18th World IMACS / MODSIM Congress, Cairns, Australia 13-17 July 2009
http://mssanz.org.au/modsim09

OpenMI – The universal glue for integrated modelling?

Knapen, M.J.R. [1], **P. Verweij** [1], **J.E. Wien** [1], **and S. Hummel** [2]

[1] *Alterra, Wageningen University and Research Centre, Wageningen, The Netherlands*
[2] *Deltares, Delft, The Netherlands*
Email: rob.knapen@wur.nl

Abstract: Integrated modelling is one of the key aspects of Integrated Policy Assessment Tools, which are used to find out whether, why and how policies are developed and what the possible options for these policies are. They do this by determining the economic, social and environmental impacts for each option. Where modelling used to be a domain-centric undertaking, to support integrated assessments the modellers now need to describe and work with models that span various domains (mostly environment, economy and social sciences). Conceptually this is challenging enough, and it would be helpful when the technical burdens of models integration could be simplified by a standard framework.

Many such frameworks already exist, and new ones seem to be invented per project. For several large Integrated Assessment projects, funded by the EU, we choose to build upon the Open Modeling Interface and Environment (OpenMI) standard, which has been in development since a few years. OpenMI is a data and model integration framework, designed to take independent data and computing systems and provide a standard means of describing how time series are communicated between the systems.

Because the original development target of the OpenMI was the hydrological domain, some work has been needed to increase the applicability for other environmental domains in order to use it as glue for integrated modeling. These modifications have been contributed to the OpenMI Association and the Technical Committee (OATC), following the open source principles, and helped steer the development of the next major version (2.0) of the OpenMI.

In this paper we discuss many of the changes that will be included in OpenMI 2 and will show how, with the addition of richer data types, the exchange of values as objects, the possibility to use both a push (set) and pull (get) approach for the data exchange between components, clearer state management and better OGC compatibility, the OpenMI will become a more universal standard for model linking.

It is also concluded that with all these changes the biggest risk to be avoided is the OpenMI interfaces becoming too flexible, introducing a lack of clarity that could result in incompatible components and Software Development Kit (SDK) implementations. The OpenMI Technical Committee should take care to prevent this from happening.

Keywords: *Open Modelling Interface (OpenMI), Integrated Assessments (IA), Integrated Modelling*

1. INTRODUCTION

1.1. Integrated assessment and modelling

Societies see the emergence of new governance concepts, based on the assumption that processes of planning and decision taking are no longer hierarchical but the product of complex interactions between governmental and non-governmental organizations, and the general public (according to the model of "co-production of knowledge" Callon, 1999). All involved are seeking to influence the collectively binding decisions that have consequences for their interests.

To account for this changing governance and the increased number of stakeholders involved, decisions need to be assessed in an integrated context. One of the possible approaches to support this process is by combining models and modelling tools into linked model chains and use it to simulate future scenarios. This approach is also known as Integrated Assessment and Modelling (IAM; Parker et al., 2002) and is one of the key aspects of Integrated Policy Assessment Tools, which are used to find out whether, why and how policies are developed and what the possible options for these policies are. They do this by determining the economic, social and environmental impacts for each option.

Where modelling used to be a domain-centric undertaking, to support IAM the modellers now need to describe and work with models that span various domains (mostly environment, economy and social sciences). Conceptually this is challenging enough, and it would be helpful when the technical burdens of models integration could be simplified by a standard framework. Many of such model integration frameworks already exist (e.g. TIME, OMS, MODCOM, KEPLER). This paper focuses on only one of them, the Open Modelling Interface (OpenMI; Moore and Tindall, 2005) which has been proposed as a European standard to facilitate the exchange of data at run time between models.

