CityGML

SIG 3D

Special Interest Group 3D

Modeling group

UtilityNetworkADE

- Core model -

- Draft version -

Authors: Thomas Becker, Claus Nagel, Thomas H. Kolbe

Department of Geodesy and Geoinformation

Berlin Institute of Technology

{thomas.becker | claus.nagel | thomas.kolbe }@tu-berlin.de

May 3, 2010

Version: Draft version 0.1.0, spelling and grammar unchecked

Content

1 Semantics

Graph

In mathematics [1], a graph is an abstract representation of a set of objects where some pairs of the objects are connected by links. The interconnected objects are represented by mathematical abstractions called vertices, and the links that connect some pairs of vertices are called edges. Typically, a graph is depicted in diagrammatic form as a set of dots for the vertices, joined by lines or curves for the edges.

_NetworkFeature

The actual topographic representation of a utility network object as for example tubes, lines, connecting pieces, switchgear cabinets, etc. *_NetworkFeature* is the base class for the thematic modeling of concrete utility networks. Therefore independent sub class hierarchies can be put into place directly below *_NetworkFeature* for the modeling of relevant objects in power, gas or water utilities.

Network

Network is a collection or arrangement of items (_NetworkFeature or NetworkGraph) to resemble a utility network.

FeatureGraph

A structure that represents a _NetworkFeature as set of nodes and edges.

NetworkGraph

A composition of linked real world objects to form a network.

Node

The dual representation of a real-world phenomenon as point-like object. For instance a terminal in a computer network. This documents distinct between interior and exterior nodes. An interior node is used to connect edges within the InteriorFeatureGraph, whereas the exterior node is used as port or connection to link different features to each other.

_Edge

The element which links two nodes to each other to form a FeatureGraph or NetworkGraph.

InteriorFeatureLink

The element which links the interior and exterior nodes of one _NetworkFeature to each other to form a FeatureGraph.

InterFeatureLink

The element which which connects the exterior nodes of two or more _FeatureGraph instances to form a NetworkGraph. The _FeatureGraph instances must belong to the same Network.

NetworkLink

The linking element to connect different NetworkGraphs to each other to form a Multi-Modal (multi-utility) Network.

2 Objectives

In this document a data model is presented (UtilityNetworkADE), which maps supplying infrastructures and utility networks (i.e. gas, water, electric, etc.) onto a CityGML Application Domain Extension (ADE) [2]. The data model is based on the work and discussions carried out by the modeling working group of the Special Interest Group 3D (SIG 3D). The concepts sketched in this working group were taken up and developed further, as well as mapped on a draft UML package and class diagram (see chapter 3).

The proposed data model can be understood specifically as a suggestion and basis for further discussions. The representations in this document are limited to the description of the core model and its relationships to additional packages. A discussion of the characteristics of specific utility networks can not to be carried out and is not part of this documentation.

3 Data model for the realization of utility networks

The UtilityNetworkADE is shown as an independent package in the centre of fig. 1. The sketched arrows of the package diagram mark the dependency between single packages. Dependency originates, e.g., when classes from a package are related to or derived from classes of another package. The direction of the arrow indicates the direction of dependency.

Accordingly, the UtilityNetworkADE shows a dependency to the CityGML Core module (blue marked) as well as to GML (green marked) which are shown both as independent packages in the upper part of the package diagram.

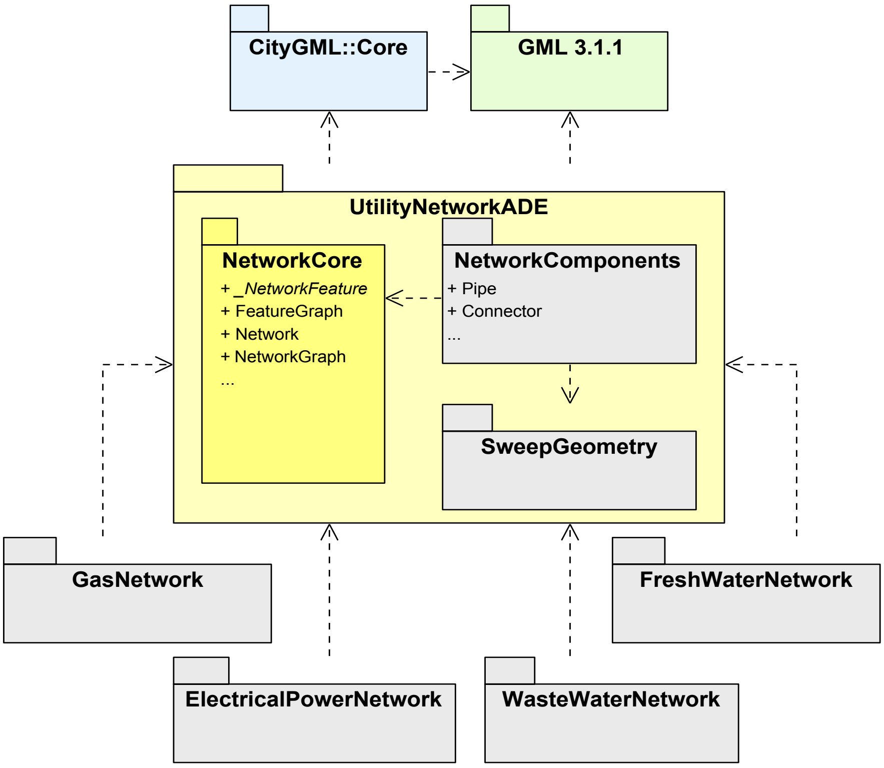


Fig. 1 Package diagram of UtilityNetworkADE

In the lower part of the figure additional packages can be found, which may represent concrete utility networks such as gas, power, freshwater, and wastewater. These packages are again dependent on UtilityNetworkADE package and its modeling elements. The list of these concrete utility networks is not complete – to describe and realize additional infrastructures or utilities the package diagram can be extended arbitrarily.

The identification of an independent UtilityNetworkADE package as well as a number of packages (describing concrete supply networks) which are dependent on it represents the first important modeling decision: A "core model" defines the UtilityNetworkADE and contains therefore exclusively generic model elements which are valid and suitable for the modeling of diverse supply networks. The modeling of specific peculiarities of a specific utility network then occurs in independent packages which refer to this generic utility core model.

