OpenMI Association Technical Committee meeting no 26

See also: OATC Wiki Home

Date: November 9 - 12, 2009 Venue: Alterra, Gaia building, room C011; Wageningen, The Netherlands

Participants:

Rob Knapen, Alterra, Wageningen UR (Rob.Knapen@wur.nl) Standa Vanecek, DHI (s.vanecek@dhi.cz) Adrian Harper, Wallingford Software (adrian.harper@wallingfordsoftware.com) Stef Hummel, Deltares (stef.hummel@deltares.nl) Unknown User (don), Deltares (gennadii.donchyts@deltares.nl) ~jgr@dhigroup.com, DHI (jgr@dhigroup.com)

Apologies:

Unknown User (jnh@dhigroup.com), DHI (jnh@dhigroup.com) Peter Schade, Bundesanstalt fuer Wasserbau (peter.schade@baw.de) Daniele Andreis,Universita` di Trento,(daniele.andreis@gmail.com) Jan Gregersen, LicTek

Documents:

http://www.openmi.org/ http://sourceforge.net/projects/openmi wiki.openmi.org

Table of contents

- Participants:
- Documents:
- Table of contents
- 1. Status of the implementation
 - GUI:
 - Compliancy xsd:
 - Standard
 Ildentifiable
 - IElementSet.GetElementId / IElementSet.GetElementIndex
 - ILinkableComponent.Prepare()
 - UserCount
 - IArgument.IsReadOnly
 - IOutputItem.Consumers and IOutputItem.Decorators
 - Terminology
 - SDK
 - Java
- Standard
- Utilities/GUI
- Model migration
- Samples
 - Simple River :
 - GroundWater :
- 2. Documentation
 - Documentation of the standard.
 - What is OpenMI Tutorials
 - Scope document
 - Whats new in 2.0
 - ° GUI
 - ^o Projects source code structure
 - Reviewing environment for the Version 2.0
 - OpenMI 2.0 Standard specifications
 - How to
 - Overview/status of documents to be prepared
- 3 Source code structure
- 4 Support for Version 2.0 reviewing
- 5 Next meeting

1. Status of the implementation

GUI:

Under development and is usable for simple compositions, clearly work on this will continue until end of year

Compliancy xsd:

Sugestion = self contained task that review can be by email during december rather than at an OATC meeting

Standard

Ildentifiable

Accepting the need to keep IIdentifiable and IDescriable separate (as discussed at length in last meeting). I still feel uncomfortable with an interface with one method in. I find in the UI I am writing more code for which I feel no constructive benefit from except for making code more obscure to read. I would like to suggest that we replace the interface with a

Id { get; }

attribute on the interfaces that require it. I wont defend this if there is great objection to it, life's too short!

Has been discussed shortly and efficiently: We will leave it the way it is.

IElementSet.GetElementId / IElementSet.GetElementIndex

In these functions as string is used as Identifier. For consistency reasons, and to facilitate element access, we will introduce the Ildentifiable here.

ILinkableComponent.Prepare()

This method has been removed from the component. However, the majority feels that it should be in again, meaning that the component can rely on the fact that it will never get a GetValues() call or Values get-access before the Prepare() method has been called.

UserCount

A component should be able to keep track of how many users are interested in an exchange item's output. Has been discussed shortly and efficiently: We will leave it the way it is.

If a component performs an update, it may neglect all "non consumed" output items. An output item is consumed if its list of consumers contains at least one item, and/or if its list of decorators contains at least one item.

Related to this: do we need void Update(params IOutputItem[] requiredOutputItems), or is void Update() enough? The alternative is that the consuming component(s) register/deregister input items to/from the output items that they want to be updated. After some discussion we concluded that we need a relevant use case for this. Rob will create one in Java, Stef will port it to C#. If we decide to keep it in, we should explain in detail how a time progressing component behaves when the optional argument is given indeed.

My vote is to keep it. Typical use case is to ask component for an update only of some specific items, think about it as a query. We can then test components independently from the rest. E.g. query values only for selected areas event when component can provide data for many areas.

IArgument.IsReadOnly

Is it really really needed? (Stef thought so, but does not consider it essential anymore).

he property is needed indeed. Given the values of other arguments, an argument may become readonly because it simply may not be changed in that situation.

Now, the initialation of a component will go like this:

- The component instance is created.
- The component exposes its arguments (IlinkableComponent.Arguments).
- The value of an argument can be changed by the user.
- This value change of the argument may lead to different possible values and/or read only flags for other arguments, so the GUI will call <code>llinkabl</code> eComponent.Arguments again.
- When the user is happy, the Initialize() call is called, without arguments.

So the result of the discussion is that:

• IlinkableComponent.Initialize(IArgument[] arguments) will be changed to IlinkableComponent.Initialize()

Gena: ┢

IOutputItem.Consumers and IOutputItem.Decorators

There is currently asymmetry in these fields. On top of that, the outer world can add/remove consumers. After short discussion we decided to have:

```
public interface IOutputItem {
    IList<IInputItem> Consumers { get; }
    void AddConsumer(IInputItem consumer);
    void RemoveConsumer(IInputItem consumer);
    IList<IDerivedOutputItem> DerivedOutputItems { get; }
    void AddDerivedOutputItem(IDerivedOutputItem derivedOutputItem);
    void RemoveDerivedOutputItem(IDerivedOutputItem derivedOutputItem);
}
```

Both the Consumers and the DerivedOutputItems preferably should be implemented as readonly lists.

