OpenMI Association Technical Committee meeting no 28

See also: OATC Wiki Home

Date: June 8 - 9, 20010
Venue: Unesco-IHE, The Netherlands

Participants:

Rob Knapen, Alterra, Wageningen UR (Rob.Knapen@wur.nl)
Standa Vanecek, DHI (s.vanecek@dhi.cz)
Stef Hummel, Deltares (stef.hummel@deltares.nl)
~jgr@dhigroup.com, DHI (jgr@dhigroup.com)
Peter Schade, Bundesanstalt fuer Wasserbau (peter.schade(at)baw.de)

Apologies:
Unknown User (jnh@dhigroup.com), DHI (jnh@dhigroup.com)
Daniele Andreis,Universita` di Trento,(daniele.andreis@gmail.com)
Jan Gregersen, LicTek

Documents:

Table of contents

1. Comments from review

Consider support for parallel computations

This is a candidate for an extension to the component, see section "Possible candidates for extensions". We will try to make the standard such that it ease the development of such an extension.

Make clear how use IElementSet to work with OGC formats

Two alternatives were discussed:

- Document how each OGC feature is to be used by the current elementset.

- Make a more general element set, and let the current element set extent from the general one. And open for the possibility for other extensions of the general one, being e.g. the different OGC feature classes.

The discussion turned into which strategy was the best for OpenMI: To document specialized interfaces, or to have general ones and allow them to be extended over time.

Add initialize method to IAdaptedOutput

Adding Initialize() to IAdaptedOutput, similar to the method of the ILinkableComponent 👍 .

See minutes of meeting 26 - the "IArgument.IsReadOnly" section - describes that the Initialize should not have any arguments. Though, that has not been implemented, and may produce some work, especially for the GUI. We will investigate the amount of work, and may leave it with the arguments in. In any case we will assure that the two have the same Initialize method signature.

Re-introduce the IState interface

Discussion turned first to whether the IIdentifiable should extend from the IDescribable, in which case the IState is redundant. Their relation has been discussed before.

Especially extra method has been introduced due to the two interfaces not being related.

Two options were discussed:

```
public interface IIdentity : IIdentifiable, IDescribable {}
```

or alternatively

```
public interface IIdentifiable : IDescribable {
  string Id { get; }
}
```

Currently all classes implementing IIdentifiable also need to implement IDescribable. Hence there is no problem in either ways.

Vote resulted in the bottom case. 👍

IManagedState still returns the IIdentifiable, and will then include the description.

Does not have any impact on SDK.

Missing simulation time in events

We are trying to put more emphasis on different type of models, that need not be time progressing. Furthermore, from the ExchangeItemChangeEventArgs you can get to the exchange item and from there to the linkable component, which has the required information. 👎

Discussion turned into whether the ExchangeItemValueChanged event should be moved to the exchange item, such that you can register for a specific exchange item directly and do not need to register globally and filter out the exchange item that are not relevant. Would be preferable to have in case of distributed computation. But it is a bit more difficult to register in case you want to listen for all ExchangeItemValueChanged events. We move the event to the exchange item 👍 .

Add URI to IDescribable

The right way to add ontology support is most likely to extend the exchange item and the value definition. The URI there by itself does not solve the problem, it is too ambiguous. 👎

Remove GetHashCode() and Equals() overrides in Backbone package

👍

Similarly, the DescribeSameAs functions are a bit unclear: When does a quantity describe the same as another, which properties should match: description, unit (with description, caption, dimension and conversion factor) or dimension.

Suggestion to make utility functions in the SDK that can do the comparison on the required level and remove the DescribeSameAs methods.

Alternatively, the DescribeSameAs should be precisely documented.

⚠️

Simplify migration from 1.4 to 2.0

we would like to, but are lacking resources.

ExchangeItem filter interface

Can be done by adding special arguments. Otherwise it is an idea for an extension to the component.

Make IAdaptedOutput implement IInput

Will be reviewed ⚠️

Refine IManageState and IByteStateConverter interfaces

This is a general problem for remoting of .Net and java classes, and can/must be solved by the remoting component.

Use DateTime type in ITime interface instead of using double and TimeSpan for duration

Has been discussed many times. No new important arguments. All these methods can be implemented in the SDK (in .NET as extension methods, which will be able to provide the exact same interface as the DateTime).

Component tool interface

Suggested as a component

Generic versions of exchange items and valueset

Cons: It adds complexity to the standard (A standard developer may not have/do not want to bother about this level of skills), and it is specific to java and .net.

pros: It adds more type safety to the code, and also some performance.

⚠️

Add missing value definition to IUnit

Discussion where it should be: Quantity/ValueDefinition or Unit. IValueDefinition.MissingDataValue 👍 .