The UtilityNetworkADE itself is separated in three sub packages:

- NetworkCore package - The package defines, on the one hand, the generic elements to the modeling of topographic network objects of a utility network, for example, to be able to represent pipes with its actual 3D object geometry within the city model. On the other hand the package contains a generic network model which allows the mapping of topographic network objects onto a graph structure which then can be used for network simulations and analyses.
- NetworkComponents package - This package defines different kinds of network objects which can be used to model various supply networks. For example, the package could contain some very general classes for the representation of a "Pipe" or a "Fitting". In the specific utility network packages these classes could be used directly or be specified further by derivation of sub classes. NetworkComponents is based on the generic elements of NetworkCore and, hence, is dependent on this package.
- ExtrusionGeometry package - This package should represent possible alternative geometry concepts which could be used to model 3d object geometry of network objects, but are not part of GML at the moment. For example, the geometry of a pipe can be often very simply described by using a Sweep solid or by using extrusion geometry. However, both geometry concepts are currently not supported by GML. Since the NetworkCore package would support these geometry concepts, a corresponding package dependency is shown in fig. 1. In order to be able to describe network objects such as pipes in a simple geometrical manner, the introduction of extended geometry concepts can be considered to the UtilityNetworkADE. Thus many tubing elements can be often described by simple extrusion or Sweep objects. In many cases the accurate description of geometries will even not be possible, because the position of underground structures are only be known approximately and can’t be measured directly. The ISO 19107 and likewise GML do not contain these or comparable geometry concepts in their current versions. However CityGML with its ImplicitGeometry provides a concept for modeling of scene graphs, which is also not a component of GML. Modeling proposals for extrusion objects are available and could be used to model components of utility networks. For the present data model from fig. 2 these concepts were not used, but will be considered in future work.

The modeling of NetworkCore package as a sub package of the higher UtilityNetworkADE package reflects the second essential modeling decision: The UtilityNetworkADE concept suggested with this document (still) does not intend to extend the Core module of CityGML by a generic, geometrical-topological network model which would then be valid in addition to utility networks for all other thematic CityGML modules. Rather this generic network model is deliberately moved in the NetworkCore package and thus realized as a component of the UtilityNetworkADE. Hence, it is beyond the CityGML core.

A migration into the CityGML core is still possible at a later time, and should take place only if the network model worked satisfactorily in the context of modeling supply networks. Eventually other packages - for example, those alternative geometry concepts could be also transferred in the CityGML core.

In the following sections the individual classes of the data model as well as their relations among each other are presented.

The following figure fig. 2 shows the developed data model in case of a UML class diagram for the realization of UtilityNetworkADE.

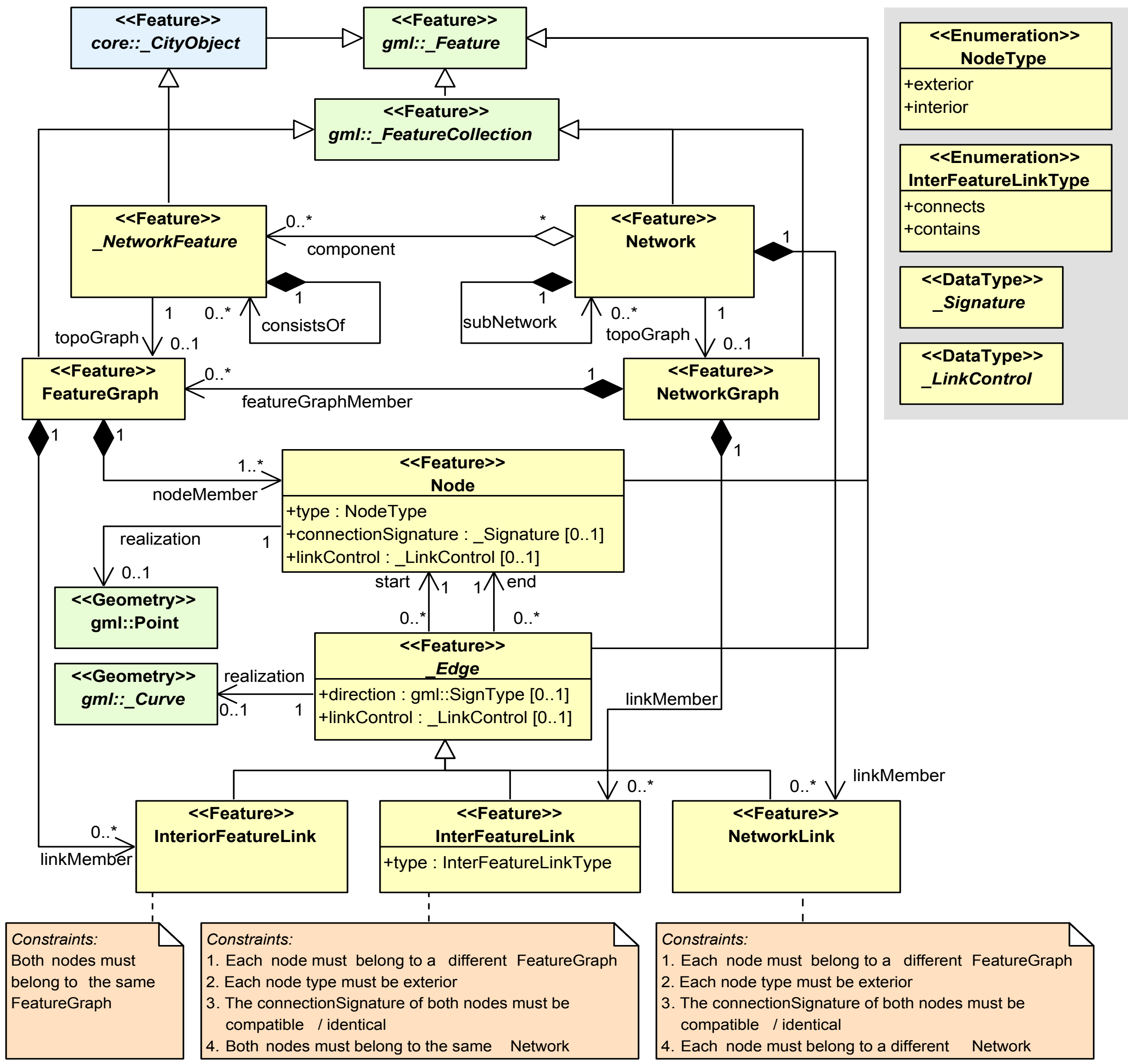


Fig. 2: UML class diagram for UtilityNetworkADE.

3.1 *_NetworkFeature*

The abstract class *_NetworkFeature* is one of the central classes within the data model of the UtilityNetworkADE. It represents the actual topographic objects of a utility network as for example tubes, lines, connecting pieces, switchgear cabinets, etc. Similar to the modeling in the existing thematic CityGML modules *_NetworkFeature* is derived from the abstract CityGML super class *_CityObject*. Thus a *_NetworkFeature* is a topographic city object within the city model and inherits all attributes and relations of *_CityObject [2]*.

_NetworkFeature represents the base class for the thematic modeling of concrete supply networks. Therefore independent sub class hierarchies take place directly below *_NetworkFeature* for the modeling of relevant objects in power, gas or water utilities and various other infrastructures. In the current data model *_NetworkFeature* can be understood as a placeholder. The development of appropriate class hierarchies for individual utility networks as well as the modeling of object relations within a utility network will be future work and will be published in future documents.

