What's New in OpenMI 2.0

Summary:

This document describes the new concepts in OpenMI 2.0, compared with OpenMI 1.4

Contact:

rvm@ceh.ac.uk
www.openmi-life.org

Version:

V1.0

Date:

30/10/2010

Status:

Final

1 Introduction

This document describes the new concepts in OpenMI 2.0, compared with OpenMI 1.4, and their elaboration in the class design.

The new features of OpenMI 2.0 can be summarized as:

- A more flexible way of linking
- More flexibility in the overall control flow
- Less difference between spatial and temporal models
- Support in categorized data values
- Extensions processing new types of components

2 Base Interfaces and extensions

OpenMI 2.0 has become more versatile. Whereas OpenMI 1.4 was mainly restricted to models that progress in time, OpenMI 2 offers a set of base interfaces that are not aware of the type of a model. These interfaces can easily be extended, e.g. with the extension for time and space dependent components towards models that progress in time. Alternatively they can be externded to models that express the exchange items and their values in terms of ontologies, instead of in terms of spatial and time scale definitions.

This implies that an OpenMI compliant component can now comply to the base interfaces (and indeed has to), but that it can also comply to one or more of these extension interfaces. The classical 1.4 compliant components would have to implement the base interfaces as well as the extension for time and space dependent components in order to become OpenMI 2.0 compliant.

The current version of this document (July 2010) specifies the base interfaces and the extension supporting the time and space dependent component. Future extensions could support parallel computing, compliance with the standards of the Open Geospatial Consortium OGC, and more. Implementing several extensions can combine the features of these extensions, e.g. a time and space dependent component that can run in parallel with other components (Figure 1, right side)

Deleted: beta

Deleted: 31

Deleted: 07

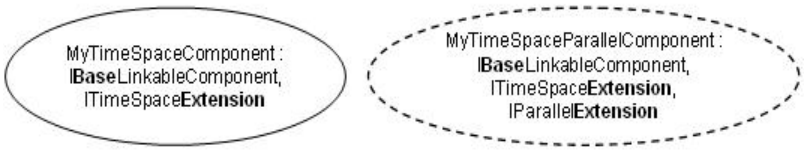

Figure 1: OpenMI 2.0 compliant components implementing uncoupled interfaces; the one on the left side is already feasible, the other one a forecast

It is recommended to check the website www.openmi.org for information about the coming extensions.

3 Linking components

In OpenMI 1.4 the link object was used to connect components, containing a reference to the source component and the target component, and to the source output (exchange) item and the target input (exchange) item. Any operations in between the output item and the input item, to make them 'compatible', was handled by a series of DataOperations, also specified by the link object (Figure 1).

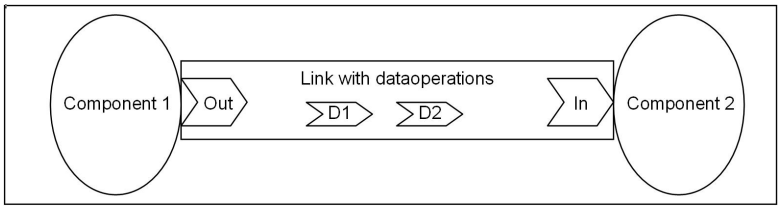

Figure 2 Linking components in OpenMI 1.4

In OpenMI 2.0 a link between a source component and a target component is realized by a direct reference between an output item and an input item. The link, as a separate object, has been removed from the specification. If the values of an output are not in the exact form in which they are needed, you can 'adapt' an output. This opens the possibility to add additional data operations in between the original output and input item, realizing a true piping and filtering pattern.

The adapted outputs take over the role of data operations in OpenMI 1.4. Typically,,such intermediate adapted outputs

are spatial and time interpolations and unit conversions. In the sequence of adapted outputs, the order in which such adapting operations are carried out is defined explicitly. Also, the possible data flows are extended, because each output and adapted output may be reused and connected to several requesting input items.

