OpenMI Association Technical Committee meeting no 24

See also: OATC Wiki Home

Date: June 8 - 11, 2009
Venue:Deltares , Rotterdamseweg 185, Delft, 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)
Unknown User (jnh@dhigroup.com), DHI (jnh@dhigroup.com)
Peter Schade, Bundesanstalt fuer Wasserbau (peter.schade@baw.de)
Rob Brinkman, Deltares (rob.brinkman@deltares.nl)
Bert Jagers, Deltares (bert.jagers@deltares.nl)

Apologies:
Jesper Grooss, DHI (jgr@dhigroup.com)
Jan Gregersen, LicTek, (Gregersen@LicTek.dk)
Unknown User (moovida), Andrea Antonello, Universita` di Trento, (andrea.antonello@gmail.com)

Documents:

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

Table of contents

agenda

1. Minutes from previous OATC meeting

1.1 Status of the Linux test

Deltares did not find time yet. This afternoon a test will be done on one of the Deltares 64 bit systems (**action**: Bert).
Later this week, preferably Wednesday or Thursday, (**action**) ~~Rob K. will test it at Alterra, on a 32 bit machine.~~ **done June 10**: Tested on my work laptop, which has a Intel Core 2 Duo (T7600) CPU (Multicore 64 bit), but currently with a 32 bit Ubuntu version installed (I should upgrade this one day). Mono is included in the Ubuntu 9.04 version I used, but I did need to install some other Mono related packages (gac, base System libraries, etc.). With that done the Configuration Editor started, loaded the sample composition provided by Peter and ran through it without problems. I also quickly tested with Mono on Mac OS X, which in principle works but has some GUI Windows Forms related problems causing exceptions and not fully being able to use the Configuration Editor.

1.2 Release 1.4.1

No action taken yet, so still pending. However, in 1.4.0 there appear not to be any really blocking problems.

1.3 www.OpenMI.org and wiki.OpenMI.org

The Dissimination Committee has concluded that the current www.OpenMI.org is is quite OK, so this site will remain the main OpenMI entry.
However, we will still move the wiki.OpenMI.org content from the public.wldelft.nl site to the new confluence server that has been prepared to serve the wiki.OpenMI.org space.
Action: Gena / Stef / Deltares automation department. Will be done by Friday 19 th.

2. Status of the implementation .NET

2.1 Changes to the standard

All proposed changes resulting from the previous meeting have been realized.

2.2 Backbone

Gena updated the backbone according to the version of the Standard after meeting 23.
Small modifications by Stef and Adrian afterwards.
Note: the documentation is not up to date at all.
Actually: the documentation of the Standard is the most important one. We could already start to work on this documentation for the more static items, like the ValueDefinition, ElementSet, etc.

2.3 Buffer

The buffer has been "ported" to the latest 2.0.
However, there is an issue to adress: the extrapolation, interpolation, etc. ⚠ Will be addressed next meeting.

2.4 Time interpolation Decorator

Stef experimented a bit. Two major issues to have a look at came up:

- when to extrapolate/interpolate
- the component needs to know which decorator is a TimeDecorator.
  See discussion in chapter 4 on decorators.

2.5 GUI

The Gui is quite on its way. To be done:

- details on working with decorators
- actually run the application
  Details on working with decorators have been presented by Adrian. 👍 We all agreed to the approach of introducing a button that shows a window containing the available decorator and/or factories. The selected one will be added to the component, and thus to the list of output items. By repeated doing this step, the end user can build a chain of decorators.

Action Adrian will also take care that the documentation on the GUI is up to date.

2.6 SimpleRiver Model - running in GUI

While developing the GUI, the Simple River Model has been updated also (there might be small issues when really running it).
Note: the Update cascade mechanism is not implemented yet. **Action**: Stef will do that.

2.7 OpenMIException : Exception - suggestions

Introducing the OpenMI exception is being rediscussed.
The opinion is that it might be done, but that it does not have a very high priority. During the present meeting we will focus on other, more urgent, issues.
The OpenMI exception then will be considered at the september meeting ⚠.

2.7 OpenMI 2.0 wrapper around OpenMI 1.4 Components

Jesper worked on it quite a bit, together with Stef when needed. Quite some progression was made.
The content of the task shifted from wrapping a 1.4 LinkableComponent to wrapping the 1.4 Sdk.Wrapper.IEngine.
So people who have implemented their wrapper based on the IEngine will be able to migrate their model to 2.0 quite easily.
Action DHI will take care that there will also be a document describing how to migrate.