Add rasterdata support in IElementSet

Suggested as an extension to ElementSet.

Remove Values2D from IValueSet

It was introduced to make generic OutputAdapters be able to efficiently access the internal data. Only valid for time-space value sets, and should only be available on that specialization, but it will not be removed. 👎

Refine adapted output factories

The idea of several factories of a component is that each factory can describe it self, before it is asked to create its adapted outputs, so you may have two factories doing grid-to-line interpolation in two different ways, and you can choose which one to use. Also, it is easy for a component developer to add yet another general adapted output factory, if that is required. 👎

The issue with the awkward GetAdaptedOutputDescription() has been solved by letting the IIdentifiable extend IDescribable.

Update the IQuality interface

First change: Return type of IQuality.Categories differs in java and C#. Options:

```
IEnumerable<ICategory>
IList<ICategory>
ICategory[]
```

IList<ICategory> 👍 .

Second and third changes: Difficult to make as well a generic as a special quality implemention that does this correctly, taking into account that the category may come from another component elsewhere. Though it was found potentially usefull 👍 .

Loop approach

Duplicate - has to do with parallelism

Create more layering in the interfaces

The extension strategy supports this.

Rename IOutput and IInput into IOutputItem and IInputItem

Has been discussed before. There is was also discussed that without "item" it is very implicit. 👎

How to make the standard extendable

Meeting with EPA on how to make OpenMI and FRAMES work together, and also several reviewers, have revealed the necessity of making the standard extendable instead of "all inclusive". since the "all inclusive" is very difficult to predict.

The idea is to make very general versions of interfaces, and provide specializations of these. These specializations can be released at a later stage than the standard, and without affecting the standard.

See the section on "Possible candidates for extensions" and this additional discussion page about the Extendable OpenMI 2.0 version.

Strategy for releasing 2.0

Releasing standard and then extensions.

Possible candidates for extensions:

1. LinkableComponent: Parallel computing component
2. ExchangeItem: FRAMES dictinaries.
3. ValueDefinition: ontologies, FRAMES dictinaries.
4. ElementSet: Advanced spatial/GIS functionality.
5. ValueSet: FRAMES dictinaries.
6. TimeSet.

ad: ElementSet: Advanced spatial/GIS functionality:

Typical geometrical element types for models are:

- Point
- LineString - values defined on each LineString coordinate (node based values on a line)
- MultiLineString - values defined on each LineString (element based values on a line)
- 2D/3D mesh based - the IElementSet of today
    - Element values (finite volume type values)
- Node values (finite element type values)
- Grid based, element/node/face values
    - Structured grid, (x0, dx, xCount)^dimension + rotation from true north
    - CurveLinear grid (coordinates for all grid points)
- Polygon/MultiPolygon

Some of these have directly an OGC feature counterpart, while others (the grid) has not. Some of these are covered by the current element set, and some are not, or only in a "generic" geometrical way, i.e. without topology information.

So it would be nice to have a general element set and in time release extensions to the standard that includes new type of "often used"/requested element sets.

Remove loop approach.

Loop approach is not really tested and possibly only half included in the standard. It is better to completely remove it and then add it fully as an extension when it is mature.

Remove the ILinkableComponent.CascadingUpdateCallsDisabled property. 👍

Check in the status flag whether there are specific loop approach flags.

Releasing full standard

The full release, reflecting all suggested review requests, including fully functional and tested SDK and GUI.

Current problems:

No model, neither test models nor real world model has really been used and tested using the 2.0 standard. Hence, there are many unknowns and in order to get around all possible issues, quite a lot of work is required.

Limited resources within OATC for testing and development, in order to get as well the standard as the SDK "correct".

Limited resources for GUI development and support.

Updates to the standard

Documentation, IValueSet:

IExchangeItem.ElementSet == null : spatially independent
IExchangeItem.ElementSet.ElementCount == 0 : spatially dependent but no values available yet.

IExchangeItem.TimeSet == null : Time independent.
IExchangeItem.TimeSet.Times == null : No meaning, not allowed.
IExchangeItem.TimeSet.Times.Count == 0 : Time dependent, but no values available yet.

👍

IValueSet.ElementCount and IValueSet.TimesCount is removed - redundant 👍 .

Whats next

- Go through those issues marked with ⚠️ on a Skype meeting in the near future.
- Go through bugs and documentation issues listed in the review.
- Implement proposal for splitting up interfaces in general ones and specific ones. Jesper will look at in within the next month.
- Dictionary implementation updated according to meeting with FRAMES people. Stef will do that.
- Updates agreed on at this and last meeting should be implemented. Stef will take the lead, and can delegate.
- GUI: Someone needs to find resources for this!