Like all city objects in CityGML *_NetworkFeature* is also a feature in terms of ISO 19109. According to this, arbitrary attributes like - spatial and non-spatial - can be modeled for *_NetworkFeature*. The spatial attributes serve for instance the mapping of actual 3D-object geometries in different level of detail. Already developed concepts of CityGML for the representation of upper and underground objects can be possibly reused, for example for the modeling of sewers.

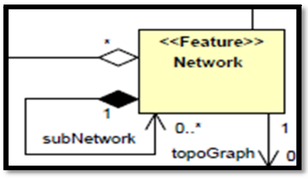
A *_NetworkFeature* can be composed of several *_NetworkFeature* objects. A switchgear cabinet within a power supply is for example itself a network object, at which other network objects can attach. At the same time a switchgear cabinet contains internal components, which again are components of the power supply network and, thus, are modeled as separate _NetworkFeature instances. The „is part of “- relationship between the internal components and the switchgear cabinet is represented in the data model by the *consistsOf* association of *_NetworkFeature*. Thus the possibility is given to model hierarchical components of a network [see further descriptions beneath InterFeatureLink.

Each *_NetworkFeature* can be represented by its own topological network. Just as a *_NetworkFeature* is a component of the **topographic representation of the entire supply network**, its representation through a topological network **is a component of the entire topological representation** of the supply network. In the simplest case a *_NetworkFeature* can be represented by a single node within the topological network. But however more complex sub graphs are also possible. The topological representation of a *_NetworkFeature* is modeled as instance of the class *FeatureGraph*, which is described in detail in section 3.3.

The suggested data model can be easily extended by the modeling of prototypical network objects. Thus network objects could be modeled in accordance with existing standards (e.g., pipe pieces according to suitable German Institute for Standardization-pipe mass). Then a suitable utility network could be built up by the instantiation of these prototypes at different places. Appropriate concepts are used for example in IFC. Even CityGML itself extends GML by the possibility of modeling of prototypical geometry objects (see. the class *ImplicitGeometry*). However, this does not enclose the modeling of semantic prototypes, which is addressed here. The topic „semantic prototypes “could represent a future extension of modeling utility networks.

3.2 Network

The class network is a further central element of the UtilityNetworkADE data model. It represents the topographic representation of an entire utility network as for instance gas, water or power supply and aggregates for this purpose a multitude of appropriate *_NetworkFeature* components (see the class CityModel as an aggregation of topographic city objects of a city model in CityGML). *Network* is derived from the abstract GML class *gml::_FeatureCollection*. Accordingly a network inherits the attributes for the modeling of a unique ID, as well as a description and a name. Further attributes are not yet intended in the actual draft version of this data model.



A utility network again can consist of individual sub networks. In the UML diagram the association *subNetwork* of the class *Network* with itself permits such a modeling. Thus the possibility exists to model an entire utility network as aggregation of different network types. In gas networks a clear separation in high pressure, medium pressure and low pressure can be done. The *Network* gas could be thus consisted and modeled as a collection of *Subnetwork* of high pressure, a *Subnetwork* medium pressure and low pressure.

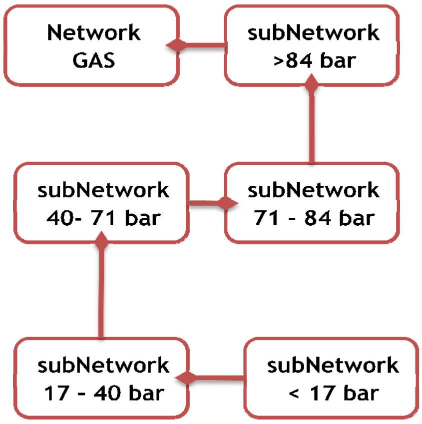


Fig. 3 Network Aggregation for utility Gas

The class Network is fully instantiable and non-abstract, because it must be possible to be able to connect two arbitrary networks. For example:

If we have a “PowerNetwork” and a “GasNetwork” and we want to link them to each other on a special place, for instance like a pumping station. The NetworkLink must be modeled in a higher “Network”, such like this:

```
<Network>
  <networkLink>
     <... xlink:href to NetworkFeature from PowerNetwork/>
     <... xlink:href to NetworkFeature from GasNetwork />
  </networkLink>
</Network>
```

Similar to the class *_NetworkFeature* a *Network* can be represented by a topological network. This topological network is represented in the data model as the class *NetworkGraph*. A *NetworkGraph* consists of individual *FeatureGraph* instances of those _NetworkFeature components, which are aggregated by a network. For a closer description of the class *NetworkGraph* see section 3.4.

3.3 FeatureGraph

The class *FeatureGraph* describes - just as *_NetworkFeature* - a network object of the utility network. However the class *FeatureGraph* represents a topological respectively functional view on the network object, whereas the class *_NetworkFeature* represents the topographical aspects of a network object (in particular its 3D-object geometry).

In its present version CityGML is a data model for the description of 3D topographic objects of a city. The class *_NetworkFeature* therefore binds the modeling of utility network to the topographic model of CityGML. The description of city objects as nodes and edges of a topological or functional network is not covered by CityGML so far. The possibility to model topological networks is given by the class *FeatureGraph*.

A *FeatureGraph* is exactly associated with one *_NetworkFeature* and, hence, describes exactly this *_NetworkFeature* by means of a graph structure. Feature graph is not intended to map the whole utility network (for this purpose the class *NetworkGraph* has to be used, cf. section 3.4). The modeling of nodes and edges follows the general principles of graph theory.

A _NetworkFeature is mapped by the class *FeatureGraph* onto a finite graph with an arbitrary set of nodes and edges. Possible further requirements to this graph structure (i.e. whether the graph has to be connected) have to be examined. In the following fig. 2 two alternative graph representations are shown for one single *_NetworkFeature*. The network feature in this example represents a t-fitting.