3 Status of the Java implementation

We once again discussed to which extend we should support Java for version 2.0.

Part	To be done or not?	How to do it?
Standard	surely t.b.d.	convert C# to Java, including comments, semi-automatically
Backbone/Buffer	most probably	take java-1.4 as base
Spatial	most probably	take java-1.4 as base (there have been only slight modifications to the IElementSet
Wrapper	probably not	
Configuration Editor	probably not	

4 Pending from previous meetings / Other items to be discussed

1. Question: what does isAvailable() mean in case of multiple consumers. Do all consumers need to have the same time(s)?
Discussion/Conclusion: 👍 We decided to extend the IsAvailable() method with an IExchangeItem argument, so that the caller can specify which availability he wants to check:
bool IsAvailable(IExchangeItem querySpecification).
The ValueDefinition/ElementSet/TimeSet definition of the querySpecification specifies exactly what is needed.
Generally speaking, it is one of the consumers that is passed as the argument:

```
IInputItem consumerThatIsAsking;
consumerThatIsAsking.TimeSet.Times[0] = requiredTime;
while (!providingOutputItem.IsAvailable(consumerThatIsAsking)) {
    providingOutputItem.Component.Update();
}
values = providingOutputItem.Values;
```

Note (Gena): We may want te consider looking again at the IsAvailable method, and separate it into 2 different indicators, e.g.

- IsValid on the exchange item
- canProvide or something like that on the decorator.

Note (Stef) On the other hand, this would be confusing for the consuming input item (is it accessing an output item or a decorator?).

For now, we will various implementation examples using IsAvailable. We will then check if IsAvailable suffices.

Related issue: The **while(!Available)**{Update} code above is actually representing the original 1.4 GetValues() call. So it might be wise, to put emphasis again on the still working 'pull approach', to introduce a getValues() call, e.g.:

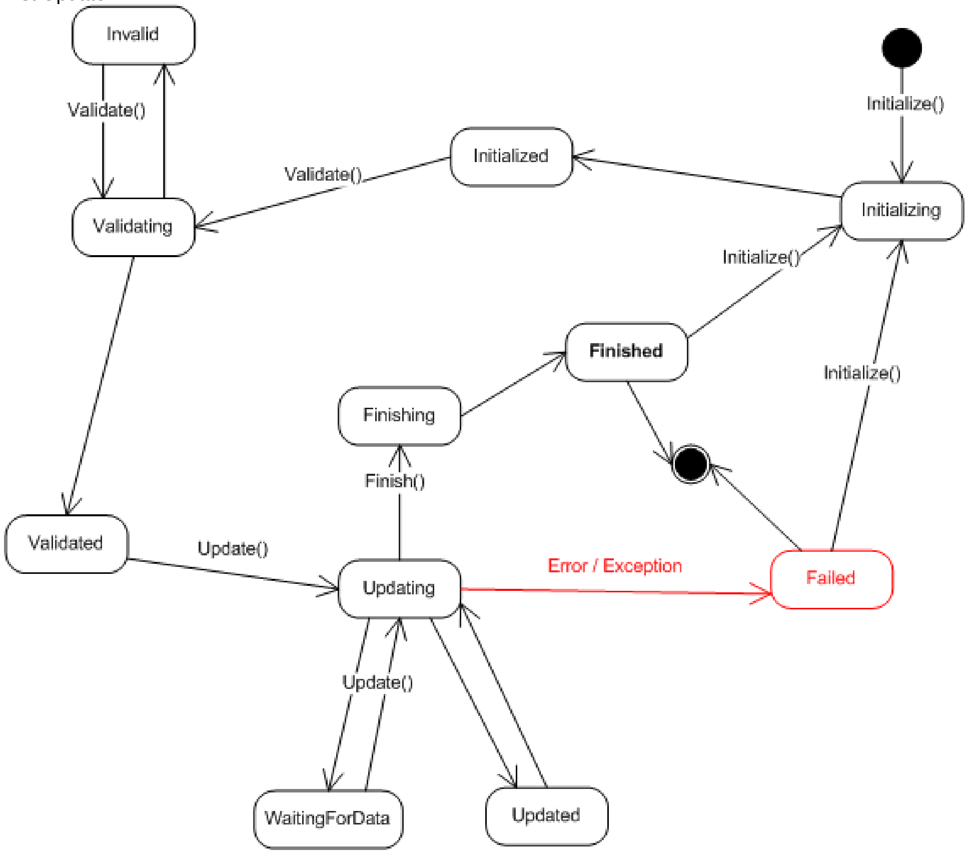
```
values = providingOutputItem.GetValues(consumerThatIsAsking);
```

This is considered to be a useful extension. It is a convenient method, and it explitly shows the pull approach, so we will add it 👍 .
It might even be considered to move the method to the LinkableComponent, see also Gena's suggestion on loop vs. pull later on this page.

