OpenMI: A Standard Interface for Linking Environmental Models

Costas Gavardinas¹, Fotis Fotopoulos¹, Peter Gijsbers² and Roger Moore³

Abstract

Modelling environmental process interactions has always been a challenge. Most attempts have resulted in domain-specific, case-tailored models representing the interaction of a small group of processes. From this work, it is now clear that there is little chance of building a single model or modelling system that will adequately represent the full range of processes. A number of teams have developed frameworks, which have provided a more generic solution to the linking problem but still tend to be domain specific and lack the flexibility that is ideally required. HarmonIT, a European Union co-funded research project bringing together developers from all over Europe, has now developed a solution to the problem by designing and implementing a standard interface for linking mathematical models; it and the supporting software are called the Open Modelling Interface & Environment (the OpenMI). The paper describes the interface and explains the principles upon which it works. It also gives an overview of the environment that has been built to facilitate the linking and running of linked models.

1. Introduction

The management of interacting environmental processes can be greatly aided by the use of integrated modelling systems for predicting the outcomes of alternative strategies. Such systems are essentially a collection of linked models, each representing an aspect of the system. Up to now these linked systems either run the models sequentially, exchanging data by file sharing, or run them in parallel. In the latter case, the set of linkable models are usually restricted to those of the developer, who will have implemented a custom, often proprietary, information exchange protocol with the links hard-coded into the system. Both approaches represented a considerable step forward at the time of their introduction. However, the increasing pressure on resources is creating a demand for improved and more reliable modelling that will give better predictions of process interactions. Serial linkage may often be adequate to represent the interaction of uni-directional processes, such water flowing down a river to the sea. Its major weakness is that it cannot represent the feedback loops that occur in most environmental processes. Running models in parallel does allow the representation of feedback, however, the disadvantage of the early approaches is that they lack flexibility and often bind users to the software of a single supplier. The HarmonIT project has developed a new approach, the OpenMI, which addresses the current limitations and advances the linking process by specifying a standard interface that enables both legacy and new models to exchange data on a time step by time step basis. The paper will briefly describe the OpenMI’s terminology and its architecture, explain how models can be migrated, linked and run and outline a real life application.

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,ffotop}@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
2. OpenMI Terminology

A number of terms are used when describing the OpenMI standard. As shown in Fig. 1, the term *model application* encompasses all parts of the modelling system software that is installed on a computer. Typically, such systems consist of a *user interface* and an *engine*. Usually, the engine is a generic representation of a process and this is where the calculations for simulating or modelling that process take place. The user supplies information through the user interface and this is converted into the input data for the engine. The data describe a specific scenario in which the process is to be simulated: for example the Rhine during a time of extreme rainfall.

When an *engine* has read its input it becomes a *model*. For example, an engine may represent the generic process of water flowing in an open channel. When it has read in the data describing the channel network of the Rhine, along with any boundary conditions and rainfall data, it becomes a model of the Rhine in the scenario to be simulated. If the code for an engine can be instantiated separately and has a well-defined *interface* through which it can accept and provide data, then it is an *engine component*. The key to enabling models to exchange data lies in standardizing the design of the engine interface. When an engine component implements the OpenMI standard interface, it becomes a *linkable component*.

3. The OpenMI Architecture

The OpenMI is based on a *context-based request and reply* mechanism, also referred to as a *pull-based pipe and filter* architecture. The communicating components interact in couples (source and target), the OpenMI specifying the interfaces that enable the exchange and prescribing 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 first component of the chain is the one which sets the computations and inter-component data exchange in motion through its initial request. It is also the final receptor, ultimately receiving the answer to the initial request.

In order for the linked models to understand the exchanged data, they must be expressed in a commonly understood format. The OpenMI describes the data to be exchanged in terms of quantities (what), element sets (where), times (when) and data operations (how). Meta-data are used to specify the data that *can* or *will actually* be exchanged over each 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 or, if they have been already calculated, by retrieving them from a buffer, by performing data operations on them (temporal and spatial aggregations and disaggregations and unit conversion) or by estimating them by interpolation or extrapolation, or even by issuing further *GetValues()* requests to components to which it is linked.

The basic coupling functionality described above is achieved by the implementation of the *Linkable Component* interface. This can be extended with state management, which is required if model execution needs to be paused and resumed. In order to make the actual linked model simulation easier the OpenMI defines the OMI files: they are XML 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.
Due to the simplicity of the OpenMI mechanism and its operation at the data handling and computation levels (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.

4. The OpenMI Environment

The OpenMI Standard is the specification of interfaces that an entity must implement in order to become a Linkable Component and hence OpenMI compliant. In order to assist the software coding process, HarmonIT has provided a collection of implementation classes that make up the OpenMI Environment and can be directly referenced in model code; however, their use is not mandatory. The org.OpenMI.Backbone classes are the default implementation of the org.OpenMI.Standard. The org.OpenMI.Utilities namespace contains a generic model Wrapper that can directly communicate via an internal interface with an 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. 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, which are particularly useful in scenario management, by providing composition, save, load and deployment functionalities. Finally, the AdvancedControl package is used in cases where 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.

5. Making a model OpenMI-compliant

Making a model OpenMI-compliant is usually a straightforward procedure, whose aim is to enable direct access to the model’s computational engine. To achieve this, the model must expose information about the modelled quantities that it can accept and provide together with the locations at which they can be calculated. The model initialization phase must also be separated from the code simulating the process for a time step. Boundary conditions should be collected during the computation phase.
In order to avoid deadlocks, the engine must always return a value for a requested variable at the specified time and location. This can be achieved by direct computation, extrapolation or interpolation or by delegating the request to a further linked component able to provide the requested 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.

6. Running a coupled-model simulation

A simulation consisting of OpenMI Linkable Components can be conceptually divided into six phases. First, 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 (quantities). 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. A trigger component starts the simulation 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.

7. A real life application of the OpenMI

The case study presented here is a typical problem requiring linked models and involves two networks functioning simultaneously, a 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 and serving an ever-expanding population, which is currently around 10,400 residents. The engineering calculations for the infrastructure networks date back to the late 1970s. Today the city has many sewer and water network issues mainly arising from construction problems and population density changes. The latter have lead to higher demands for water than were originally anticipated.

The models used were the TLWaterNET, a 1-D pressure water distribution network, the TLSewerNET, a 1-D sewer network and the TLStormNET. The forcing model is the water network and the rainfall. Approximately 80% of the water consumed is considered to end up in the sewers while 20% ends up 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 networks’ 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 super-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 significantly accelerated the simulation of the infrastructure networks. It also 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.

<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>

Tab. 1: - Results of model application before and after OpenMI linkage

8. 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 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. The HarmonIT partners are preparing to commit 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 as 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