Gena: I find it redundant, if you can add and remove consumers, why not to keep it IList - it can add and remove and count things. My vote is for either:

public interface IOutputItem {
 IList<IInputItem> Consumers { get; }
 IList<IDerivedOutputItem> DerivedOutputItems { get; }
}

or:

```
public interface IOutputItem {
    IEnumerable<IInputItem> Consumers { get; }
    void AddConsumer(IInputItem consumer);
    void RemoveConsumer(IInputItem consumer);
    IEnumerable <IDerivedOutputItem> DerivedOutputItems { get; }
    void AddDerivedOutputItem(IDerivedOutputItem derivedOutputItem);
    void RemoveDerivedOutputItem(IDerivedOutputItem derivedOutputItem);
}
```

However in the later case performace may be an issue in remote scenarios - you'll have to iterate via the whole enumeration in order to get a single element or to compute Count().

Minor comment: IDerivedOutputItem sounds less explicit than IOutputItemDecorator. The later one decorates a single output item.

Terminology

- Name space: after some discussion we decided to change the name space to **OpenMI.Standard2**. The reason to do this, is to facilitate 1.4 model wrapping
- Are we sure that we want to use the name IOutputItemDecorator instead of the well known name IDataOperation?

Gena: no preferences, extept that from the name IDataOperation it is not clear that it is used for **Output** item. Should be at least IOutputDataOperation.

After some discussion a few suggestions are at the table in the table below.

base class	derived class	factoryname	remarks
lOutput	IDerivedOutput	IDerivedOutputFactory	methods in the factory need to be renamed as well
lOutput	IAdaptedOutput	IAdaptedOutputFactor y	Rename methods and properties, e.g. use "adaptee"

Gena: ... it does not sound more explicit than Decorator, the word **Derived** is more generic compare to **Decorated**. My suggestion is to keep it as it is unless there are good arguments and examples why it should be changed.

I'd keep the word **Item** in anyway, then it is clear what we work with one item here. Meaning of the word **Output** feels much wider compare to IOutputItem (output can be a set of all model output items).

• We should rename either IOutputItemDecorator.Update() or ILinkableComponent.Update() (or both). (Note: during the discussion on the latter, it appeared to be unclear if of IOutputItemDecorator.Update() should at its turn call its own decorators' U pdate functions. This is the case indeed. Jesper has stated this more explicitly in the documentation. After an extremely short discussion we choose Rob's suggestion: IOutputItemDecorator.Refresh()



SDK

The original packages in the 1.4 SDK are not that explitely identifiable in 2.0. Besides of that, more then one model wrapping utility has been developed (by Adrian and by Stef/Jesper), so we should determine what to put in which package. Other things that to take care of:

- split time buffering and time interpolation
- distinguish one or more decorator packages, implemenented as a third party factory

Java

Standard

During this week, the Java version of the standard has to be implemented and documented.

Rob: First iteration of the Java version has been completed and made available on SourceForge in the /trunk/src/java/OpenMI. Standard folder. There is a /bin folder with the compiled jar library of the Standard interfaces, a ./javadoc folder with the source code documentation in HTML format, and the ./src folder with the Java source code. In this folder you also should find the changes. txt file that briefly lists the things I have modified compared to the C# version of the Standard that I based the Java implementation on. I used the C# source code from SourceForge, versions of November 9. Modifications mostly reflect latest things discussed in this TC meeting, programming language differences, and rewritten source code comments.

Now that this work been completed we need to agree on a way to keep both (C# and Java version of the Standard) updated in sync.

Utilities/GUI

We will currently put no effort in addinonal java developments, like the backbone, the GUI etc. Most probaby, there will be no time left in the OpenMI-Life project (ending end of Januari) to put any real effor to java. However, next to having the standard, we will take to:

- have examples of IQuality/ICategory available in the C# version
- put the already available java utilities, like the Buffer and the ElementMapper, on sourceforge and point out that they are there.

Rob: Alterra is now considering to host Java Open/II SDK and Editor development on its own server with svn, trac (ticket system) and possible Hudson (continuous integration and maybe automatic QA). Looks like next year at least there will be resources available to work on a 2.0 SDK and maybe Editor. We can open the server for public contributions / collaboration on the development work for other interested parties. Will proceed along these lines unless OATC votes against it (the official Java Standard interfaces will stay on SourceForge and under OATC control).

Model migration

Samples

Simple River :

Have reviewed and modified src as Stef requested

GroundWater :

NOTE JESPER, have also started modifying the simple GW model for use in examples/tutorials etc., currently WIP will take you through that at next weeks meeting. You might notice that I have moved these examples into new folder structure to facilitate sharing of common code. This common code might find its way into the SDK in the future, or remain seperate as a seperate example. For us to decide lator.