1. Remark: When experimenting with the IsAvailable(), Rob B. and Stef introduced the IExchangeDefinition (see below). Rob K. actually introduced the same interface in the java version. Indeed it is quite natural entity, the exchange definition. **Question**: do we want to introduce it ❓ .

```
public interface IExchangeDefinition
    {
        IValueDefinition ValueDefinition { get; }
        ITimeSet TimeSet { get; }
        IElementSet ElementSet { get; }
    }
```

1. Are we happy with IOutputItem.IsAvailable() and with ILinkableComponent.Validate(), or do we need additional methods on LinkableComponent and/or ExchangeItem to check validity.
   Conclusion after a first discussion: we are happy. The only small change that has been proposed and has been agreed on is 👍:
   The return value will be an array of strings, to let the component compose a multi line message.
   Some second thoughts came up when Gena emphasized that by this mechanism the outer world can not determine which input/output exchange items are in erroneous state, and that a component is not able provide messages during initialization.
   Conclusion after a second discussion 👍: The Validate will indeed remain as decided. The StatusChangedEventArgument will be extend with a array of message strings, thus offering each component to provide also messages on other status changes.
   During the discussion the issue came up on when to call Validate. 👍 We agree that it only can be (repeatedly) called between Initialize and the first Update.



1. Should we introduce a boolean argument that indicates whether the component should run in a pull mode (like in OpenMI 1.4) or in loop mode?
2. Discuss component operation mode here **comment from Peter Gijsbers need to be discussed**
   Discussion/Conclusion: 👍 We decided to let the LinkableComponent have a propery: **CascadingUpdateCallsDisabled**.
   The default is **false**, indicating that the component is running in Pull Driven mode (there will be a cascade of update() calls. This Pull Driven mode has to be supported by every component.
   If set to **true**, the component is expected to run Update()-step by Update-Step(), controlled by some outer world (which may be another component). If the component does not support this Update() by Update() way of running, it will throw an Exception when **CascadingUpdateCalls Disabled= true** is called.
   In the GUI, the user hat to tell which component is at the end of the chain. This *controlling component* will be triggered first.

1. The Timezone issue has been re-adressed.
   Discussion/Conclusion led to the following decisions 👍:
   a. Daylight Saving Time jumps in time series are not allowed
   b. The TimeSet will contain a property that expresses its offset from UTC, expressed in hours (**OffsetFromUtcInHours**). The Gui will check these offsets, and if they are different, it will ask the user to put a decorator in between. The offset has to be applied to both the timeSet's Times and its TimeHorizon.
      Note: in future and/or more platform specific versions of the standard we may reconsider introducing the timezone info again.

1. Decorator issues:

        - Do we need to recognize the type of decorator (time, space, SI-conversion)?
        - Do we want to support 3th-party decorators indeed?
          Discussion/Conclusion: 👍 We will add an Update() method to the OutputDecorator. The linkable component calls this Update() on all its decorators at the end of its own Update(). This actually is completely the same as what happens in the current 1.4 Wrapper: after IEngine.PerformTimeStep the wrapper updates the buffers.

- Decorator issues, GUI:
    - How to support the end user, when "chaining" decorators? 👍 Taken care of, see section 2.5
    - To identify/describe the various decorator factories, the IOutputDecoratorFactory will be IIdentifiable/IDescribable

- Decorator issues, other:
    - Is is useful to let the component have a property OutputDecorators, containing the list of already created decorators? 👎 For now, we will stick to the "keep it lean and mean" principle. As soon as it is really clear that it is needed ~~(**action** Gena)~~ it will be added.
      Gena: **example added here**: OutputItemDecorators property in ILinkableComponent
      ⚠️ All Oatc members should have a look at it, and give their opinion as soon as possible.

