OpenMI: A standard interface for linking environmental models

Costas Gavardinas¹, Fotis Fotopoulos¹, Peter Gijsbers² and Roger Moore³

Abstract

Environmental processes and interactions have always been a modelling challenge not only because of their complexity but also because they usually demand an integrated analysis that can be achieved only by interlinking domain-specific and case-tailored models. However, the scientific community has reached a general consensus that no single model or modelling system can represent adequately the full range of these processes. Moreover, up to now the flexibility of such linkages has been minimal and entirely dependent on modelling software developers. The HarmonIT project, a European Union co-funded research project bringing together developers from all over Europe, has tried to solve this problem by designing and implementing a standard interface for linking mathematical models: the Open Modelling Interface & Environment (OpenMI).

1. Introduction

The integrated management of environmental processes requires an analysis supported by integrated modelling systems. Such systems are essentially a collection of interlinked models, each representing an aspect of the system, that exchange data according to the physical interactions modelled. Up to now these linked systems were either sequential, exchanging data by file sharing, or based on models developed by the same vendor, implementing a custom and proprietary information exchange protocol and having the links hard-coded into the models. In the first case, the representation of physical processes was severely limited or even flawed, whereas in the second case the modeller was bound to a single software developer and the data exchange was obscure. The OpenMI Standard aims at solving this inherent coupling limitation by specifying an architecture that enables models, both legacy and new ones, to exchange data on a time basis.

In order to fully understand the role of OpenMI in model linking, let us first look into model applications. This term is used to describe the entire software system running on a computing infrastructure and can be roughly split into the following components: the user interface, the input files, the model engine and the output files (Figure 1). In order to establish inter-model communication at run-time, OpenMI effectively needs direct access to the model computational core. In most models the engine can be instantiated separately from the user interface and is described by a well-defined interface; it is thus an engine component. By implementing the OpenMI standard interface, the engine component becomes OpenMI-compliant and is henceforth called a Linkable Component.

¹ National Technical University of Athens, School of Civil Engineering, Department of Water Resources, Hydraulic and Maritime Engineering, Laboratory of Hydrology and Water Resources Management, Iroon Polytechniou 5, 157 80 Athens, Greece, Tel. +30 210 772 2878, Fax. +30 210 772 2879, E-mail: {cgavar,fistopol}@chi.civil.ntua.gr
² WL|Delft Hydraulics, P.O. Box 177, 2600 MH Delft, The Netherlands, E-mail: peter.gijsbers@wldelft.nl
³ Centre for Ecology & Hydrology, Wallingford, Oxfordshire, OX10 8BB, United Kingdom, E-mail: rvm@ceh.ac.uk

14.08.2006, GavardinasC.doc
2. The OpenMI Architecture

The OpenMI basis is a context-based request and reply mechanism, also classified as a pull-based pipe and filter architecture. The communicating components interact in couples (source and target) and OpenMI both specifies the interfaces enabling the exchange and prescribes the format of exchanged data.

The communication between Linkable Component instances is single-threaded: each component must reply to a request before handling another request. Components are chained together by their data requirement relation, which they enforce autonomously without any external orchestration. The ultimate component of the chain is the one which sets the computations and inter-component data exchange in motion through its initial request and is the final receptor of data: the answer to the initial request.

In order for the linked models to understand the exchanged data, it must be expressed in a commonly understood format. The OpenMI defines the base data (numerical values) and its semantics in terms of quantity (what), element set (where), time (when) and data operations (how). Meta-data are used to specify the data that can or will actually be exchanged over the link between two Linkable Components. The data transfer is realized by calling a single method of the source component: GetValues(). The source component will then try to satisfy the request by calculating the data, retrieving it from a buffer (if it has been already calculated), performing data operations on it (unit change) or “guessing” it (interpolation, extrapolation, etc.) or even by issuing further GetValues() requests to components it is linked to.

The basic coupling functionality described above is achieved by the implementation of the Linkable Component interface. This can be extended with state management (required if the model execution needs to be paused and resumed) and discrete time information. Moreover, in order to make the actual linked model simulation easier, an XML Schema has been defined for OMI files, which describe the model engine class name, the assembly it is contained in, initialization arguments and its physical location in a networked environment.

