3 Wrapping

3. Wrapping

The OpenMI standard was designed to allow easy migration of existing model engines. The standard is implemented in C# running under the .NET framework. Almost all existing model engines are implemented in other programming languages, such as Fortran, Pascal, C and C++. In order to bridge the gap between the different technologies and to minimize the amount of changes needed to be made to the engine core a wrapping pattern will be the most attractive choice in most cases.

This chapter describes the process of wrapping and the generic wrapper that is provided by the OpenMI Software Development Kit (SDK).

3.1. A general wrapping pattern

Wrapping basically means that you create a C# class that implements the ILinkableComponent interface. This wrapper will communicate internally with your engine core. The wrapper will appear to the users as a 'black box', which means that all
communication will take place through the ILinkebleComponent interface (Figure 5).

Fig. 5 OpenMI wrapping pattern

One further advantage of using the wrapping pattern is that you can keep the OpenMI specific implementations separated from your engine core. Typically, the engines will also be used as standalone applications where OpenMI is not used and it is naturally an advantage to be able to use the same engine in different contexts. This means that even in situations where new engines are built the wrapping pattern may still be the best choice.

3.2 The LinkableEngine

Model engines that are doing timestep-based computations have many things in common. It is therefore possible to develop a generic wrapper that can be used for these engines. This wrapper is called LinkableEngine and is located in the Oatc.OpenMI.Sdk.Wrapper package. Basically, the LinkableEngine provides a default implementation of the OpenMI.Standard.ILinkableComponent interface. Naturally, the LinkableEngine cannot know the specific behavior of your model engine; this information is obtained though the IEngine interface.
The recommended design pattern for model engine migration when using the LinkableEngine is shown in Figure 6. The design includes the following classes:

- The MyEngineDLL is the compiled core engine code (compiled dll from e.g. Fortran).
- The MyEngineDLLAccess class is responsible for translating the Win32Api from MyEngineDLL to .NET (C#).
- Calling conventions and exception handling are different for .NET and Fortran. The MyEngineDotNetAccess class ensures that these operations follow the .NET conventions.
- The MyEngineWrapper class implements the IEngine interface, which means that it can be accessed by the LinkableEngine class.
- The MyLinkableEngine class is responsible for the creation of the MyEngineWrapper class and for assigning a reference to this class to a protected field variable in the LinkableEngine class, thus enabling this class to access the MyEngineWrapper class. The MyLinkableEngine class implement the OpenMI.Standard.IlinkableComponent interface and therefor is the LinkableComponent (the OpenMI compliant component).

More details of these classes are provided in the following sections.
The OpenMI standard puts a lot of responsibilities on the LinkableComponents. The main idea is that when the GetValues method is invoked the providing component must be able to deliver the requested values so that these apply to the requested time and the requested location. To be able to do this the LinkableComponent may have to interpolate, extrapolate or aggregate both in time and space. These and other things are handled by the LinkableEngine.

The LinkableEngine class includes the following features:

- Buffering: When a model is running as an OpenMI component it may be queried for values that correspond to a time that is before the current time of the model. Most models will only keep values for the current timestep and the previous timestep in memory. It is therefore necessary to store data associated with the OpenMI links in a buffer. The LinkableEngine handles the buffering for you.
- Temporal interpolation and extrapolation: Most models are only capable of delivering results at times that correspond to their internal timesteps. The LinkableEngine class handles all the temporal operations that are required for LinkableComponents.
- Spatial operations: The LinkableEngine provides a range of spatial data operations.
- Link book-keeping: The LinkableEngine handles book-keeping for links added to your component.
- Event handling: The LinkableEngine sends events that enable an event-listener to monitor the progress of the linked system when running.

More details about how the LinkableEngine works is given in OATC.OpenMI.SDK technical documentation.