- IArgument needs a mechanism for identifying the type of argument e.g. File, Path, int double etc Very usful for providing customised GUI functionality
  Various solutions were proposed (see below). Disadvantage of having an enumeration in the Standard is that it can not be extended easily. An object type definition would be more specific.
  Rob K. concludes that Alterra's approach works quite well for what we want. The approach is based on recognized strings in the key of the Argument.
  Action: ~~Rob K. will provide an example.~~
  Done June 10: Please see the IArgumentV2.java class file that I have attached to these meeting notes. This interface documents all the extensions we made to standard OpenMI 1.2 IArgument interface to make it more usable for our (GUI related) purposes (some time ago). The comments in the source file should provide all the explanation needed 🙂
  We have studied the example and various propositions, and choose 👍 for to the following solution:

```
public interface IArgument : IIdentifiable, IDescribable
{
    Type ValueType { get; }
    bool IsRequired { get; }
    bool IsReadOnly { get; }
    object Value { get; set; }
    object DefaultValue { get; }
    IList<object> GetPossibleValues();
    string Characteristics { get; }
}
```

The **Characteristics** attribute can be used to describe requirements on or additional qualification on the argument, e.g.: "NewFile", "ExistingFile". Although understanding the characteristics is GUI-dependent, we expect that a list of generally known qualifiers will gradually arise.

The IsReadOnly attribute defines whether the Values property may be edited from outside, as in OpenMI 1.4.
Some Oatc members think it should be removed (assuming that every argument is editable), others think it is needed.
Stef has given two examples of 1.4. omi files where to his opinion it is needed indeed.

Adrian's proposal:

```
namespace OpenMI.Standard
{
    public enum EArgType
    {
        String = 0, // default
        Bool,
        Int,
        Double,
        Path, // presummed to exist and accessable
        FilePath, // presummed to exist and accessable
        PathNew,
        FilePathNew,
        XML, // Rob: can be used together with a schema for complex argument data
        MIME, // Rob: might be useful to pass in images, or text documents, etc.
    }

    /// <summary>
    /// The IArgument interface defines a key - value pair. If the property ReadOnly is
    /// false the value is editable otherwise it is read-only.
    /// </summary>
    public interface IArgument : IDescribable
    {
        /// <summary>
        /// Type that string value represents and can be converted to
        /// </summary>
        EArgType ValueType { get; }

        /// <summary>
        /// The key (string) in key-value pair.
        /// </summary>
        string Key {get;}

        /// <summary>
        /// <para>The value (double) in key-value pair.</para>
        ///
        /// <para>If the ReadOnly property is true and the property is attempted to be changed
        /// from outside an exception must be thrown.</para>
        /// </summary>
        string Value { get; set; }

        /// <summary>
        /// Defines whether the Values property may be edited from outside.
        /// </summary>
        bool ReadOnly {get;}
    }
}
```

Gena's proposal:

Current version:

```
    public interface IArgument : IDescribable
    {
        string Key { get; }
        string Value { get; set; }
        bool ReadOnly { get; }
    }
```

Proposed change to cover types:

```
public interface IDescribable
{
    string Caption { get; set; }
    string Description { get; set; }
}

public interface IValueDefinition : IDescribable
{
    Type ValueType { get; }
    bool DescribesSameAs(IValueDefinition otherValueDefinition);
}

public interface IArgument : IValueDefinition
{
     object Value { get; set; }
     bool ReadOnly { get; }
}
```

1. Suggestion by Gena on pull / loop approach
   Discussion/Conclusion: 👍 A good idea to introduce the GetValues, to recognize the pull approach. However, instead of on the LinkableComponent, for now it has been put on the output item (see above). If it turns out to be more logical to be on the LinkableComponent, we may move it there. To be considered at next meeting ⚠️.

```
// =================================== pull approach

var triggerExchangeItem = component.OutputItems[0];
triggerComponent.GetValues(triggerExchangeItem);


// =================================== loop approach

while(!allComponentsAreFinished)
{
    foreach(var component in components)
    {
        if(component.Status != LinkableComponentStatus.Finished)
        {
            component.Update();
        }
    }
}

public interface ILinkableComponent
{
    LinkableComponentStatus Status { get; }
    void Initialize();
    void Validate();
    void Update();
    void Finish();
    IList<IInputItem> InputItems { get; }
    IList<IOutputItem> OutputItems { get; }
    bool CascadeUpdateCallsDisabled { get; set; }

    // Suggestion: ask for specific values: e.g. filtered data
    IList GetValues(IInputExchangeItem query);

}
```