In Figure 2 below there are three adapted outputs added using the same output item, and an input item can connect using any of the four relations marked by the arrow-lines.

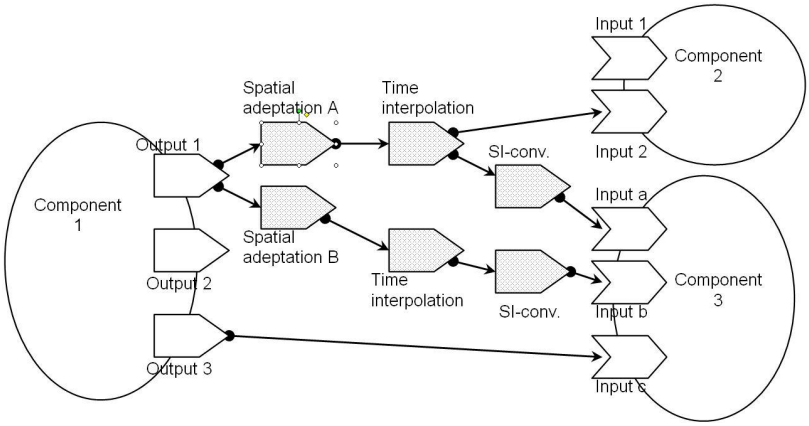

Figure 3 Linking components in OpenMI 2.0

To implement this, the GetValues call of the linkable component has moved to the output item. The data operation is gone and instead, an adapted output, which is an output item, is introduced. The adapted output is configurable, in order to control operations such as unit conversion and interpolation. It also performs data exchange between components with different extensions, meaning the adapter has to be aware of the concepts in both extensions.

The adapted output has a Refresh() method, which is used to notify an adapted output that the values of its parent output may have been changed.

Whereas in OpenMI 1.4 the requested timestamp or time span was the main argument in the GetValues() call, in OpenMI 2.0 the argument has become a full 'query specification'. Of course, the requested timestamp or time span can still be provided, but now also the requested element set can be specified, so that OpenMI 2.0 is fitter for use with GIS systems. This refers to the extension for time and space dependent components.

4 Requesting values and control flow

In OpenMI 1.4, a linkable (model) component was asked for values for a certain time for a particular link. The component had the responsibility to return values meeting the requirements specified through the data definition in the link: basically, the target quantity and target element set. If the calculation required input from other components, those would be triggered as well (the pull-based chain computation approach).

In OpenMI 2.0, the concept of the link has been removed. The data definition is now passed in the GetValues() call. For the classical time and space dependent components this call contains the requested timestamps or time spans, element set and value type. (Note that multiple timestamps and time spans are supported.)

Further extensions may use different arguments in the GetValues() call. The designated extension supporting ontologies will use indexes to specify the exchange data. If components with different extensions to the standard transfer data, it is recommended to develop an adapted output, performing the necessary conversions.

4.1 Calibration and data assimilation support

Development of calibration, optimization and data assimilation tools for OpenMI-compliant models is possible with version 1.4 of the standard. However, the pull-driven architecture sometimes makes this complicated.

In OpenMI 2.0 the data exposed for exchange is a precise reflection of the data inside the component. Therefore, version 2 supports the setting of values before running a component, thus allowing the use of the OpenMI in calibration, optimization, data assimilation and decision support systems.

4.2 Component status

Linkable components implement a property status. This property describes the status of the component at any time. Figure 3 is a diagram of the possible changes between different states.

The status can be used to check what a component is actually doing, which is relevant for the outer world (e.g. de GUI), but also for the component itself. If one of the component's output items receives a GetValues() call when the component is in the 'Waiting for data' state, this means that the GetValues() call was self-imposed: i.e. it is the result of a feedback loop.

Deleted: In OpenMI 1.4, the control flow was purely pull driven, since linkable components were triggered to perform an action when there was a request to produce data.¶
In OpenMI 2.0 this is still the same but a designated extension offers the loop-approach, an alternative control flow with an external controller. ¶

Deleted: could

Deleted: in the loop approach ('Waiting for data' versus 'Updated) as well as in the pull driven approach.

