Identifier (Major ID)
  - Name
  - Description
  - List of tags (user definable keywords)
  - CreationDate and TerminationDate (Mandatory)
    - Reflect database transaction time
  - ValidFrom and ValidTo (Optional)
    - Reflect the actual world time

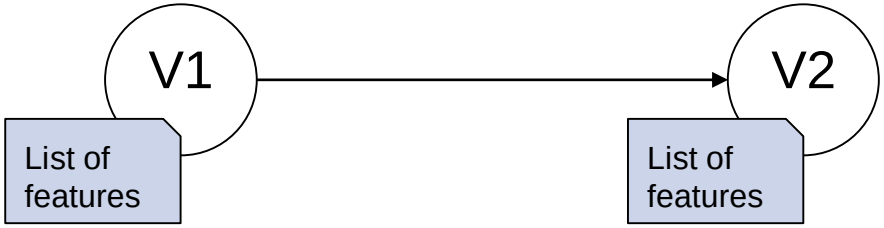
Should be the same as we don‘t want to have different versions of version objects

Snapshot / Change Approach

- To represent multiple versions of a city model, both Snapshot and Change approach can be supported.
- A snapshot is a representation of the state of all features of the entire city model at a specific point in time. It explicitly links to all objects in their version belonging to the respective city model version.

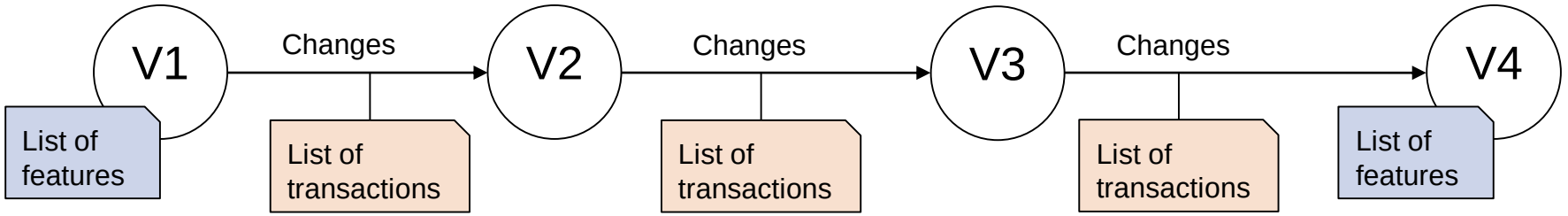
- **Scenario 1**: Snapshots, with no connection between versions

- **Scenario 2**: Snapshots, with explicit connection between versions (e.g. to express causality)



Combined Snapshot / Change Approach

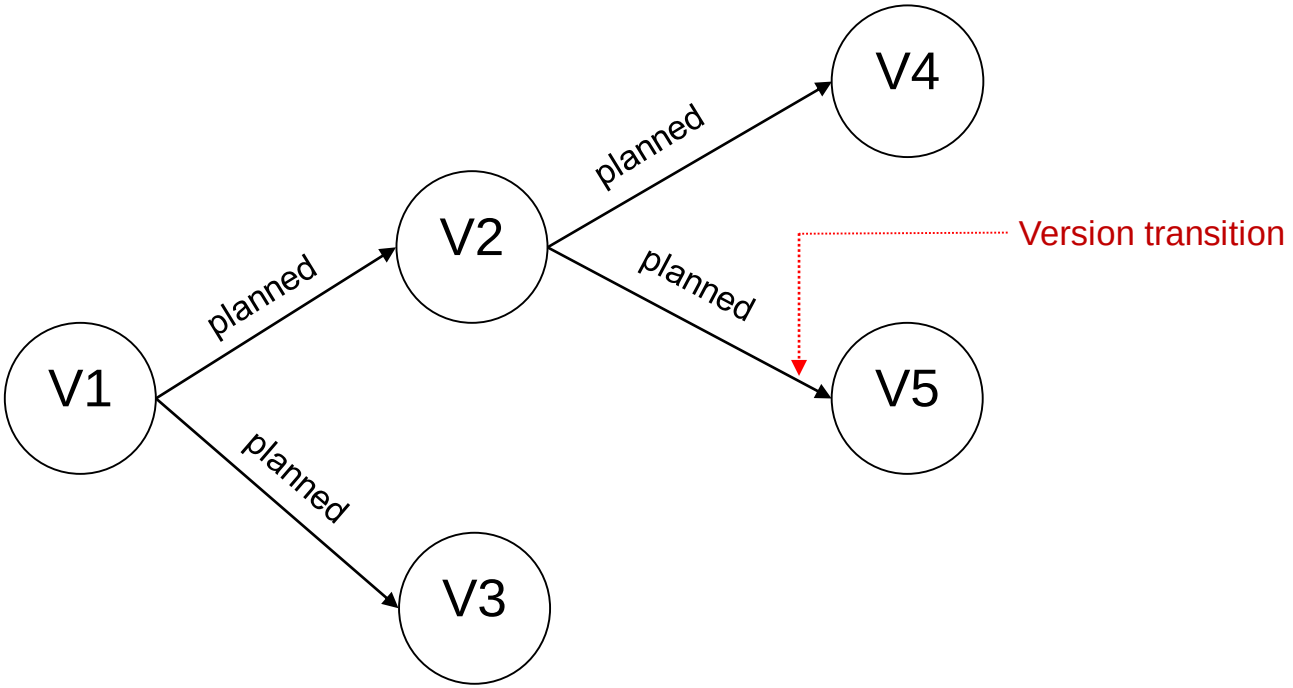
- **Scenario 3**: Snapshot + changes, with explicit connections, representing version transitions



