Conference Paper, Published Version

Gijsbers, Peter; Hummel, Stef; Vanecek, Stanislav; Groos, Jesper; Harper, Adrian; Knapen, Rob; Gregersen, Jan; Schade, Peter; Antonello, Andrea; Donchyts, Gennadii

From OpenMI 1.4 to OpenMI 2.0

International Environmental Modelling and Software Society (iEMSs)
2010 International Congress on Environmental Modelling and Software
Modelling for Environment's Sake, Fifth Biennial Meeting, Ottawa, Canada
David A. Swayne, Wanhong Yang, A. A. Voinov, A. Rizzoli, T. Filatova (Eds.)
http://www.iemss.org/iemss2010/index.php?n=Main.Proceedings

From OpenMI 1.4 to 2.0

Peter Gijsbers[1,2], Stef Hummel[1,3], Stanislav Vaneçek[1,4], Jesper Groos [1,5], Adrian Harper [1,6], Rob Knapen[1,7], Jan Gregersen [1,8], Peter Schade[1,9], Andrea Antonello[1,10], Gena Donchyts[1,11]

[1] *The OpenMI Association, Lelystad, The Netherlands*
[2] *Deltares USA Inc., Silver spring, Maryland, USA, Peter.Gijsbers@deltares-usa.us*
[3] *Stichting Deltares, Delft, The Netherlands, Stef.Hummel@deltares.nl;*
[4] *DHI, Prague, Czech Republic, S.Vanecek@dhi.cz*
[5] *DHI, Hørsholm, Denmark, jgr@dhigroup.com*
[6] *MHW Soft, Wallingford, UK Adrian.Harper@wallingfordsoftware.com*
[7] *Alterra, Wageningen, The Netherlands, Rob.Knapen@wur.nl*
[8] *Hydroinform, Kirke Hyllinge, Denmark, Gregersen@hydroinform.com*
[9] *Bundesanstalt für Wasserbau, Hamburg, German,y Peter.Schade@baw.de*
[10] *University of Trento, Italy, Andrea.Antonello@ing.unitn.it*
[11] *Stichting Deltares, Delft, The Netherlands, Gennadii.Donchyts@deltares.nl*

Abstract: The Open Modelling Interface (OpenMI) was launched end of 2005 with the aim to become a global standard for linking models and tools in the environmental domain with focus on water. Over the past few years, the user and development community has grown substantially and various well known models have become compliant. Some of the uses did not adopt the OpenMI.Standard interfaces completely, but used a slight deviation to achieve their goal in a similar style. Improvements would be necessary to become a true global interface standard instead of a style for developing new model codes. Starting in 2007, a core group of six institutes has worked on an upgrade of the OpenMI towards version 2.0. A long list of deficiencies was composed, having a few use cases as general guidance for improvement. The proposed redesign, based on similar leading concepts and a similar data model, required however a non-backward compatible upgrade of the interface standard to remove the weak points from the first version. This decision allowed the OpenMI to become more suitable for a larger range of applications, from non-time dependent Geographical Information Systems (GIS) towards e.g. master-slave controlled modelling frameworks. The OpenMI 2.0 standard has been open for review early 2010. Once completed and processed, the release of OpenMI 2.0, in both C# and Java is expected late 2010. This paper will discuss the reasons for change in more detail and highlights how the proposed solution meets the needs in a better way.

Keywords*:* Open Modelling Interface; linking; model interoperability; integrated modelling

1. THE USER COMMUNITY OF OPENMI EXPANDS

The Open Modelling Interface (OpenMI) was launched end of 2005 with the aim to become a global interface standard for linking models and tools in the water domain. A scope that later was expanded to the environmental domain. The OpenMI [Gregersen et al. 2007] was developed by a consortium existing of NERC-CEH (UK), DHI (DK), HR Wallingford/Wallingford Software (now MHW Soft, UK), Delft Hydraulics (now Deltares, NL), RIZA (now partly Deltares, NL) and many others in the HarmonIT-project, an EU 5th Framework project (Contract EVK1-CT-2001-00090). From the beginning onwards OpenMI was presented as a software interface that could be utilized to couple models or even frameworks. To stimulate uptake of the Application Programming Interface (API) standard and demonstrate its suitability, a software implementation was delivered for the API, supported with a Software Development Kit (SDK) and Graphical User Interface (GUI) to facilitate component wrapping and model coupling. Although OpenMI users are not obliged to apply these supporting products, the SDK and GUI do give many scientists

in the field the impression that OpenMI is another framework, similar to e.g. TIME, CSDMS, ESMF or OMS. The OpenMI Association has not the intention to provide a framework, but deems an open source software implementation necessary to reach world wide adaptation of the standard, given its current development and acceptance phase.