1.2. What is the Open Modelling Interface (OpenMI)

The OpenMI (www.openmi.org) is a data and model integration framework, designed to take independent data and computing systems and provide a standard means of describing how time series are communicated between the systems. It has been developed from the need to answer integrated hydrological catchments management questions within the EU 5th framework program project HarmonIT. The main objective of the HarmonIT project was to provide a widely accepted unified method to link models, both legacy code and new ones (Gijsbers *et al*, 2002). The OpenMI (version 1.*) provides a standardized interface to define, describe and transfer data between software components that run sequentially, based on a *pipes and filters architecture* (Gregersen *et al*, 2007). The data definition concerns what the data is about (*quantity*) and where (*element set*) and when (*time*) it applies. Each component (*LinkableComponent*) has a meta data description of its exchangeable data in terms of a *quantity* and an *element set.* Each unique exchangeable *quantity* is registered and published in a so-called *ExchangeItem*. Connections between *ExchangeItems* of *LinkableComponents* are defined by a *Link* and exist as a separate entity.

1.3. OpenMI Governance

The OpenMI is constantly under development under supervision of the OpenMI Association, which has as main objectives to promote the development, use, management and maintenance. It is an entirely open international group of organizations and people, with a small core team that supports, responds to and is guided by a growing active worldwide user community. It is a not for profit organization and therefore depends on the willingness of its members. Organizationally, the OpenMI Association consists of a General Assembly, an Executive Committee (OAEC), a Technical Committee (OATC), and a Dissemination Committee (OADC).

1.4. OpenMI compared to other frameworks: OMS, MODCOM, TIME, KEPLER

Many integrated modelling frameworks already exist, and new ones seem to be invented per project. A few well known solutions are:

- OMS – The Object Modelling System: A pure Java, object-oriented modeling system framework that enables interactive model construction and application based on components. It is a collaborative project active among the U.S. Department of Agriculture and partner agencies and organizations involved with agro-environmental modeling. (Carlson *et al.*, AgSAP 2009).
- MODCOM: A framework that facilitates the assembly of simulation models from previously and independently developed and tested component models. A small, but dedicated group of developers

build the MODCOM software and it is distributed under the terms of the GNU General Public License, available at http://www.modcom.wur.nl. (Hillyer *et al.*, 2003).

- TIME – The Invisible Modelling Environment: a software development framework for creating, testing and delivering environmental simulation models. TIME includes support for the representation, management and visualization of a variety of data types, as well as support for testing, integrating and calibrating simulation models. (http://www.toolkit.net.au/Tools/TIME). TIME is intended for use by modelers who program in Visual Basic, C# or Fortran 95. Other languages, such as C++ and Java are also usable with TIME, though their use is not explicitly supported. While all modelling frameworks simplify the task of creating models, by providing reusable components for data handling, visualization and model execution, TIME further simplifies the task by providing a high level, meta data driven environment for automating common tasks, such as creating user interfaces for models, or optimizing model parameters. This reduces the learning curve for new developers while the use of commercial programming languages gives advanced users unbridled flexibility.
- KEPLER: A scientific work flow application designed to help scientists, analysts, and computer programmers create, execute, and share models and analyses across a broad range of scientific and engineering disciplines. Kepler can operate on data stored in a variety of formats, locally and over the internet, and is an effective environment for integrating disparate software components, such as merging "R" scripts with compiled "C" code, or facilitating remote, distributed execution of models. Using Kepler's graphical user interface, users simply select and then connect pertinent analytical components and data sources to create a "scientific work flow"—an executable representation of the steps required to generate results. The Kepler software helps users share and reuse data, work flows, and components developed by the scientific community to address common needs. The Kepler software is developed and maintained by the cross-project Kepler collaboration, which is led by a team consisting of several of the key institutions that originated the project: UC Davis, UC Santa Barbara, and UC San Diego. Primary responsibility for achieving the goals of the Kepler Project reside with the Leadership Team, which works to assure the long-term technical and financial viability of Kepler by making strategic decisions on behalf of the Kepler user community, as well as providing an official and durable point-of-contact to articulate and represent the interests of the Kepler Project and the Kepler software application. http://kepler-project.org/.

