How to link models with different grids (spatial mapping)

Spatial Mapping

Hydrological models usually have a schematization or grid that represents the spatial resolution of the model. For groundwater models, regular or non-regular two- or three-dimensional grids may be used, whereas river models typically use a one-dimensional grid. Conceptual catchment models may use a closed polygon to describe the catchment boundary.

When models with different spatial schematizations are linked, the values associated to one schematization in the source model must be transformed to be represented on the schematization of the target model. In order to make such transformations possible for any combination of models a standardized representation, the ISpatialDefinition interface has been introduced in the TimeSpace extension of the OpenMI Standard 2 (see Figure below). The type of spatial reference is defined by a the Wkt (well-known text) string as it is common in the OGC compliant GIS world, s. [http://spatialreference.org](http://spatialreference.org). The version property allows dynamical spatial constructions, e.g. following the track of a ship or the eye of a hurricane. The IElementSet interface, known from previous versions, is now an extension to ISpatialDefinition.

The two-dimensional and three-dimensional ElementTypes have been melted in OpenMI Standard 2, e.g. the 2D XYPoint and the 3D XYZPoint have been melted to Point with an optional z value. The ElementType PolyLine represents now all line constructions.

Spatial mapping only works with geo-referenced ElementTypes, meaning all types apart from IdBased. A two-dimensional grid for a groundwater model and a one-dimensional grid for a river model are shown in the figure below.
The groundwater model grid consists of four elements GE1, GE2, GE3 and GE4. These elements have the type Polygon. Each corner of these polygons is a Vertex, and each Vertex has Co-ordinates. The type Polygon is defined in the standard in the enumeration ElementType, whereas there are no interface definitions for Vertex and Co-ordinate; these are only part of the OpenMI terminology and are used in the naming of some of the methods in the IElementSet interface.

The grid for the river model can be represented by an implementation of the IElementSet interface, where each branch is an element of type PolyLine. Each PolyLine element will have two vertices, one at each end of the line. The elements for the river model are shown in figure above as RE1, RE2 and RE3.

Assume that the river model and the groundwater model illustrated in the figure above are linked for the purpose of transferring information about groundwater leakage from the river model to the groundwater model. When the groundwater model invokes the GetValues method in the river model in order to obtain the leakages, the river model must return a ITimeSpaceValueSet, where each value represents the leakage that enters each element in the groundwater model for the requested time. With this in mind it may seem like a huge task to implement the GetValues method in the linkable component. However, since all spatial representations can be accessed generically through the IElementSet interface, a generic element mapper could be developed. Such a mapper is provided in the open source OpenMI SDK with the namespace Oatc.OpenMI.Sdk.Spatial.

The two most essential methods in the ElementMapper class are Initialise and MapValues:

```java
public void Initialise(IIdentifiable method, IElementSet fromElements, IElementSet toElements)

public TimeSpaceValueSet MapValues(ITimeSpaceValueSet inputValues)
```

If Initialise is called at the beginning, a mapping matrix will be calculated for mapping the fromElements to the toElements. The mapping method is specified by method, a string of type IIdentifiable.

During computation MapValues uses the mapping matrix calculated in Initialise. When the GetValues method is invoked, the source component uses this matrix to make the spatial conversion, simply by multiplying the vector of input values associated to the grid of the source components with the matrix.

\[
A = \begin{pmatrix}
1 & 1/3 & 0 \\
0 & 2/3 & 1/2 \\
0 & 0 & 0 \\
0 & 0 & 1/2
\end{pmatrix} \tag{eq.1}
\]

For the example shown in the figure above, the mapping matrix will look as shown below:

When the groundwater model invokes the GetValues method in the river model, the river model can make the spatial transformation of its internal calculated leakages using the following multiplication:

\[
I = I \times A \tag{eq.2}
\]

where I is a vector with four components describing the leakage contribution to each grid cell in the groundwater model and L is a vector with three components, each value being the calculated leakage in a river branch.
A hard-coded example of the coupling of a river and a groundwater model is the unit test LinkingRiverGWTest.cs in the Oatc.OpenMI.Wrappers.EngineWrapper.UnitTest namespace. (C#-Code on SourceForge).

**Adapted outputs**

In OpenMI 2 spatial mapping can be done in a flexible way by plugging a spatial Adapted output between Provider and Consumer. A set of simple interpolations will be provided and application developers can integrate their own Adapted outputs with new interpolation methods. Furthermore, a series connection of Adapted outputs is possible.

The figure above displays a series connection of Adapted outputs mapping the Quantity H for water level from a grid to two single points. In a first step the ElementSet Grid is converted to two ElementSets of type Point. The second step is a further interpolation from these points to the single ElementSets Point1 and Point2.

The user can interactively define the order the Adapted outputs are run, optimizing the mapping performance.