To take the OpenMI from a research output to an operationally viable technology, the OpenMI Life started under the EU-Life programme (LIFE06 ENV/UK/000409). In this project, water management agencies in Belgium, the Netherlands and Greece applied the OpenMI to connect their models from different suppliers and conduct crosscutting analysis on real world issues that require integrated assessments.

In order to support this process, a legal association was established to take ownership of the OpenMI and stimulate the use and support for the OpenMI: The OpenMI Association. After some initial hurdles to understand what it takes to deliver an open source standard and SDK, the confidence grew outside the core group that the OpenMI was a highly potential technology for the larger community. Other parties started joining the Association and contributed to the development. Alterra (part of Wageningen UR, NL) adopted the OpenMI in various EU-projects (e.g. SEAMLESS, SENSOR), developing and launching an open source Java implementation of the OpenMI, with support from Univ.Trento (I). The Bundesanstalt für Wasserbau (BAW, GER) ported the C# implementation from Microsoft .Net to the Mono platform and tested this implementation on a Linux based proprietary file based database to exchange initial and boundary data with the (Windows based) three-dimensional flow model Delft3D [Schade et al. 2008]. New public models have become OpenMI compliant (e.g. SWAT, migrated by UNESCO-IHE) while the USACE-HEC and USGS are collaborating towards an OpenMI compliant implementation of HEC-RAS and Modflow. CUAHSI has demonstrated that its WaterOneFlow web service can be connected to OpenMI compliant models [Goodall et al. 2007] and are keen to explore model-based web services using OpenMI as an important vehicle. CSDMS, has adopted the OpenMI as its model interoperability interface using the Common Component Architecture (CCA) as implementation framework for its HPC-model platform [Peckham, et al. 2009]. USDA has implemented the OpenMI interface on top of its Object modelling System (OMS) [David et al. 2009]. CSIRO is experimenting with OpenMI while US-EPA has announced that the new version of its multi-model multi-media framework, Frames 3, will be using the OpenMI interface. Finally, the Open Geospatial Consortium (OGC) shows a keen interest in OpenMI as a means towards model interoperability services. In addition to those known users, i.e. known to the Association, many other users exist. Bulatewitz et al. [2009] demonstrates that, even without questions to the help forum, people take the OpenMI concept and build integrated multi-disciplinary modelling applications with it.

Some of those uses did not adopt the OpenMI.Standard interfaces completely, but used a slightly deviation to achieve their goal in a similar style. This indicated room for improvement, an observation that was supported by experiences in the core group as well. This paper will take those experiences as a starting point to discuss in general terms the changes that have been made to make OpenMI more flexible, more efficient and more intuitive for integrated modelling applications in the wider environmental domain.

2. USER EXPERIENCES EXPRESSING A DESIRE FOR IMPROVEMENT

2.1 User experiences with OpenMI 1.x

OpenMI 1.x was developed with an extensive focus on numerical models, primarily in the field of hydraulics and groundwater-surface water interaction. Gregersen at al. [2010] demonstrate such application. Although notice was taken of the needs of other domains, e.g. economics, this hardly influenced the outcome of OpenMI version 1.

After OpenMI 1.x was released and OpenMI was being used in real practice, many successful applications have been developed. Simultaneously however, a few design decisions became apparent as being not well equipped to meet the desired flexibility and performance. An important nuisance appeared to be the link object, i.e. the interface that

defines what data is exchanged between two components. The link had no clear ownership, thus creating a confusing situation when adding data operations. Another less favourable feature of the link was the need of a full interface implementation on both sides of a link, also for potentially simple components such as data viewers. The link object also stimulated memory resource intensive implementations, as each link would typically buffer the source data it needed to support the data request from its target. Viewing the data being passed over a link to a model actually required instantiation of a second similar link, often resulting in the same data being duplicated in memory to support the second link. Especially with larger grids, the memory consumption could become substantial. The link also played a prominent role in the request for data (the GetValues method). Unfortunately, this call referred to a link by its linkId, introducing a performance hit at each call when the internal list of links became large. Finally, the link design also stimulated the use of a non-intuitive trigger object, an object which often was added to create a 'trigger' link for starting the data exchange and computation process.