- Advantages:
  - Low memory / storage requirements
  - Similiarity to full backup/incremental backup
  - May be used to stream dynamic changes
- Disadvantage:
  - The actual members of a version may be required to be determined by going back along the predecessors

Version transitions

- Version transitions express causal relations between versions

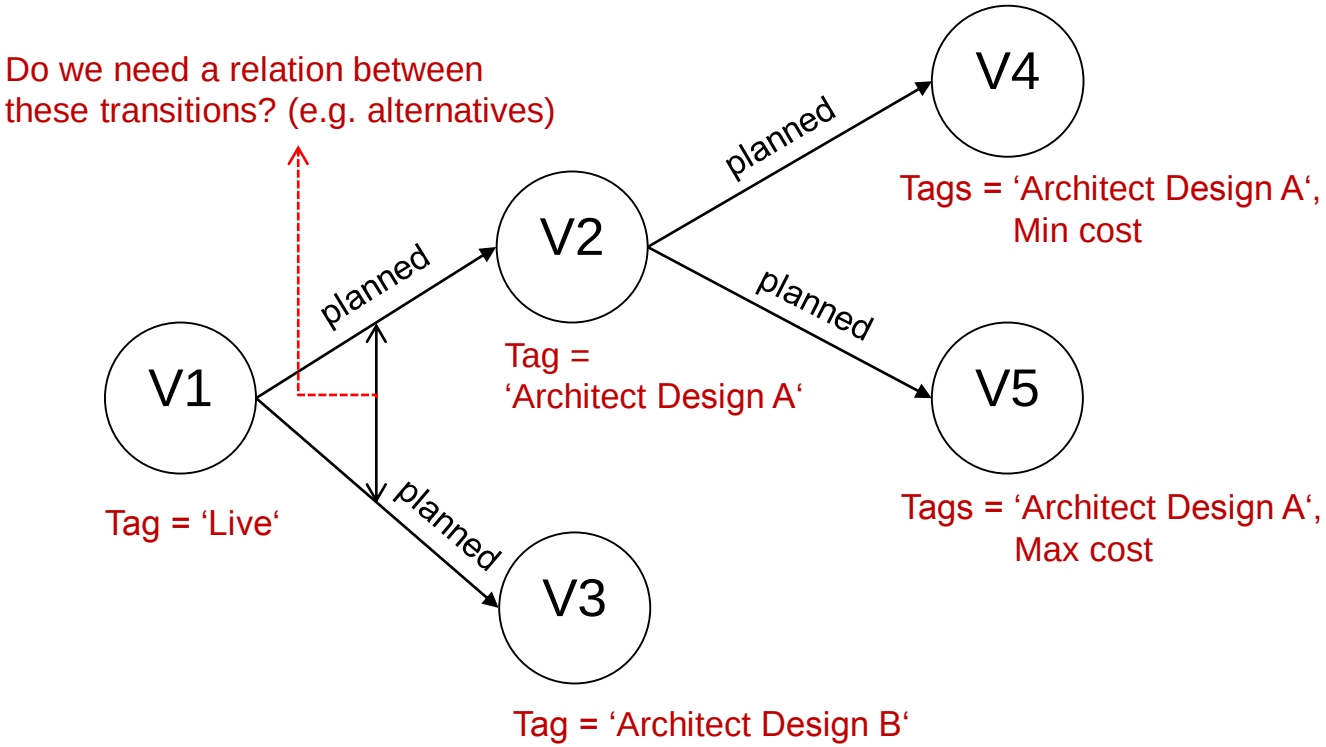


Representation of the version transitions

- Version transitions need to be represented with attributes
  - ID
  - Identifier
  - Name
  - Description
  - Reason
  - Indicator, whether the list of features is derived from a predecessor version
  - Type (planned, realized, historical succession, fork, merge)
  - List of updates/transactions from predecessor version (optional)