Compared to the frameworks listed above the OpenMI is the youngest and thus a bit less evolved. On the other hand it has the unique feature that it in principle only sets a standard based on interfaces, currently defined for both the .NET and the Java languages. A reference implementation is provided as Software Development Kit (SDK), but using it is not mandatory. Fulfilling the defined interfaces in the proper way, which is rather easy to do since they are minimal and partly only descriptive, is enough to ensure OpenMI compliance. Since it is less bound to a specific environment it is a good candidate for cross-framework linking and supporting multi-framework models (Argent and Rizolli, 2004). Other key points are the existence of the OpenMI Association and the open source character of the OpenMI development, and the use of OpenMI in several large European Commission (Sixth Framework Programme) projects, see http://ec.europa.eu/research/fp6/index_en.cfm.

2. USING THE CURRENT OPENMI (1.4)

2.1. Case studies (“other” domains)

For several large Integrated Assessment projects funded by the EU Sixth Framework Programme, the OpenMI was selected as basis to build upon instead of starting yet another modelling framework. Since its release in early 2006 the OpenMI is well applied in the hydrological domain, and the following IA projects tried to adopt (and adept) it for the broader environmental domain:

- SEAMLESS – assess agricultural and agro-environmental policies.
- SENSOR – assess sustainability impacts of land use related policies.
- EFORWOOD – assess sustainability impacts of European forest wood chains, which are influenced by policy, market drivers and technological innovations.

All these projects end in 2009.

2.2. Proposals for OpenMI 2.* changes

Since the original development target of the OpenMI was the hydrological domain and linking mostly time-stepping, quantitative model engines, some work was needed to increase its applicability for the other environmental areas addressed by the projects listed in the previous section. These modifications were contributed to the OpenMI Association and the Technical Committee, following the open source principles, for consideration for the next major OpenMI release. Next to that the years of practical use in the hydrological domain also resulted in feedback on the original (version 1.*) standard and requests for changes. A few of the larger issues:

- Support for qualitative and other (e.g. structured) types of data, besides quantitative data.
- Improved support for other components than those based on time-stepping model engines, e.g. constant value provider, 0D component, GIS-based models, analytic functions, databases, optimizer and scenario managers, etc.
- Allow one input to get data from multiple outputs and to define how to process it (e.g. sum).
- Store spatial and other descriptive information with exchanged data.
- Improve the simplicity of use, e.g. by supporting some native language elements like collections.

3. THE NEXT VERSION OF OPENMI (2.0)

Partly funded by the LIFE-Programme of the European Commission (http://ec.europa.eu/environment/life/) in the OpenMI-Life project (http://www.openmi-life.org/) and with contributions of the European projects mentioned before, the OpenMI Technical Committee took up the list of change requests and currently is defining and building the next version of the OpenMI (2.0). The long term aim is that the OpenMI should become the European and global standard for model linking in the environmental domain.

3.1. Key changes

In the new version, as in version 1.*, the exchange items play a significant role. An *ILinkableComponent* has a list of input and output items. The input items describe what can be accepted as input, and the output items describe what can be provided as output by the component. For version 2 the exchange items will now also contain the actual values, previously you could only ask the component to provide values for a quantity given specific spatial and time period references. In the new version values are represented as a list, that can contain *any object* as opposed to version 1.4 where the values always were either an array of doubles, or an array of X, Y, Z vectors. Data can now also be qualitative or quantitative (supported in the standard, see Figure 1), or any custom implementation (complex types).