Within OpenMI, data is exchanged on a request-reply basis, in other words: data is pulled to a component. This pull-concept enabled Goodall et al. [2007] to implement OpenMI as a web-service, hence demonstrating its applicability as a service-oriented architecture. However, the implementation as a modelling web-service was not intuitive as OpenMI 1.x only allowed values being exchanged per timestamp, and not as a time series object in one go [Goodall et al. in press]. Such capability would be highly desirable for web-service implementations to reduce the number of calls needed over a network.

Prototype applications with OpenMI 1.x illustrated that data assimilation [Seymour, 2005] and decision support applications [Dirksen et al., 2005, Knapen et al., 2009a, b] could be developed with OpenMI, but the pull-based concept was not very fit for purpose. These applications typically conduct multiple model runs, redefining the input after evaluation results from a previous run. A 'set value' approach would be natural, but OpenMI 1.x required an unnatural call back construct to pull new input into the engine.

OpenMI 1.x provides optional interfaces to implement state management. Unfortunately, the interface could not guarantee support for persistent state management. So far, few applications have been using state management functions, partly since its application was mostly limited to engines iterating back and forth to arrive at numerical stable conditions. Generation of multi-model restart files, desirable to prevent numerical instability of complex interacting models at start-up, could not be facilitated. The same applies to forecasting systems requiring frequent model restarts from 'warm states'.

Application in the SEAMLESS and SENSOR projects [Knapen et al. 2009a, b, Wien et al. 2009, Verweij et al. 2007] illustrated that the OpenMI data model was not very suitable for interdisciplinary modelling applications requiring exchange of non-numerical data. Furthermore, its tight connection with the concept of time as well as its over elaborated spatial data model made OpenMI 1.x not very attractive for combination with Geographic Information Systems or databases holding non-transient data.

For above reasons, various OpenMI users deviated from the Standard specification in the applications. Given these experiences, the OpenMI Association Technical Committee (OATC) proposed to the OpenMI Association Executive Board to develop a non-backward compatible upgrade to overcome some of the issues identified. This redesign, conducted over the past three years, was guided by the user experiences and an integrated modelling domain analysis to highlight the needs that had not sufficiently been met by OpenMI 1.4.

2.2 Domain analysis

The OATC started the development of the next version of the OpenMI with an analysis of experiences, focussing on domain modelling characteristics in (to be) integrated modelling disciplines. Its results are expressed in an abstract perspective focussing on spatial and temporal characteristics of various component types as illustrated in Table 1.

Table 1 Characteristics of selected component types

Component Type	Spatial characteristic	Temporal characteristic
Numerical model	ID-based and/or geo-referenced (x, y , z)	proceeds in time, may step backwards for numerical stability
Constant value provider	constant in space	constant in time
0-D time series (boundary condition) provider	ID-based	data varies over time
Geographical Information Systems	geo-referenced	constant in time
Tools for Optimization, Calibration and Scenario Management / Decision Support	with/without geo-reference	proceed in time
Analytical functions	exact geo-referenced (without interpolation)	exact in time (without interpolation)
Database or reader	with/without geo-reference	values of all time steps are available from the start

As indicated, OpenMI 1.4 was relatively strong in support of numerical models, but required considerable improvement for the other types. The major changes are:

- The link object is replaced by a direct reference to the consumers respectively producers of data.
- Exchange items get a new role as the port for data provision and consumption.
- Time has become an integrated part of the query for data.
- Progress in time is less tightly bounded to the call for data transfer.
- Enhanced support for non-numerical data.

3. CHANGES FROM OPENMI 1.4 TO 2.0

3.1 Replacing the link - A new role for exchange items

In OpenMI 1.4 the link object was used to connect components, containing a reference to the source component and the target component, as well as to the source output (exchange) item and the target input (exchange) item. Exchange items had been introduced to describe the data being exchanged in terms of 'what' (IQuantity) and 'where' (IElementSet). To make output items compatible with input items, a series of data operations (IDataOperation) was specified in the link object. Experience has shown that the link object was not very efficient and intuitive for many applications. 'Ownership' of the object was unclear, therefore creating confusion how additional data operations could be implemented.

Adopting a port concept would make the addition of data operations much more intuitive, especially if a chain of data operations could be implemented on top of a port. In addition, a port concept would make connecting a (non-compliant) data viewer to an OpenMI component much easier. Many model providers used the internal exchange item object for holding the data that the IExchangeItem interface described. This interface thus would offer an intuitive place to act as a port for data provision and consumption. Two types are identified with different tasks. InputItems provide an entry point for data input, while OutputItems provide an exit point for output. Data viewers only need to implement the input interface such that they can register themselves as a consumer to output of another component. Data viewers thus do not need to implement all data providing capabilities.