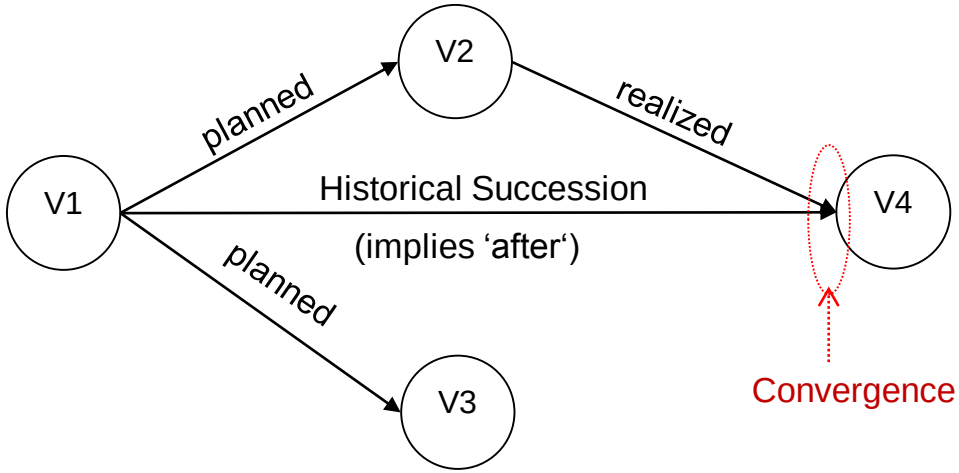
Version Graphs (Versions + Transitions)

- **Scenario 1**: Branching, with explicit relations between versions
- By using the “tag” attribute we could search for the version developed by a specific worker



Version Graphs (Versions + Transitions)

- **Scenario 2**: Convergence of version transitions
- The model should be capable of handling such version transitions