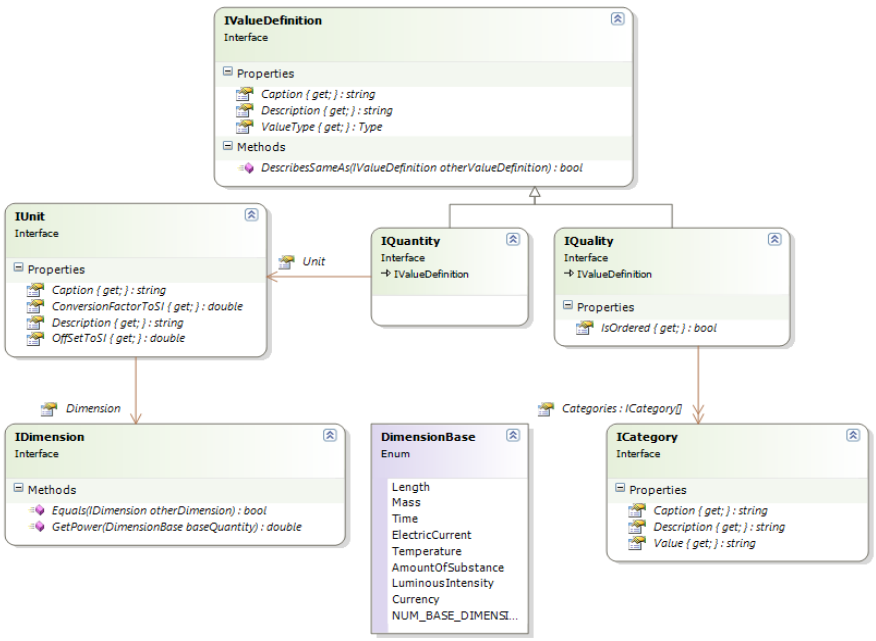

Figure 1: Extended value definition interfaces of OpenMI 2

For input items the values list is of type set (write), whereas the values list in output items is of type get (read). Allowing to *set values* is also a major change as compared to version 1.* and is introduced mainly to make it easier to work with optimizer and scenario manager type of components. It also, together with some changes to the linkable component interface, makes it possible to move beyond the strictly pull-based calculation approach of OpenMI 1.*. In version 2 each component can specify which operation modes it supports (e.g. pull and/or push). When the component is initialised an operation mode is selected, which then is expected to be used for calculation and handling requests for values.

Inputs and outputs contain the actual values – no spatial, temporal or unit conversions. Getting values from the input or output will not trigger any calculations. When a component needs to get values not already available in the output (there is an *isAvailable* property that can be checked), the component can invoke the update method in the *ILinkableComponent* interface.

The component is then to investigate what data is requested (i.e. the update call is for certain inputs connected to outputs of the component), and perform calculations or other actions needed to present the data in its outputs.

One possible way to use this is as follows (for a pull-based system):

1. Component1 <>--[OutputItem1]→[InputItem1]--<>Component2
2. Component2 updates and sets times, elements, quantity/quality which it needs in the InputItem1.
3. Component1 updates and sets new computed values into its OutputItem1.
4. OutputItem1 will check its consumers (*IInputItems*) and if they are compatible will set values into them.
5. If *inputItems* are not compatible there should be a decorator inserted in between (or more than one), or an error should be generated.

This, however, is at the moment of writing still under discussion within the OpenMI Technical Committee. What is clear is that the separate *ILink* interface from the version 1.* will be removed and replaced by direct references between inputs and outputs (i.e. an input knows its provider, and an output knows its consumers).

In version 1.4, to retrieve values from a model component, its *getValues* method must be called and the model should deliver these values represented at time, locations and the unit defined by the requesting component. When needed one or more *DataOperations* could be applied to convert the data. In version 2.* this will be achieved by requesting a decorated output item from the model component by invoking its *getDecorator* method. A decorator is an output item adapted specifically to the input item that is passed as argument to the *getDecorator* method. Decorated output items have an *argument* property that makes it possible to define further details about how the data is to be provided – e.g. by defining the interpolation method to be used. Figure 2 shows an example of the use of decorators to convert between spatial representation of the data, from what is delivered as standard output of the component to what is requested by another component. The lines existing of edges and nodes (circles) in the diagram illustrate the spatial representation and its transformation by the decorators.