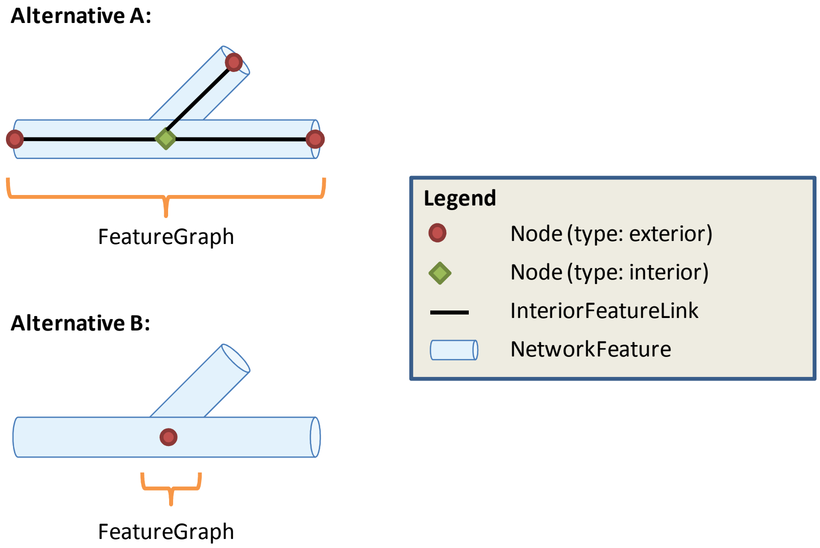


Fig. 4: two alternative graph representations for one single *_NetworkFeature*

Alternative A of fig. 4 results from a rather functional description of the T-fitting. All external connection points of the T-fitting, at which other network objects of the utility networkcan be connected to the T-fitting, are represented as separate nodes (shown as red nodes) in *FeatureGraph*. In order to be able to map the internal structure of the T-fitting, all external connection nodes must be linked by edges. This requires the modeling of a further, internal node (shown as green node). Contrary to the external connection nodes an internal node must not be linked to other network objects of the *Network*.

In alternative B the entire T-fitting is represented as a single node. This is the minimum representation of a *FeatureGraph*. Therefore it’s not possible to map a *_NetworkFeature* onto a single edge without mapping additional nodes, because an edge must always be bounded by exactly two nodes (see data model in fig. 2).

Alternative A as well as alternative B are valid *FeatureGraph* representations of the T-fitting. Further alternatives are possible. For example additional internal nodes may be inserted, which reflect the change of functional, physical or other arbitrarily values of a *_NetworkFeature*. This additional information can be used for analyses and simulations. As future research task, different generalization levels of the network representation can be identified (comparable to the existing LOD concept of CityGML) for which strict modeling rules can be specified.

The nodes and edges of a *FeatureGraph* form a purely topological network without geometrical or metrical information. Nevertheless the *FeatureGraph* can be embedded in geometric space. This permits for instance the assignment of (3D) point coordinates to nodes or permits the computations of edge lengths. Thus, further application fields are set up for *FeatureGraph*.

The modeling of nodes and edges of a *FeatureGraph* is realized in accordance to the data model presented in fig. 2 by using the classes *Node* and *InteriorFeatureLink* (sub class of _Edge). Their detailed description can be found in section 3.5. The class *Node* as well as *InteriorFeatureLink* are derived from the abstract GML class gml::*_Feature*. Hence, the class *FeatureGraph* is modeled as sub class of gml::*_FeatureCollection*.

3.4 NetworkGraph

Similar to the modeling of *_NetworkFeature* and *FeatureGraph*, the class *NetworkGraph* represents the topological view on the entire utility network described by the class network. *NetworkGraph* therefore links the individual *FeatureGraph* instances to a topological network.

If two network objects are connected, this can be expressed in a *NetworkGraph* by an edge between the associated *FeatureGraph* instances. The edge is modeled by the class *InterFeatureLink* and connects exactly two external connection nodes. Both external connection nodes must belong to different *FeatureGraph* instances. Thus the InterFeatureLink differs from *InteriorFeatureLink*, which is only used to connect nodes within the same FeatureGraph (see. section 3.3 as well as section 3.6.1). Furthermore an *InterFeatureLink* may only be modeled between two external connection nodes, if both nodes are compatible to each other. For more information on this constraint, see section 3.6.2.

In summary a *NetworkGraph* consists of a set of *FeatureGraph* instances as well as of *InterFeatureLink* objects, which aggregate the *FeatureGraph* instances to a topological network. The following Fig. 5 gives an example for two connected components (a t-pipe and a pipe) of a water supply network. For a better visualization of the *InterFeatureLink* the pipes are spatially separated from each other. The both in section 3.3 presented alternatives for a *FeatureGraph* are as well suitable to model a *NetworkGraph*.

In addition to the modeling of network topology the *NetworkGraph* can be also embedded geometrically. In the first place, this requires of course the geometrical embedding of individual *FeatureGraph* representations (see. section 2.3). In variation A of Fig. 5 a geometrical embedding of the *InterFeatureLink* is not necessary, since both connection nodes of the respective pipes collapse geometrically and thus the *InterFeatureLink* does not have an additional spatial representation. However in variation B, the *InterFeatureLink* can be assigned its own geometry, which introduces for example the information about edge lengths to *NetworkGraph*. . The section 3.6.2 contains further remarks to the class *InterFeatureLink*.

3.5 Node

The class *Node* describes a node within the topological graph of one *_NetworkFeature*. Hence, nodes may only be modeled for a *FeatureGraph* (and not for NetworkGraph). Each *FeatureGraph* may contain an unbounded number of nodes, but however it must consist of at least one node. Each node is again associated to exactly one FeatureGraph. Thus, a single node may not be a part of two or more *FeatureGraph* instances.

The class Node is directly derived from *gml::_Feature* and represents a feature in terms of ISO 19109. Thus the use of predefined GML topology classes is renounced.

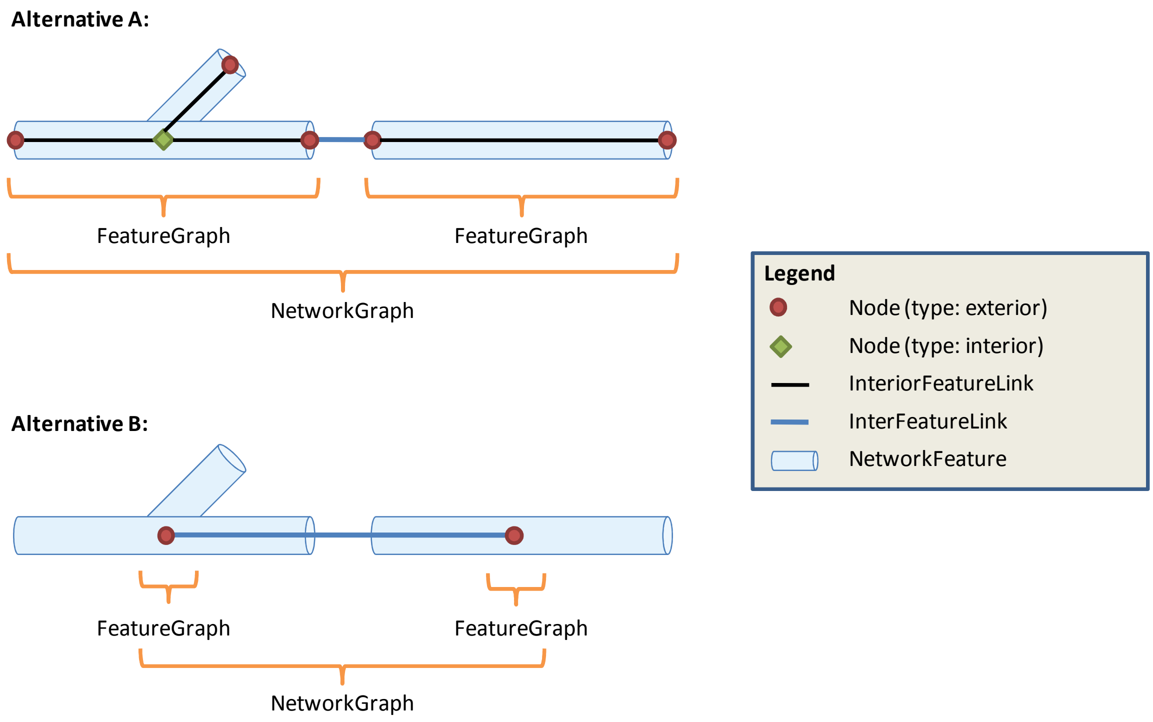


Fig. 5: Two alternative modeling variations of a NetworkGraph, which consists of *Feature-Graph* instances.