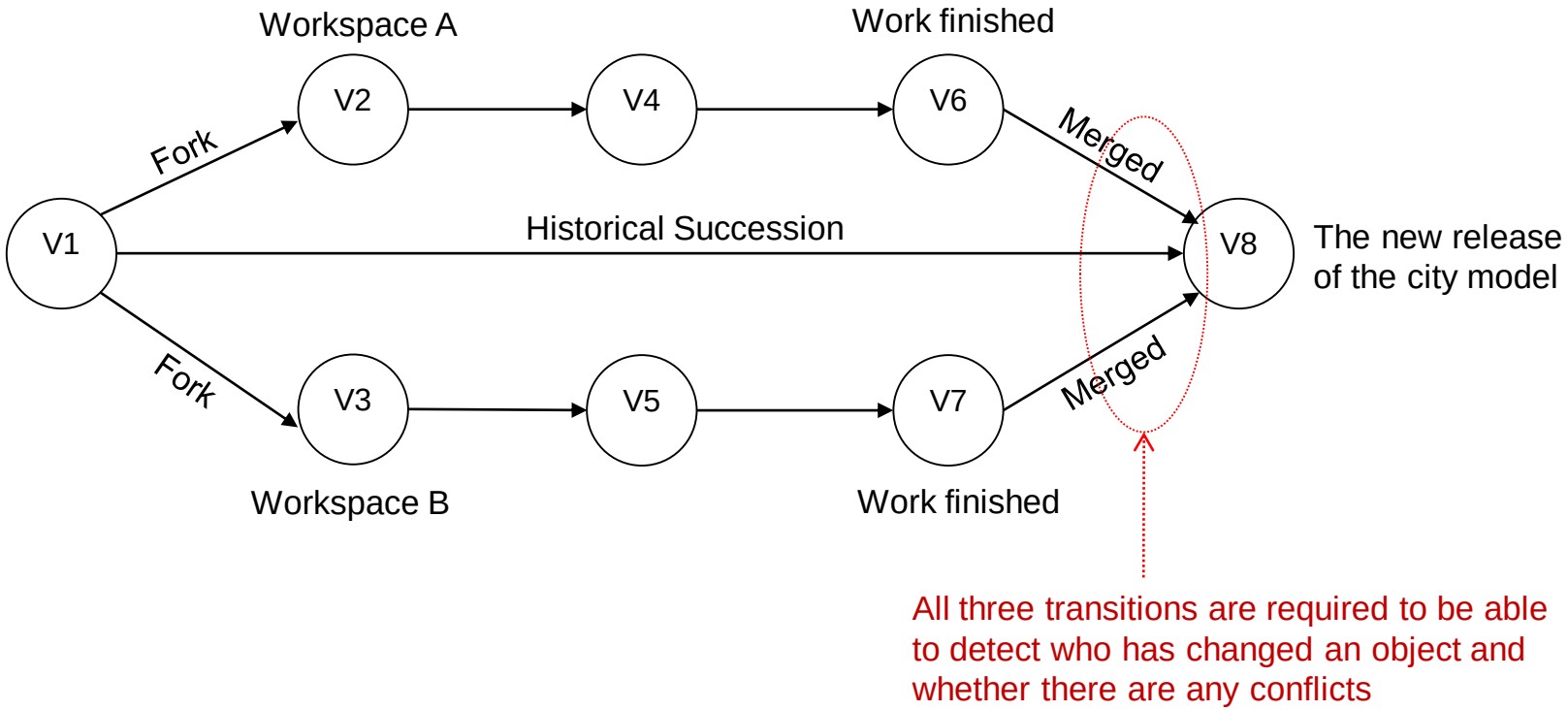


- For all convergence situations it must be ensured that the members of the converged version / state can be determined unambiguously

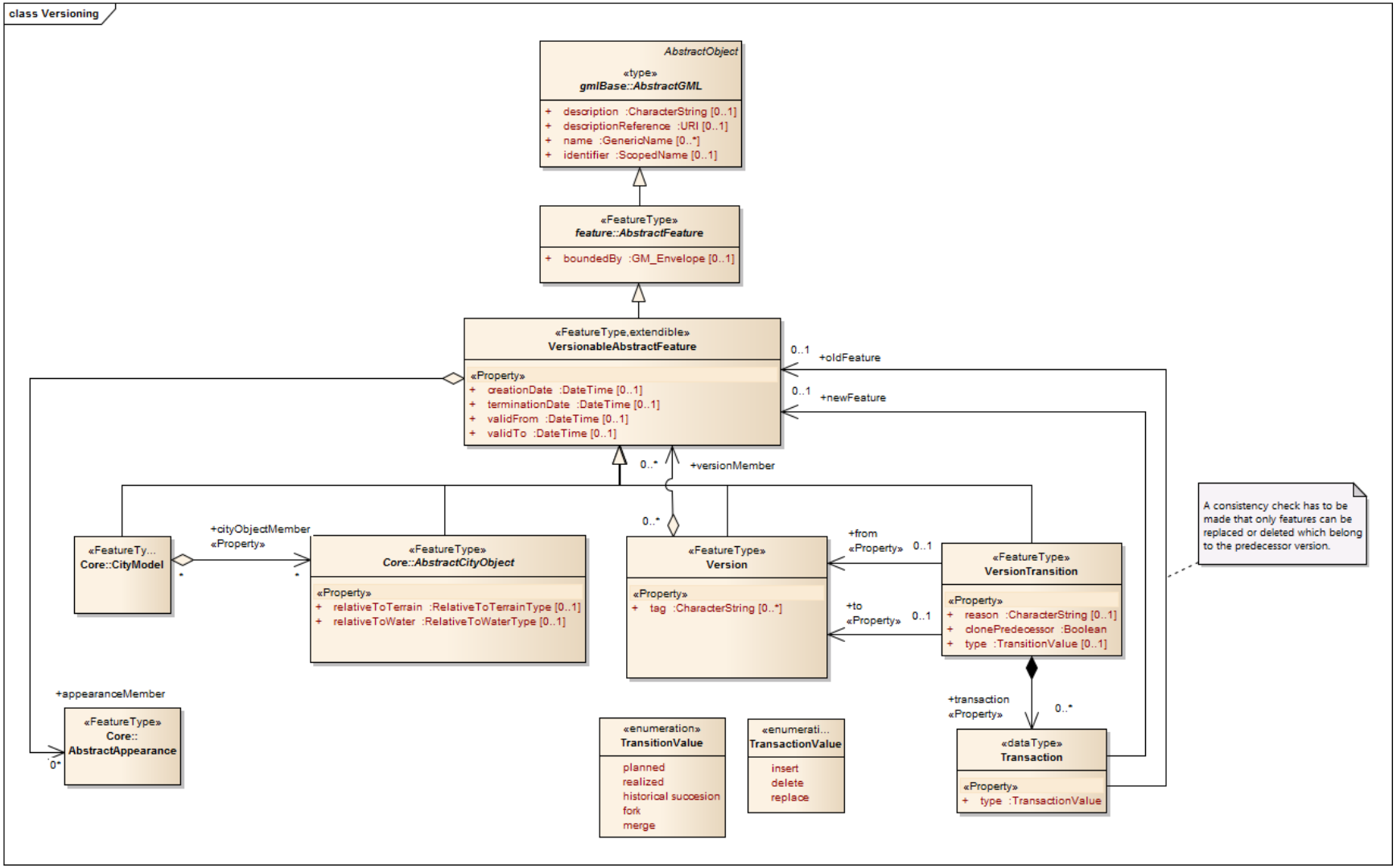
  → The easiest / safest way is to require that at maximum one of the incoming transitions has transactions