In OpenMI 2.0 a connection between a source component and a target component is realized by a direct reference between an output item and an input item. The GetValues method, called for actual data transfer, has been moved from the LinkableComponent level to the exchange item level. The link, as a separate interface, has been removed from the specification. If the values of an output do not exactly fits the required input, an 'adapted' output needs to be provided. The adapter pattern (similar to a decorator pattern) is used for

this purpose, enabling the addition of data-operations in between the original output and input item, realizing a truly piping and filtering pattern.

The adapted outputs take over the role of data operations in OpenMI 1.4. Typically, such intermediate adapted outputs are spatial and time interpolations and unit conversions. The sequence of adapted outputs defines explicitly in which order the operations are carried out. The adapted output can, via the Refresh() method, be notified that the values of its parent output may have been changed.

An important advantage of the ability to 'stack output adaptations' is the ability to reuse adapted outputs for various consumers (Figure 1). E.g. a model requiring point input data at to different locations from a grid model, forced the grid model - under OpenMI 1.4 - to hold two internal copies of the grid data. With version 2.0, the source model only needs to hold an internal copy of the interpolated points. Similarly, data viewers monitoring a connection do not enforce a duplication of the link (and its data) anymore, as they can directly tap the same adapted output as the direct model-to-model connection.

Finally, the adapted output mechanism enables the user to define the order of a series of operations. By performing the operation with the greatest data reduction first, the subsequent operations can significantly be sped up.

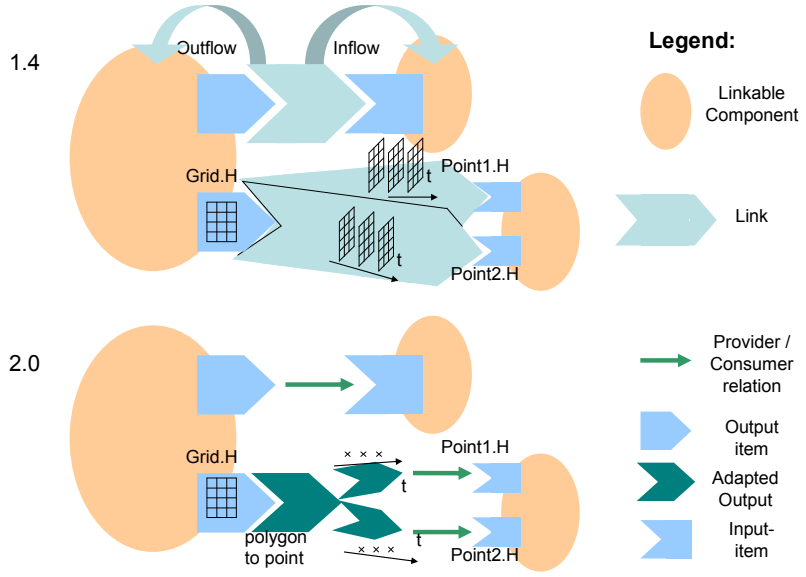

Figure 1 Re-use of adapted outputs

3.2 Reformulating the query for values

In OpenMI 1.4, values were asked from a linkable (model) component for a certain time and for a certain link through the GetValues call. The component had the responsibility to return the values asked as specified by the target quantity and target element set of the link. Components that had no temporal knowledge still were enforced to deal with time. This annoyance has been removed.

In OpenMI 2.0, the link identifier is gone, as a direct relation between producers and consumers defines the connection between components. The query specifies the value definition and either the element set (space), the time set or both. The query is passed as an InputItem argument in the GetValues call on the OutputItem. In this manner, all variations in component types - as identified in the domain analysis - can be supported with their specific query needs. Components dealing with time can be informed beforehand at what time stamps the consumer requires data. This allows a providing component to prepare itself e.g. by creating appropriate internal buffers or by computing ahead if possible.

3.3 Redefining the data model

Users requests have also resulted in a some adjustments and extensions of the data model in order to accommodate non-numerical data, to improve interoperability with OGC-feature types and to enable retrieval of time series objects in one go.

To support non-numerical data, the values have changed from doubles - representing a numerical scalar or (xyz) vector - to objects. The value definition has been extended from numerical quantities to both quantities and so-called qualities, the latter being categorized items enabling representation of e.g. soil types, land use, seasons and fuzzy values.