The suggested data model from fig. 2 distinguishes two fundamental types of nodes: **internal nodes** and **external connection nodes**. The distinction is made by the attribute type of the class *Node*, which can take an appropriate value of the code list *NodeType*. External nodes represent connection points, where other network objects can be attached to the *_NetworkFeature*. In contrast, internal nodes are used to describe the internal structure of the *_NetworkFeature*. A *FeatureGraph* may contain an arbitrary number of internal nodes, for instance to model the change of functional, physical or other values within the *NetworkFeature*.

For a Node further user-specific attributes can be modeled. For example, the pipe diameter could be assigned to the external connection node at the connection point of the pipe, or however, the medium which is expected at this connection point. An internal node could be used, for example, to represent the maximum internal pressure at a certain location within the pipe.

Another attribute of the class *Node* is *connectionSignature*. The aim of this attribute is the definition of a connection signature for a node. Two nodes may only be linked, if their connection signature allows this connection. This means the connection signature of the both nodes must be identically or at least compatible. The connection signature is primarily used for external connection nodes. If for instance the connection node of a pipe indicates a specific pipe diameter, then a second pipe, which shall be attached to this node, must have the same pipe diameter (which again is modeled as connection signature for the external node of the second pipe). This condition can be expressed by using the attribute *connectionSignature*. Thus additional examinations and analyses of the network are possible.

A *Node* can have a geometrical representation. This is realized by the realization association, to *gml::POINT*.

3.6 *_Edge*

The abstract class *_Edge* is used to model an edge within a topological graph. An *_Edge* must be limited by exactly two nodes (an startNode and an endNode, which induce a direction). Like *Node* the class *_Edge* is directly derived from *gml::_Feature*. An *_Edge* can be embedded in geometric space, which is realized through its realization association to gml::_Curve. Additionally some user-specific attributes can be modeled for the class *_Edge*.

The data model differentiates 3 fundamental kinds of edges, which are namely the sub classes *InteriorFeatureLink, InterFeatureLink, and NetworkLink*. Special constraints apply to these edge types, which are discussed in more detail in the following sections.

Within the model we use the inheritance relation between the superclass *_Edge* and its child classes *InterFeatureLink*, *InteriorFeatureLink* and *NetworkLink,* to define edges which are only permitted in certain sub graphs. Basically, each of these edges can be directed or undirected. If a class *DirectedLink* would be derived from _Edge, we would have to fall back on multiple inheritances to represent this.

We decided against the multiple inheritances. Hence, we have modeled the edge direction as an attribute of _Edge. For an undirected edge the *direction* is not modeled (see, direction : gml::SignType [0..1]). A directed edge has always a *direction*. The direction is given (like in GML3) by "+" or "-". The "+" (default) means the direction goes from the start node to the end node, the "-" direction assigns the other way round. That's why the class *_Edge* has two associations to *Node* with suitable role names.

A special attribute of *_Edge* is *linkControl*. *linkControl* allows for the modeling of, for example, switches within the supply network, which can interrupt the flow of a medium, such as water. By using the abstract class _LinkControl one is able to model in specific utility network packages, specific control objects, such as switches. This will take part in future documentations and is not implemented yet.

3.6.1 InteriorFeatureLink

An *InteriorFeatureLink* describes an edge between two nodes, both of which must belong to the same FeatureGraph instance. Thus the class *InteriorFeatureLink* may only be modeled for the class *FeatureGraph* (and not for *NetworkGraph*) and may be used to describe the internal structure of a *_NetworkFeature*. An InteriorFeatureLink may connect both external and internal nodes within the same FeatureGraph.

3.6.2 InterFeatureLink

The *InterFeatureLink* links feature graph instances of two *_NetworkFeature* and, thus, represents the connection between two network objects within the topological graph structure of the utility network. The following constraints apply for *InterFeatureLink*:

1. An *InterFeatureLink* can only be modeled between two Nodes which belong to different feature graph instances (and therefore represent different network objects [*_NetworkFeature*]).
2. The *Node* type of the both linked Nodes must be exterior, i.e. only external connection nodes may be connected by an InterFeatureLink.
3. The connection nodes of both feature graph instances may only be linked to each other through an *InterFeatureLink* iff their connection signature (attribute: connectionSignature from _Edge) is identical or at least compatible.

Using the attribute *type* (data type: *InterFeatureLinkType*) of *InterFeatureLink* two more special types can be distinguished: the connections of network objects at the same hierarchy level as well as the connection of network objects at different hierarchy level. These hierarchy levels directly correspond to the modeling of aggregation hierarchies between *_NetworkFeature* (cf. already mentioned in segment 3.1).

The following fig. 6 represents an example of modeling different types of *InterFeatureConnection*. It shows a switchgear cabinet (blue cube), which is modeled as a *_NetworkFeature* and part of the power network. The switchgear cabinet has two external connection nodes (type: exterior), which can be used to attach further network objects of the power network. The switchgear cabinet contains further internal components (yellow cubes), which constitute its actual functionality. These individual parts are also modeled as a *_NetworkFeature* and therefore they are represented by an own feature graph instance.

The „is-part-of "- relationship of the internal components to the switchgear cabinet is mapped on the level of topographic objects by using the self-association *consistsOf* of *_NetworkFeature*.

The linkage of the respective topological feature graph instances takes place via the modeling of an *InterFeatureLink* between two connection nodes. Within the example each internal component is connected to exactly one connection node of the switchgear cabinet. In order to illustrate the aggregation relationship in a topological network between the switchgear cabinet and it internal components, the connecting *InterFeatureLink* is assigned the value "contains" for its "type". Furthermore, both internal components are connected through an InterFeatureLink at their external connection nodes. Since both components are on the same hierarchy level, the value connects (blue lines in fig. 5) is assigned to the attribute "type" of the corresponding *InterFeatureLink* otherwise if they are on different hierarchy level the value contains (orange lines in fig. 5) is assigned to the attribute "type" of the corresponding *InterFeatureLink*.