Example use case

- An example use case scenario, where the changes are made by different authorities or workers and merged into the new release of the city model.



UML Model – Versioning ADE



Remarks on the UML Model

- By modelling versions and version transitions as feature types, they can also be queried from a WFS
  - E.g. the “tag” attribute could be used to search for a version developed by a specific worker
- In GML 3.1.1 the identifier attribute is not defined in the class AbstractGML
  - But we can introduce it by an ADE element into the abstract class _CityObject

Suggestions / Recommendations

- Every version represents a consistent state of the real world.
- Consistency Rule: if a valid time is given for a feature, it must at least overlap with the time period of the version.
- Identifier attributes should be used with lifelong stable Ids
- Aggregated objects should point to the parts by referring to the identifier attribute
- If an object is being changed, the termination date should be set accordingly, and a new object is to be created with the same identifier.
- By referencing the identifier and not the GML id, we avoid that all parent objects in an aggregation would have to be updated, too.
- There are no specific requirements on a naming rule for GML ids, but there can be a recommendation to compose the GML id of the identifier, a separator character, and an encoded point in time from which this version of the object is valid (similar to INSPIRE and AAA, but needs to be checked).

Suggestions / Recommendations

- In order to support both, versioning and dynamic properties, we should have two modules: “Versioning” and “Dynamic properties”.
- An overloaded representation of the city where multiple versions of the city may be visible at the same time. However, the identifier attribute of all features allows to identify for which objects there are multiple representations.
- The GML id should be derived from the identifier value to support debugging and inspection

Crucial questions / discussions

- Should we make the versioning approach a best practice paper?
- Should Versions and VersionTransitions be versionable features (or more generally GML objects) and not CityGML objects?
- Should the version graph only contain every version once?
- Can every Feature (e.g. Addresses, Textures) be made versionable?
- Can geometry objects be made versionable?
- Can one version add data items from an ADE?
- Do we need to take special care on the UML modeling side, especially regarding the use of the identifier attribute and the referencing of Features by the identifier?
- Should we allow cycles in the version graph?
- Should we avoid that features with sub features refer to the definition of these sub features within another version of the main feature?

Crucial questions / discussions

- Can we have the same version object in different parts of the graph?
- Is it important to be able to create an empty version?
- Can our version concept be mapped to the native versioning concept of oracle, etc.?
- Is “version” the best word for our versioning concept?
  - Or are there better words such as “City state” or “city states model”?

References

- GML referencing identifiers
  - Example from Geography Markup Language (GML) (ISO 19136:2007):

    A reference to an object element in a remote XML document (or GML object repository) using the gml:identifier property value of that object may be encoded as:

    <myProperty xlink:href="http://my.big.org/test.xml#element (//gml:GeodeticCRS[./gml:identifier[@codeSpace="urn:x-ogc:def:crs:EPSG:6.3:"]="4326"])"/>

  - Blog entry from Simon Cox

    https://www.seegrid.csiro.au/wiki/AppSchemas/GmlIdentifiers#gml:identifier_element

Future work

- Review: How does the versioning work in the related standards / tools?
- To be prepared until the next telephone conference mid December 2014 by:
  - IFC – Gilles
  - ESRI Versioning Schema for ArcGIS – Kanishk
  - Oracle Workspace Manager – Kanishk
  - GIT/SVN – Steve
  - AAA/INSPIRE - Tatjana