1. Gena's suggestion for 2D return type (see below)
   Discussion/Conclusion: 👎 Not to be included in the standard 2.0. Could be used in the backbone, and probably in later versions of the standard.

```
public interface IInputItem : IExchangeItem
{
    IMatrix Values { get; set; }
}

public interface IOutputItem : IExchangeItem
{
    IMatrix Values { get; }
}

public interface IMatrix : IList
{
    int Rows { get; }

    int Columns { get; }

    object GetValue(int row, int column);

    void SetValue(object value, int row, int column);

    object[,] GetValues(int startRow, int endRow, int startColumn, int endColumn);

    void SetValues(object[,] values, int startRow, int endRow, int startColumn, int endColumn);
}
```

5. Release 2.0 plan

5.1. Documentation / basic / changes

We will identify the parts of the Standard that are quite stable by now. These Items are:

- IValueDefinition and all related interfaces/enumerations
- IElementSet and related enumerations
- ITimeSet / ITime

This means that effectively the only interfaces not to be documented directly are:

- ILinkableComponent
- IExchangeItem and its descendents.

Once the documentation of the more stable items is complete, we will check what the documentation will look like in the Object Browser, Enterprise Architect, and in Doxygen.

Steps to take before we can invite external reviewers to have a look at things:

- **Action**: Johan will take the lead in gathering reviewers (all of us have to think of names).
- **Action**: Rob B. will write a 2 page document on what are the main differences between 1.4 and 2.0.
- **Action**: All: cleanup all code in 2.0 from code which is not relevant anymore (left from 1.4). This is including unit and integration tests.
- **Action**: All of us will update the documentation of the interfaces, properties and methods whereever and whenever possible
- **Action**: On july 10th, Standa will tag a version, generate EA-documentation, and inform the reviewers.

5.2. Alpha release

The result of this meeting is a candidate for the alpha release.
In the weeks to come, every Oatc member can plead for minor adjustments. If the Oatc agrees to the adjustment, it will be incorporated in the final alpha release. The last day for these changes will be July 2 nd, because:
👍 The alpha release will be produced on Friday July 3 th, so it can be presented on the July 4 th.

On the page Version 2.0 Alpha We will keep track of:

- the opinions and comments on the standard
- the results of various tests
- etc.

5.3. External review and further migration of models

The external reviewers can start to do their work from Monday July 6 th on.
They will be asked to submit their remarks before September 1 st.

5.4. Migration of commercial models and documentation

Has not been discussed.

6. OATC Procedures

7. Miscellaneous issues

During the various discussion some things came up as may useful for, or even strongly recommened to be in version 3.0.
We will keep track of a wish list for version 3.0.

7.1 SourceForge clean up

Recently, I checked out the entire OpenMI source forge repository. This took forever, the size of the whole repository is 4.2GB. From this, 3.8GB is located in directories called "Wrappers", which is mainly code and data from HEC, RAS, CUAHI and MODFLOW. Long time ago we made a folder on the OpenMI source forge called MyOpenSource. The idea behind this directory was to have a place where people could put some code, that did not fit into the SDK, but still could be useful for OpenMI developers. Placing an entire model, data and binary files there is in my opinion too much. It is annoying to have so long download times when checking out and it may also add to some confusion, that the OpenMI repository mainly consists of various wrappers. My suggestion is to ask those responsible for the code in the wrappers directories to make their own repositories on source forge, where after we can remove all these files from the OpenMI repository. In order to make sure that people know about these important wrapper developments we could make a page on the wiki with a short description of the various projects and link to the location of the code.
Kind regards
Jan Gregersen

Discussion/Conclusion: Gena will have a session with the Hec-Ras and Modflow developers at the beginning of july. Most propably their developments will be moved out of the current repository.
Action Gena will take care that, once this is done, the OpenMI sourgeforge project will contain a reference to the Hec-Ras and Modflow developments.
Comment by Jan: Thanks 🙂

7.2. September OATC meeting in Hamburg

- Do the OATC members agree with a public talk?
- Aim: Spreading the OpenMI in Germany
- Target group: Developers
- Tuesday afternoon, 9th of September

Discussion/Conclusion: We all agree that indeed it will be useful to have a public session on on Tuesday afternoon on the OpenMI 2.0 developments. The main part of the public will be java oriented, and we may not have java up and running by then, but the .Net version most probably will suffice to introduce the new concepts to the developers.
Before mid of july the OATC will confirm that indeed the public session will take place.

8. Tasks and unresolved issues

All tasks are handled by sourceForge. GOTO: OpenMI Tasks on source forge

They have not been reviewed at this meeting.

9. Any other business

No other business.