The identifier *connects* and *contains* of the code list InterFeatureLinkType may not be confused with the topological relations of Egenhofer.

3.6.3 NetworkLink

The class *NetworkLink* enables the linkage of two *NetworkGraph* with different *NetworkFeature*. In order to build up a multi-modal or interdependent network model different network types such as power supply, fresh water, and waste water must be able to be linked together. This can be realized by using the class *NetworkLink*, which is the linking element to connect different NetworkGraph's at common connecting points. A pump for instance is part of a fresh water network as well as of power supply network, thus a *NetworkLink* must be created with a node representing the pump in Power supply network and a node representing the pump in fresh water network as boundary. Please see the example below for a better understanding of *NetworkLink*.

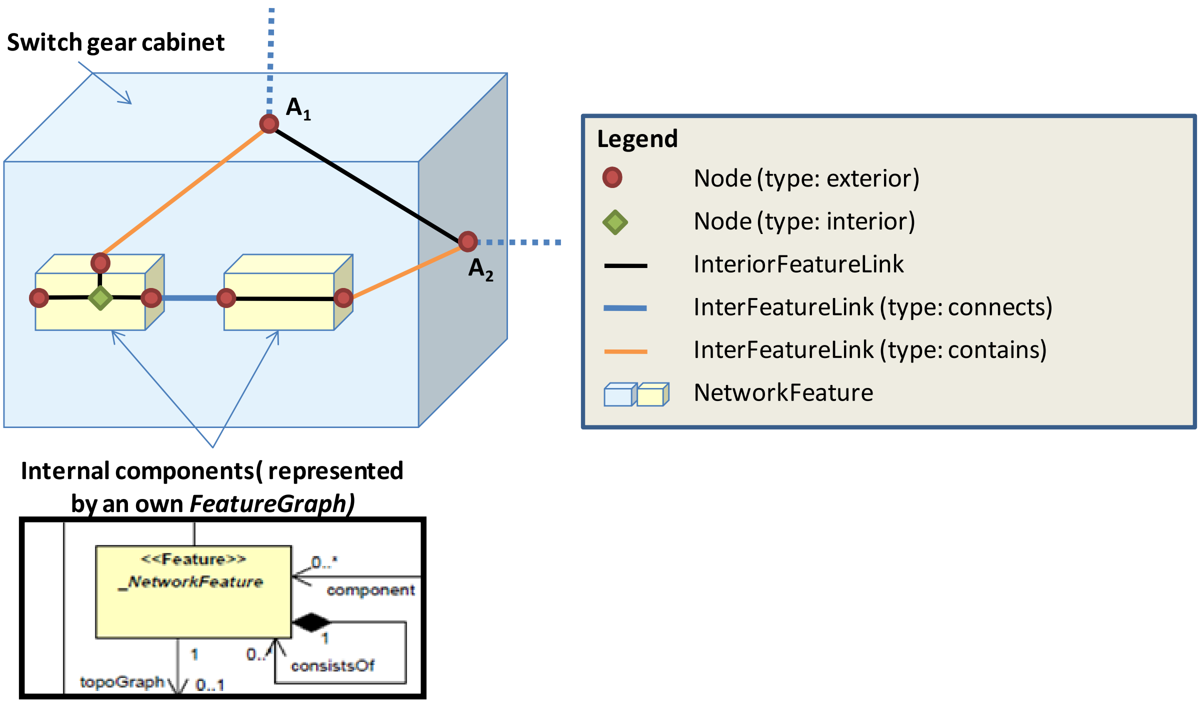


Fig.. 6: modeling of different hierarchy levels by using the attribute "type" of InterFeatureLink

4 Simple Example

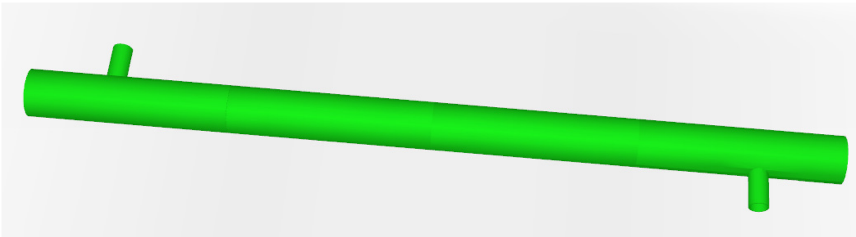


Fig. 7 Simple example pipe

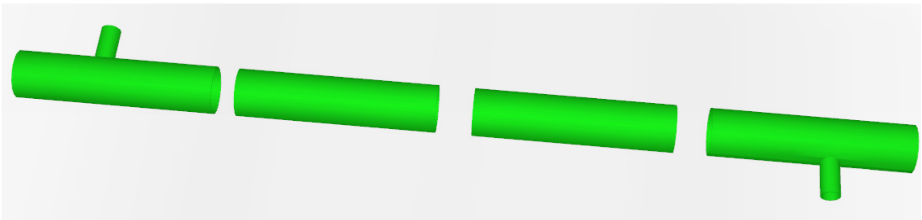
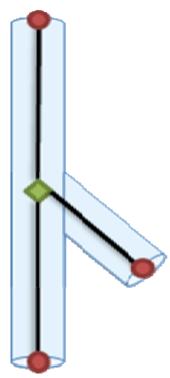