2. Documentation

Main task for the meeting

Create maximum documents to be send to Stephen for review. Assign rest of the documents to the individual persons

Documentation of the standard.

What it the status of Stephen's review on the C#-oode for the standard? Standa will send Stephen an e-mail.

What is OpenMI - Tutorials

1 minute tutorial - users // suggestion Download Bin + install Run GUI Load composition Run Composition See Messages - progress Look to the results

Status: To be realized as part of the on line help. Will be done by Adrian.

10 minute tutorial - users // suggestion Download source code??? Download documentation (standard) ?? Download / compile + install Open GUI Create composition - Run composition See Messages - progress Look to the results Add decorator - run Open Reference manual - see basic schema Clases Status diagram Composition XLS OnenMI model compliensy Load complex composition Review of the Components / decorators, links ... properties Open source code of the hard coded running of the composition

Status: To be realized as part of the on line help. Will be done by Adrian.

Tutorial for developers Gena will sugest

Scope document

taken from the 1.4 and expanded

Whats new in 2.0

Final review of existin document need to be done

GUI

Help document, with 2 turials Started, using SimpleRiver and SimpleGW models, first draft on schedule for Nov end

Projects source code structure

This document will be also used as the text for How To

Reviewing environment for the Version 2.0

Description of the process and supporting tools /Wiki, voiting system /reviwers registration....

OpenMI 2.0 Standard specifications

Standard Reference manual (class library) need to be checked if all comments are in the source code -> generate UML - Steve will send the corrections to the Stef

SDK Developer guide/documentation

SDK Reference manual (class library) comments need to be placed in source code -> generate UML

How to

Various HowTo's will be provided (see the How To). The table below shows the status of and the persons working on these HowTo's. Possible statuses:

- todo
- to adjust (according to new Standerd/SDK)
- first version
- basic skeleton exists (fill in)
- reviewed
- sent (to Stephen Morris)
- ready

More How To will be taken moved there from the guidance - only references will stay there

subject	status	page- author	page- reviewer
How to generate a XML file describing compliancy with OpenMI 2.0	first version	Peter	Stef
How to migrate from version 1.4 IEngine	first version	Jesper	Adrian
How to work with OMI files	first version	Peter	Adrian
How to link OpenMI components via hardcoded interface calls	first version (see note 1)	all	Standa
How to turn an Ascii file reader into a Linkable Component	first version	Jesper	Stef
How to connect to GIS	first version	Standa	Stef
How to turn a database into a LinkableComponent	first version	Jesper	Stef
How to link models with different grids (spatial mapping)	to adjust	Stef	Jesper
How to download the most recent source code	reviewed	Peter	Standa
How to get started with OpenMI and Java	first version	Peter	Rob
How to migrate existing Fortran based models codes	todo (see note 2)	Stef	Adrian
How to generate a LinkableComponent with a Fortran engine on Linux	todo (see note 3)	Peter	Gena
How to port the OpenMI from Windows to Linux	todo	Peter	Gena

Note 1: The unit test compositions serve as a base for this. The various unit test programmers will take care that the HowTo page refers to the right source conde.

Note 2: This is a big task, lot of changes need to be done

Note 3: The OpenMI 2.0 compliant component has not been not been generated on Linux yet. The generation should follow the same rules as the generation of a component being OpenMI 1.4 compliant

Overview/status of documents to be prepared

Standa will look at all 1.4 documents, and will check what can be reused directly or with minor changes for the new 2.0 documents.

The following persons will work on the following documents:

subject	status	author	reviewer	remarks
Whats new in 2.0	first version	Standa	Stef	
Scope document	todo	Jesper	??	
OpenMI in a nutshell (covers both .Net and Java)	todo	Adrian	Rob	
OpenMI 2.0 Standard specifications	todo	Stef	??	
1 minute tutorial	todo	Adrian	??	
10 minute tutorial	todo	Adrian	??	
GUI user manual	todo	Adrian	??	
SDK guide/documentation editing	todo	Standa	??	
. sdk Backbone:	todo	Stef	??	
. sdk Buffer:	todo	Stef	??	
. sdk Spatial:	todo	Stef	??	
. sdk Modelwrapper(s):	todo	Jesper	??	

. sdk 1.4 Engine wrapper:	todo	Jesper	??	
Projects source code structure	todo	Gena	Jesper	
Reviewing environment for the Version 2.0	todo	Gena	??	

3 Source code structure

Whole structure of the source code, examples, unitest... were reorganised (not used files deleted...) to provide clear based for the Version 2.0. Gena will and more details there + prepare the document describing new projects structure

4 Support for Version 2.0 reviewing

Supporting structure for the Reviewing needs to be prepared

- Wiki
- Downloads
- Reviewer registration
- Voting system ???

Gena will prepare sugestion for the structureAll came with the ideasWhen will be agreed - Gena will implement it and write short description

5 Next meeting

Next meeting will be handeled as the extension of the January OpenLife final meeting 13-14.1 2010. Place will be specified later.