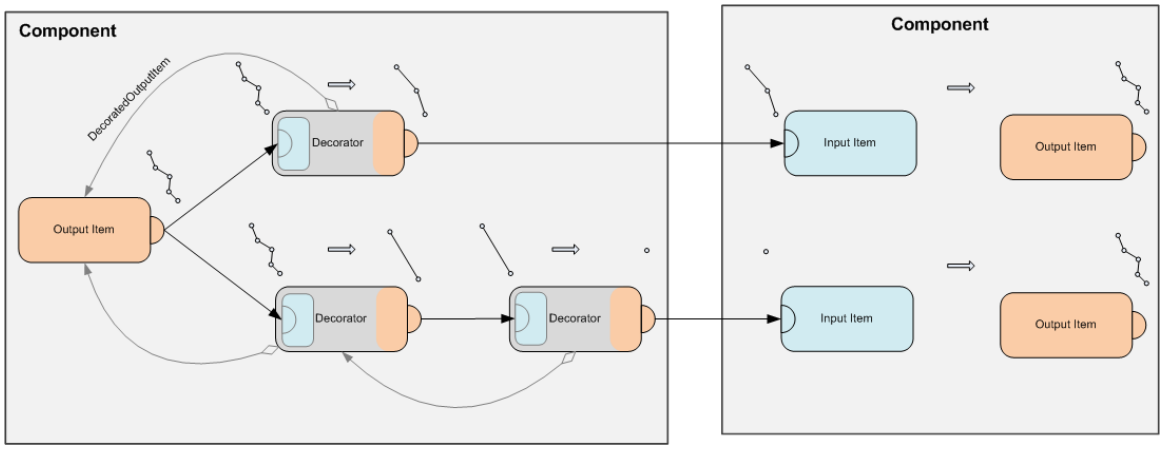

Figure 2: Use of output item decorators in OpenMI 2

And finally there are some smaller changes being worked on:

- Increased use of implementation language specific elements, e.g. the native event mechanism of .NET and the Observer/Observable pattern of Java, native collection classes and native date/time classes. In version 1.*, the OpenMI interfaces were kept as language independent as possible, resulting in the introduction of a custom event system and methods like **getSize()* and **getItem(index)*. Although more generic, this approach required more boilerplate code to convert from the OpenMI standard to the language specific counterparts.
- An improved definition of the *IState* and *IManageState* interfaces of OpenMI, which a component can implement to support saving and restoring of internal state.
- OpenMI *ElementSet* and spatial reference definitions used brought more in-line with the Open Geospatial Consortium (OGC, http://www.opengeospatial.org/) OpenGIS Simple Features Interface Standard specification and well-known text (WKT) standards. WKT is a text mark-up language for representing vector geometry objects on a map, spatial reference systems of spatial objects and transformations between spatial reference systems.

3.2. Development time line

At the moment of writing this paper (March 2009) version 2.0 of the OpenMI is still under development. The initial new version of the interfaces is frozen (called the "Trento" version), but some issues remain to be solved or clarified in more detail. The target for the end of March is to do so by implementing a draft SDK and use it to test the main concepts. Language specific details from the Java platform are also included. In the next steps up to the release by end December 2009, further testing, refining and finalizing is planned, currently according to the following time line:

- End January 2009: "Trento" version of 2.0 interfaces (standard).
- End March 2009: .NET SDK and testing of main concepts, Java interfaces.
- End April 2009: Configuration editor and further refinements.
- End October 2009: External review and further migration of models.
- End December 2009: Adjustments according to reviews, submit standard release to OAEC.

3.3. Does it get more universal?

With al these (planned) changes, does the OpenMI get more universal? We think so. With the addition of richer data types, exchange of objects, the possibility to use both a push (set) and pull (get) approach for the data exchange between components, clearer state management and better OGC compatibility, integrating different other types of components (than just time-stepping model engines), application of the OpenMI becomes possible with less overhead required. For the projects referenced in this paper, much custom coding could have been avoided.

Increased use of native programming language elements like the event mechanism and collections frameworks will make the OpenMI more convenient to work with for developers. The biggest risk to be avoided is that the OpenMI interfaces get to flexible, introducing a lack of clarity that could result in incompatible components and SDK implementations. The OATC should prevent this.

It is also clear that the work of the OpenMI Association and having it available as open source is really needed to get the required input and community support to set it as a standard for model linking in the environmental domain.

4. FUTURE ENHANCEMENTS

With the release of the OpenMI version 2.0 the development of the OpenMI will not stop. There are many more interesting areas to research and potentially include in the OpenMI. A few current ideas:

- Increased use of semantic information to describe components and exchange items. By using ontology the OpenMI would better fit into the semantic web world and, for example, reasoning engines could be used to facilitate model integration. Some steps towards this have been taken in the SEAMLESS project (Knapen *et al*, 2009).