Fig. 8 example pipe consisting of 2 t-pipes and 2 normal pipe pieces

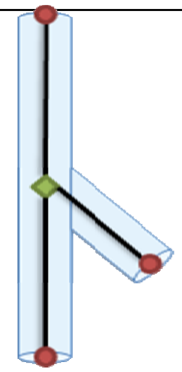
In this example we would like to show the different ways of modeling a network or especially one of its component. The first three components will be modeled as a graph while the last one will be modeled as a point shape object. Therefore the *InterFeatureLink* connecting the third and the last component of our example pipe will have geometry. Because it realizes the distance from endpoint of component 3 to a representative point of component 4 (in this case the middle point).

```
<?xml version="1.0" encoding="UTF-8"?>
<CityModel                    xmlns="http://www.opengis.net/citygml/1.0"                    xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:util_core="http://www.citygml.org/ade/utility/core/0.1.0"
xmlns:util_comp="http://www.citygml.org/ade/utility/networkComponents/0.1.0" xmlns:gml="http://www.opengis.net/gml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.citygml.org/ade/utility/networkComponents/0.1.0
../UtilityNetworkADE/utility_network_components.xsd">
  <cityObjectMember>
    <util_core:Network>
      <util_core:type>Water</util_core:type>
      <util_core:networkFeatureMember>
        <util_comp:Section gml:id="MainPipe">
          <util_core:consistsOf>
            <util_comp:MainLine gml:id="TPipeSection1">
              <util_core:featureGraph>
                <util_core:FeatureGraph>
                  <util_core:nodeMembers>
                    <util_core:Node gml:id="StartTPipeSection1">
                      <util_core:type>exterior</util_core:type>
                      <util_core:geometry>
                        <gml:Point gml:id="StartTPipeSection1Point">
                          <gml:pos> -3.96198 -0.223113 0</gml:pos>
                        </gml:Point>
                      </util_core:geometry>
                    </util_core:Node>
                    <util_core:Node gml:id="EndTPipeSection1">
                      <util_core:type>exterior</util_core:type>
                      <util_core:geometry>
                        <gml:Point gml:id="EndTPipeSection1Point">
                          <gml:pos> 6.03802 -0.223113 0</gml:pos>
                        </gml:Point>
                      </util_core:geometry>
                    </util_core:Node>
                    <util_core:Node gml:id="StartTPipeTSection1">
                      <util_core:type>interior</util_core:type>
                      <util_core:geometry>
                        <gml:Point gml:id="StartTPipeTSection1Point">
                          <gml:pos> 0.086927 -0.223113 0</gml:pos>
                        </gml:Point>
                      </util_core:geometry>
                    </util_core:Node>
                    <util_core:Node gml:id="EndTPipeTSection1">
                      <util_core:type>exterior</util_core:type>
                      <util_core:geometry>
                        <gml:Point gml:id="EndTPipeTSection1Point">
                          <gml:pos> 0.086927 2.72877 0</gml:pos>
                        </gml:Point>
                      </util_core:geometry>
                    </util_core:Node>
                  </util_core:nodeMembers>
                  <util_core:interiorFeatureLinkMembers>
                    <util_core:InteriorFeatureLink>
                      <util_core:start xlink:href="#StartTPipeSection1"/>
                      <util_core:end xlink:href="#StartTPipeTSection1"/>
                      <util_core:geometry>
                        <gml:LineString>
                          <gml:pos> -3.96198 -0.223113 0</gml:pos>
                          <gml:pos> 0.086927 -0.223113 0</gml:pos>
                          <gml:pointProperty xlink:href="#StartTPipeSection1Point"/>
                          <gml:pointProperty xlink:href="#StartTPipeTSection1Point"/>
                        </gml:LineString>
                      </util_core:geometry>
                    </util_core:InteriorFeatureLink>
                    <util_core:InteriorFeatureLink>
                      <util_core:start xlink:href="#StartTPipeTSection1"/>
                      <util_core:end xlink:href="#EndTPipeTSection1"/>
                      <util_core:geometry>
                        <gml:LineString>
                          <gml:pos> 0.086927 -0.223113 0</gml:pos>
                          <gml:pos> 0.086927 2.72877 0</gml:pos>
                          <gml:pointProperty xlink:href="#StartTPipeTSection1Point"/>
                          <gml:pointProperty xlink:href="#EndTPipeTSection1Point"/>
                        </gml:LineString>
                      </util_core:geometry>
                    </util_core:InteriorFeatureLink>
                    <util_core:InteriorFeatureLink>
                      <util_core:start xlink:href="#StartTPipeTSection1"/>
                      <util_core:end xlink:href="#EndTPipeSection1"/>
```

TPipeSection1

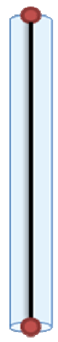


```xml
                <util_core:geometry>
                    <gml:LineString>
                        <gml:pos> 0.086927 -0.223113 0</gml:pos>
                        <gml:pos> 6.03802 -0.223113 0</gml:pos>
                        <gml:pointProperty xlink:href="#StartTPipeTSection1Point"/>
                        <gml:pointProperty xlink:href="#EndTPipeSection1Point"/>
                    </gml:LineString>
                </util_core:geometry>
            </util_core:InteriorFeatureLink>
        </util_core:interiorFeatureLinkMembers>
      </util_core:FeatureGraph>
    </util_core:featureGraph>
  </util_comp:MainLine>
</util_core:consistsOf>
<util_core:consistsOf>
  <util_comp:MainLine gml:id="PipeSection1">
    <util_core:featureGraph>
      <util_core:FeatureGraph>
        <util_core:nodeMember>
            <util_core:Node gml:id="StartPipeSection1">
                <util_core:type>exterior</util_core:type>
                <util_core:geometry>
                    <gml:Point gml:id="StartPipeSection1Point">
                        <gml:pos> 6.03802 -0.223113 0</gml:pos>
                    </gml:Point>
                </util_core:geometry>
            </util_core:Node>
        </util_core:nodeMember>
        <util_core:nodeMember>
            <util_core:Node gml:id="EndPipeSection1">
                <util_core:type>exterior</util_core:type>
                <util_core:geometry>
                    <gml:Point gml:id="EndPipeSection1Point">
                        <gml:pos> 16.03802 -0.223113 0</gml:pos>
                    </gml:Point>
                </util_core:geometry>
            </util_core:Node>
        </util_core:nodeMember>
        <util_core:interiorFeatureLinkMember>
            <util_core:InteriorFeatureLink>
                <util_core:direction>+</util_core:direction>
                <util_core:start xlink:href="#StartPipeSection1"/>
                <util_core:end xlink:href="#EndPipeSection1"/>
                <util_core:geometry>
                    <gml:LineString>
                        <gml:pos> 6.03802 -0.223113 0</gml:pos>
                        <gml:pos> 16.03802 -0.223113 0</gml:pos>
                        <gml:pointProperty xlink:href="#StartPipeSection1Point"/>
                        <gml:pointProperty xlink:href="#EndPipeSection1Point"/>
                    </gml:LineString>
                </util_core:geometry>
            </util_core:InteriorFeatureLink>
        </util_core:interiorFeatureLinkMember>
      </util_core:FeatureGraph>
    </util_core:featureGraph>
  </util_comp:MainLine>
</util_core:consistsOf>
<util_core:consistsOf>
  <util_comp:MainLine gml:id="PipeSection2">
    <util_core:featureGraph>
      <util_core:FeatureGraph>
        <util_core:nodeMember>
            <util_core:Node gml:id="StartPipeSection2">
                <util_core:type>exterior</util_core:type>
                <util_core:geometry>
                    <gml:Point gml:id="StartPipeSection2Point">
                        <gml:pos> 16.03802 -0.223113 0</gml:pos>
                    </gml:Point>
                </util_core:geometry>
            </util_core:Node>
        </util_core:nodeMember>
        <util_core:nodeMember>
            <util_core:Node gml:id="EndPipeSection2">
                <util_core:type>exterior</util_core:type>
                <util_core:geometry>
                    <gml:Point gml:id="EndPipeSection2Point">
                        <gml:pos> 26.03802 -0.223113 0</gml:pos>
                    </gml:Point>
```