Apart from the basic data exchange mechanism, a lightweight event framework is also available to allow for messages to be communicated between components. This is an asynchronous interaction which can be used for call stack tracing, progress monitoring, fault handling, data visualization and other cases where components need to be notified to request data by issuing a GetValues() call. Exceptions are not redefined and the default implementations of the development environment should be used to signal an irrecoverable error state that halts the entire simulation.

Thanks to the inherent simplicity of the OpenMI mechanism and its operation at the actual data handling and computation level (the model engine), its applicability is not restricted to water-related models (where its development was initiated) but extends to all mathematical models involving data manipulation and output. Furthermore, non-computing entities easily fit into this data exchange paradigm: databases, data visualization components, real-time monitors, GIS interfaces and any other numerical data producers or consumers can be linked into a chain of OpenMI-compliant components.

![Model Application](image1.png)

**Figure 1**

Model application structure

![The OpenMI package hierarchy](image2.png)

**Figure 2**

The OpenMI package hierarchy
3. The OpenMI Environment

The OpenMI Standard is the specification of interfaces that an entity must implement in order to become a Linkable Component. In order to assist the software coding process, HarmonIT has provided a collection of concrete implementation classes that make up the OpenMI Environment and can be directly referenced in the model code, but their use is not mandatory. The `org.OpenMI.Backbone` classes are the default implementation of the `org.OpenMI.Standard`. The `org.OpenMI.Utilites` namespace contains a generic model `Wrapper` that can directly communicate via an internal interface to the existing engine core and thus reduce substantially the model migration time. It also provides a `Buffer` that holds calculated values and has been extended with data operations (unit conversion, temporal aggregation). The `Spatial` package enables the representation of spatial information (primitive data types) and geometrical transformations on it. The `Configuration` package enables the reuse of linked component chains (particularly useful in scenario management) by providing composition, save, load and deployment functionalities. Finally, the `AdvancedControl` package is used in cases external component coordination is required, such as directed iterations, calibration and optimization.

The OpenMI Environment contains two further namespaces: the `org.OpenMI.Tools` provides high-level utilities for the developer (a data monitor for real-time visualization and an event logger) and the `org.OpenMI.DevelopmentSupport` includes useful data handling functionality such as XML reading and writing and calendar conversion.

4. Making a model OpenMI-compliant

Although OpenMI drastically changes the way environmental models co-operate, making a model OpenMI-compliant is usually a straightforward procedure that aims at enabling direct access to its computational engine. Firstly, the model must expose information about the modelled variables it can input and output as well as their potential locations. Furthermore, the model initialization phase should be separated from the actual computation and boundary conditions should be collected during the computation.

In order to avoid deadlocks, the engine must be always able to return a value for a requested variable at the specified time and location. This can be achieved by direct computation, data operation and manipulation or by delegating the request to a further linked component able to provide this value. If the returned value set contains missing values they should be flagged and in case of inability to compute an entire result set, an exception should be thrown.

5. Running a coupled-model simulation

A simulation consisting of OpenMI Linkable Components can be conceptually divided into six phases. Firstly, the individual components are initialized, execution parameters are passed and the models are possibly populated with data, ending in a state where each component can expose its exchange items (variables and data). During the next phase of identification and configuration the links between components are added and the scenario is validated. After the phase of preparation components have performed any required setup actions and are ready to begin calculations. The bulk of the simulation takes place during computation/execution: the trigger component kicks off the request-reply mechanism and data is transferred between components. After all computations have been completed, components enter the finalization phase and clean up the memory and files that were used. Finally, the linkable components are disposed and all remaining objects are de-allocated from memory. The deployment phases, the methods called and the properties specified in each phase are shown in Figure 3.
6. A real life application of the OpenMI

The case study presented here is a typical problem involving two networks functioning simultaneously, a pressure water distribution network and a sewer network. Up to now, it has been common practice to design the water network and then manually transfer a percentage of the demand at each node (most regulations suggest 80 to 90%) to the sewer network pipes. However, there are many limitations to this procedure: firstly, the base demand in a water network is time-variant. This implies that when a percentage of a node demand is transferred to the sewer network, one has to choose a single value from a wide range of inflows. Secondly, if one changes one or more properties of the water network, a new manual solution of the sewer network needs to be calculated; this can be a lengthy procedure prone to errors. Finally, studying the real-time interaction of the two networks or replacing a model with another equivalent, cannot be achieved without significant effort.