- Apply the OpenMI for modeling framework interoperability. By using only the interfaces defined in the OpenMI standard and integrating them into existing model integration frameworks, it can potentially be used for framework interoperability. Definitely it would be easier to have such a standard in-between than attempting to make N:M connections between the frameworks.
- Combining the previous two points together with merging the web standards for Service Oriented Architecture (SOA), like UDDI, WSDL and SOAP, in general could make using models within an enterprise or across organizations easier. Users could be assisted (semi-automatically) in finding and selecting models and creating mash-ups of them. The version 2.0 of the OpenMI already is less stateless that the old version (although more attention to this is still needed) which is needed to support Enterprise Model Mash-ups, but it will also require a lot of awareness about this new purpose and use of models by the model creators.
- On the SDK side of the OpenMI, working with it could be made less invasive, for example following approaches from other frameworks like Spring (http://www.springsource.org) and Hibernate (http://www.hibernate.org), e.g. by using plain classes and annotations or XML configuration files to use them with the OpenMI.

5. DISCUSSION AND CONCLUSIONS

This paper provided some details of the development of the OpenMI, from its current version 1.4 to the new major 2.0 release that will be finalized by the end of December 2009. It was concluded that this version took well into account all the comments and feedback from a number of large European projects that applied the OpenMI into the environmental domain, mainly in the field of Decision Support Systems for Integrated Assessments. In general, the feeling is that the new version of the OpenMI has lots of potential to take integrated modeling from the hydrological domain into the environmental domain and perhaps beyond that. Some nice potential future work for the OATC has been listed as well.

ACKNOWLEDGEMENTS

The Sixth Framework Programme for Research and Technological Development (FP6) is a collection of the actions at EU level to fund and promote research. With a budget of 17.5 billion euros for the years 2002 - 2006 it represents about 4 to 5 percent of the overall expenditure on RTD in EU Member States. The main objective of FP6 is to contribute to the creation of the European Research Area (ERA) by improving integration and co-ordination of research in Europe.

The OpenMI Association, and the members of its Technical Committee in particular provided much of the information about the upcoming 2.0 release of the OpenMI standard.

REFERENCES

Argent R.M. and Rizolli, A.E. (2004), Development of multi-framework model components. *International Environmental Modelling and Software Society (IEMSs)*, 2004.

Callon, M. (1999), The Role of Lay People in the Production and Dissemination of Scientific Knowledge. *Science, Technology, and Society*, 4(1), 81-94.

Carlson, J., David, O., Ascough, J., Geter, F., Ahuja, L. (2009), The role of the Object Modelling System (OMS) for integrated assessments of conservation on agricultural land in the United States. *Proceedings AgSAP Conference 2009, Egmond aan Zee, The Netherlands*, 324-325.

Gijsbers, P.J.A., R.V. Moore, C.I. Tindall (2002), Towards OMI, an Open Modeling Interface and Environment to harmonise European developments in water related simulation software, *Hydro-informatics*, 2002.

Gregersen, J.P., Gijsbers P.J.A. and Westen, S.J.P. (2007), OpenMI: Open Modelling Interface. *Journal of Hydroinformatics*, 9(3), 175-191.

Hillyer, C., Bolte, J., Van Evert, F., Lamake, A. (2003), The ModCom modular simulation system. *European Journal of Agronomy*, 18(3-4), 333-343.

Knapen, M.J.R., Athanasiadis, I.N., Jonsson, B., Huber, D., Wien, J.J.F., Rizolli, A.E., Janssen, S. (2009). Use of OpenMI in SEAMLESS. *Proceedings AgSAP Conference 2009, Egmond aan Zee, The Netherlands, 330-331*.

Moore, R. and Tindall, C. (2005), An overview of the open modeling interface and environment (OpenMI). *Environmental Science and Policy*, 8(3), 279-286.

Parker, P., Letcher, R., Jakeman, A., Beck, M., Harris, G. (2002), Progress in integrated assessment and modeling. *Environmental Modelling and Software*, 17(3), 209-217.