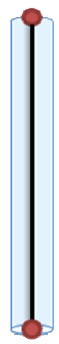


TPipeSection1

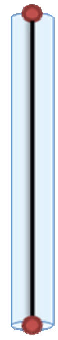
PipeSection1



PipeSection2

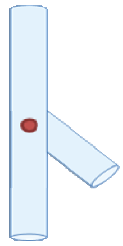


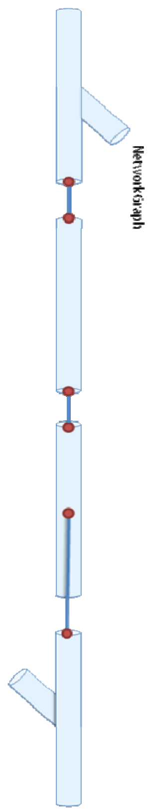
```
                        </util_core:geometry>
                    </util_core:Node>
                </util_core:nodeMember>
                <util_core:interiorFeatureLinkMember>
                    <util_core:InteriorFeatureLink>
                        <util_core:direction>+</util_core:direction>
                        <util_core:start xlink:href="#StartPipeSection2"/>
                        <util_core:end xlink:href="#EndPipeSection2"/>
                        <util_core:geometry>
                            <gml:LineString>
                                <gml:pos> 16.03802 -0.223113 0</gml:pos>
                                <gml:pos> 26.03802 -0.223113 0</gml:pos>
                                <gml:pointProperty xlink:href="#StartPipeSection2Point"/>
                                <gml:pointProperty xlink:href="#EndPipeSection2Point"/>
                            </gml:LineString>
                        </util_core:geometry>
                    </util_core:InteriorFeatureLink>
                </util_core:interiorFeatureLinkMember>
            </util_core:FeatureGraph>
        </util_core:featureGraph>
    </util_comp:MainLine>
</util_core:consistsOf>
<util_core:consistsOf>
    <util_comp:MainLine gml:id="TPipeSection2">
        <util_core:featureGraph>
            <util_core:FeatureGraph>
                <util_core:nodeMember>
                    <util_core:Node gml:id="MidPointTPipeSection2">
                        <util_core:type>exterior</util_core:type>
                        <util_core:geometry>
                            <gml:Point gml:id="MidPointTPipeSection2Point">
                                <gml:pos> 31.1628 -0.223113 0</gml:pos>
                            </gml:Point>
                        </util_core:geometry>
                    </util_core:Node>
                </util_core:nodeMember>
            </util_core:FeatureGraph>
        </util_core:featureGraph>
    </util_comp:MainLine>
</util_core:consistsOf>
</util_comp:Section>
</util_core:networkFeatureMember>
<util_core:networkGraph>
    <util_core:NetworkGraph>
        <util_core:interFeatureLinkMembers>
            <util_core:InterFeatureLink>
                <util_core:start xlink:href="#EndTPipeSection1"/>
                <util_core:end xlink:href="#StartPipeSection1"/>
                <util_core:type>connects</util_core:type>
            </util_core:InterFeatureLink>
            <util_core:InterFeatureLink>
                <util_core:start xlink:href="#EndPipeSection1"/>
                <util_core:end xlink:href="#StartPipeSection2"/>
                <util_core:type>connects</util_core:type>
            </util_core:InterFeatureLink>
            <util_core:InterFeatureLink>
                <util_core:start  xlink:href="#EndPipeSection2"/>
                <util_core:end xlink:href="#MidPointTPipeSection2"/>
                <util_core:geometry>
                    <gml:LineString>
                        <gml:pos> 26.03802 -0.223113 0</gml:pos>
                        <gml:pos> 31.1628 -0.223113 0</gml:pos>
                    </gml:LineString>
                </util_core:geometry>
                <util_core:type>connects</util_core:type>
            </util_core:InterFeatureLink>
        </util_core:interFeatureLinkMembers>
    </util_core:NetworkGraph>
</util_core:networkGraph>
</util_core:networkGraph>
</util_core:Network>
</cityObjectMember>
</CityModel>
```

PipeSection2

TPipeSection2





5 Comments

5.1 Directed edges

This point has released the most discussions inside the modeling group. In many models directed and undirected edges are represented by an inheritance relation. Therefore the object type defines whether it concerns a directed or undirected edge.

Within our model we use the inheritance relation between the superclass *_Edge* and its child classes *InterFeatureLink*, *InteriorFeatureLink* and *NetworkLink*, to define edges which are only permitted in certain sub graphs. Basically, each of these edges can be directed or undirected. If a class *DirectedLink* would be derived from _Edge, we would have to fall back on multiple inheritances to represent this.

We decided against the multiple inheritances. Hence, we have modeled the edge direction as an attribute of _Edge. For an undirected edge the *direction* is not modeled (see, direction : gml::SignType [0..1]). A directed edge has always a *direction*. The direction is given (like in GML3) by "+" or "-". The "+" (default) means the direction goes from the start node to the end node, the "-" direction assigns the other way round. That's why the class *_Edge* has two associations to *Node* with suitable role names.

5.2 Generalization of Switches and Valves etc.

We decided against the possibility to represent every possible kind of flow control by concrete classes within the core model. Rather every utility networkoffers its own kinds of switches which can be modeled in special network models as an extension of this core model.

Therefore this draft version of the core model inherits merely the abstract data type *_LinkControl* from which then in special modules the respective switches can be derived. As also done by *direction* we assigned *_LinkControl* as an attribute to *_Edge* and avoid the problems of the multiple inheritances. Therefore every edge of the graphs can be directed or be undirected as well as act as a counter.

Furthermore a *Node* inherits also the attribute _LinkControl. At least, it is possible to represent a *NetworkFeature* about a single node (without edges). If this *NetworkFeature* is a switch or something similar, the information about the flow control must be assigned to the node.

6 References

[1] Diestel, Reinhard: Graph Theory – Electronic Edition 2005, *in: Graduate Texts in Mathematics, vol. 173*. Springer-Verlag, 2005.

[2] Gerhard Gröger, Thomas H. Kolbe, Angela Czerwinski and Claus Nagel: OpenGIS® City Geography Markup Language (CityGML) Encoding Standard. Version: 1.0.0, OGC 08-007r1, http://www.opengeospatial.org/standards/citygml

[3] Becker, T., Nagel, C., Kolbe, T. H.: A Multilayered Space-Event Model for Navigation in Indoor Spaces. In: *Lee, Jiyeong / Zlatanova, Sisi (Hg.): Proceedings of the*

3rd International Workshop on 3D Geo-Information, Seoul, Korea. Lecture Notes in Geoinformation & Cartography. Springer Verlag, Seoul 2008.

[4] Becker, T., Nagel, C., Kolbe, T. H.: Supporting Contexts for Indoor Navigation using a Multilayered Space Model. In: *2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware*. IEEE Computer Society, Taipei, Taiwan, ISBN: 978-0-7695-3650-7, 2009, S. 6.