The water and sewer networks of the city of Amfissa in Greece were used to test the coupling between the water, storm and sewer network models. The city of Amfissa features all three networks, gradually constructed over the last two decades to serve an ever-expanding population which is currently around 10,400 residents. The engineering calculations for the infrastructure networks date back to the late 1970s. However, nowadays the city deals with severe sewer and water network issues mainly due to construction problems and land and population density changes that were the main cause for different water consumptions than the originally expected.

The models used were the TLWaterNET, a 1-D pressure water distribution network, the TLSewerNET, a 1-D sewer network and the TLStormNET. Coupling the three models together significantly accelerated the simulation of the infrastructure networks. The forcing model is the water network and the rainfall. Approximately 80% of the water consumed is considered to end in the sewers while 20% ends in the storm network together with the rainfall. Using OpenMI ID-based links, it was no longer necessary to specify the inflow at the sewer or storm network’s inlets, since the node demand from the water network was automatically passed to the linked sewer and storm network nodes. Assuming that the links have been correctly defined, which is a trivial procedure once the networks are over-imposed, one of the most time-consuming and error prone activities during data input has been avoided. Once the three networks were coupled, it was fairly simple to account for the changes in population density and land changes by altering the water consumptions (water demands) at specific nodes in the water networks. Through the OpenMI links these changes were propagated to the other networks at all time steps.

The coupling of the models for Amfissa revealed several discrepancies in the original hydraulic calculations using disjoint models, in respect to water consumptions in the water network and the design of sewer
flows in the sewer and storm networks. Table 1 summarizes the results of the models when applied separately and within the OpenMI framework: using the OpenMI it was concluded that several parts of the storm and sewer networks could not accommodate the required quantity of water or sewage because the diameter of the piping was too small. These discrepancies may have been caused due to the different design assumptions or varying regulations but were untraceable before the application of OpenMI for coupling the models involved.

<table>
<thead>
<tr>
<th></th>
<th>Length (km)</th>
<th>Standalone Solution</th>
<th>Using OpenMI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Water Network</td>
<td>40.6</td>
<td>No problems</td>
<td>No problems</td>
</tr>
<tr>
<td>Sewer Network</td>
<td>35.3</td>
<td>No problems</td>
<td>Inadequate diameter for ~9.5 km</td>
</tr>
<tr>
<td>Storm Network</td>
<td>17.2</td>
<td>No problems</td>
<td>Inadequate diameter for ~2.7 km</td>
</tr>
</tbody>
</table>

Table 1 - Results of model application before and after OpenMI linkage

7. Final remarks and conclusion

The OpenMI Standard is an open specification: the full interface description, the source code of the Standard and the supporting Environment (in C# and Java) and extensive documentation are freely available through the largest repository of Open Source software, SourceForge, under the Lesser General Public License (LGPL). Making an existing or new model OpenMI-compliant does not require any licensing or copyright limitations. This is why several commercial software developers and academics from around the world have already adopted the OpenMI into their practices and are taking advantage of its benefits while providing invaluable feedback for future improvements. Moreover, the HarmonIT partners have committed themselves to supporting, maintaining and improving OpenMI well into the future.

The success of the Open Modelling Interface and Environment will unveil new perspectives for advanced environmental research as it will relieve the environmental modelling and management community of the physical linking problem and enable it to focus on yet unexplored aspects of integrated systems. Existing knowledge and experience spanning multiple scientific domains will be consolidated into mature decision support systems, leading to sensible resource management and planning. The adoption of OpenMI as a universal model linking standard promises to be the key to an exciting and innovative modelling reality.

Bibliography

Buschmann et al., Pattern-Oriented Software Architecture, a System of Patterns, John Wiley & Sons, 1996