Deleted: (A

Deleted: on an output

Deleted: its

Deleted:)

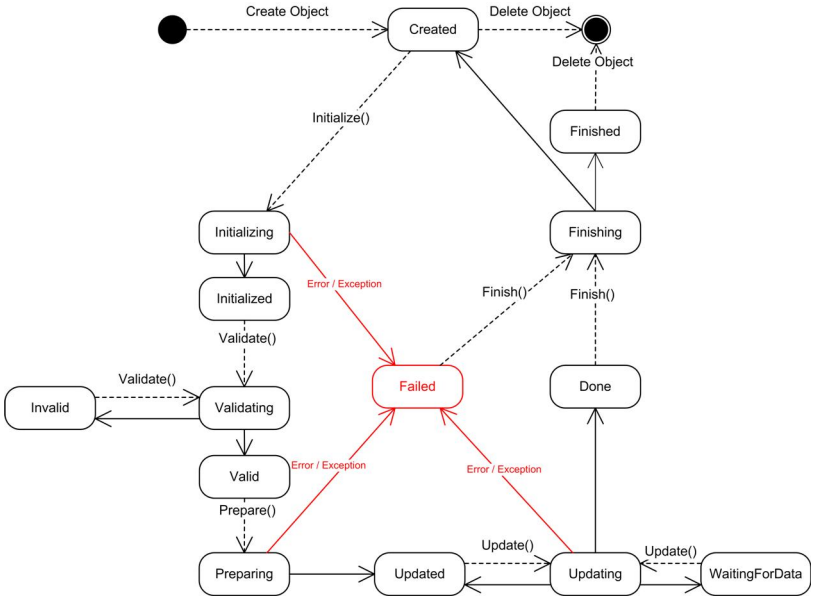

Figure 4 Linkable component status

4.3 Less focused on time stepping models

The OpenMI 1.4 Standard was developed specifically with the linkage between numerical time-stepping models in mind. However, in integrated modelling systems other components such as data providers, databases and viewers are also used.

Therefore OpenMI 2.0 provides the flexible extension for time and space dependent components. The various types of models can be accessed as unified a way as possible:

- Varying in time or in space
- Varying in time but not varying in space
- Varying in space but not varying in time

4.4 Events

Version 1.4 had its own event system. For version 2 the standard .NET and Java mechanisms are applied in the Standard.

5 Data definitions

5.1 Quantity and Quality

Values in OpenMI 1.4 could only be quantitative, but OpenMI 2.0 adds support for qualitative information (e.g. land use types or indications such as ‘hot’ and ‘cold’ or ‘more sustainable’ and ‘less sustainable’). The standard also allows custom types of values.

5.2 Element set

The element set definition was expanded to follow the OpenGIS standard. The distinction between element sets in the horizontal plane and the three-dimensional space (e.g. XYPoint / XYZPoint) has been removed, so geo-referenced items are now positioned on a Point, a Polyline or a Polygon, while a HasZ property indicates whether the vertical dimension is involved or not. Also, M-coordinates (linear referencing) have been introduced.

The ISpatialReference interface has been replaced by the ISpatialDefinition interface. Due to the latter an element set has now a string that defines the spatial reference. This string follows the OGC standard WKT (well known text) for spatial reference.

5.3 Time set

Handling of time has been made similar to the handling of the element set. An exchange item now has a time set, specifying for which timestamps or time spans the item can provide values (output) or wants to retrieve values (input).

5.4 ValueSet

In OpenMI 1.4, the ValueSet could contain double-precision scalars, or vectors consisting of double-precision X/Y/Z-components, with one value for each element. A ValueSet instance applied to one timestamp or time span.

In OpenMI 2.0 the values are organized in a two-dimensional array (times and elements) and the stored values can be objects of any type. The most common type is the double-precision real value but there are many quantities that can be expressed by an integer value, a boolean value or a user-defined type such as a vector. Another type now supported is categorized data, used to represent qualitative information.