The spatial domain of an OpenMI component is described by the element set. IElementSet is an overarching interface allowing high-performance access to individual elements of a specific element type. The list of element types is GIS-oriented, but the separation between the horizontal and vertical dimension, as defined in OpenMI 1.4, showed to be overdone. Both the GIS-world and the numerical modelling world did not fit into this design. Hence, the number of element types has been reduced and is harmonized with the feature types as common in the GIS-world. For numerical modellers, both Z-coordinates (geo-referenced) and M-coordinates (model layer reference) are supported.

The actual data values being exchanged are held in a so-called value set. For OpenMI 2.0, the IValueSet interface has been redesigned as well. New methods have been introduced to retrieve the value objects for all element values for a specific time or for all time series values of a specific element. In addition, the interface provides a method to set values into the object. This method, combined with the accessibility of a value set as a property of an input exchange item, allows external components to set a value if needed. Tools used for data assimilation, calibration and optimization will benefit from this capability, while the dominant data exchange mechanism remains pull-based.

3.4 Extending control flow options

In OpenMI 1.4, the GetValues call served various purposes: data transfer, progress in time and triggering calculations at other components when needed to progress. The control flow was purely pull driven, since linkable components were triggered to perform an action when there was a request to produce data.

In OpenMI 2.0 this is pull mechanism is still the same, with components actively triggering other components when they need data. However, a second control flow option, called the ‘loop’ approach, has been added to accommodate progress control from the outside. An external controller can invoke each linkable component separately, by calling its Update() method. This will let the linkable component progress to its next step, but only if all necessary data for that component is available. If the data is not available yet – e.g. because another component has not computed it yet - the component specifies its state as "waiting for data", and returns the control flow. During the next Update() call, the component checks whether the required data is available, (and if so) computes the new value and sets its state to "updated". A typical application of this approach could be master-slave computational system where a master component controls the computation sequence of its various computational domains enabling parallel computing.

Note that the loop approach doesn’t offer wider usage of linked components, it is only an alternative way of controlling the data flow. Implementing the loop approach is optional. A linkable component that does not support it is still OpenMI compliant.

3.5 Enabling persistent states

While OpenMI 1.4 allowed models to manage their state and roll back to a previous state, it was up to the developer to implement this in memory or with so-called restart files. Various use cases would benefit if OpenMI could enforce a persistent state to be saved and restored. A complex model combination could reduce its numerical instability problems at start-up if the various components could save a restart file on request when the shared system is in balance. In addition, operational forecasting systems need this ability to keep the latest

model state update-to-date. OpenMI 2.0 therefore accommodates conversion of a model state from a memory object to a persistent byte stream and back.

3.6 Improving component status information

OpenMI 1.4 contained a message mechanism to broadcast component status. This design has shown to impact performance as the composition of messages could take substantial time as compared to the calculation time. In OpenMI 2.0, status is handled as property of the linkable component. The states can be used in both the pull-based as the loop-based control flow to make decisions. The available states can be defined in two types: action states (e.g. updating) and idle states (e.g. updated). States are related to the call sequence conducted on the component.

4. CONCLUSIONS AND FINAL REMARKS

The redesign presented in this paper, supported with more technical details in Donchyts et al. (2010), is expected to be a major step forward towards improving the applicability of OpenMI as an Application Programming Interface for interoperability between computational models, databases, geographic information systems and web-services. Its updated interface specification is much more in line with the actual implementation at computational cores, while it simultaneously has improved capabilities for utilization in GIS or web service environments.

However, not all requests and wishes are supported yet. Additional funding and resources will be needed to test the OpenMI 2.0 interface design in a High Performance Computing environment in order to decide if extensions are essential. Similarly, more experience needs to be gained in the utilization of ontology in combination with OpenMI 2.0, based on the initial steps taken in the SEAMLESS project [Knapen, 2009a,b]. Furthermore, new web-service experience needs to be gained to leverage the capabilities of the OpenMI in this field if needed.

The expectation is that extending the available or implementing new Software Development Kits can facilitate most of these requests. Most likely, some extension will be required to the interface standard to accommodate specific needs. However, the current interface design is foreseen to be sufficiently mature, such that any of those requests does not need to result in another non-backward compatible upgrade.

ACKNOWLEDGEMENTS

The development of OpenMI version 2.0 has been funded by the members of the OpenMI Association as well as the European Commission through various 6th Framework Programs and the LIFE Environment program.

REFERENCES

Bulatewicz, T., X. Yang, J. M. Peterson, S. Staggenborg, S. M. Welch, and D. R. Steward (2009) Accessible integration of agriculture, groundwater, and economic models using the Open Modeling Interface (OpenMI): methodology and initial results. *Hydrol. Earth Syst. Sci. Discuss*., 6, 7213–7246, 2009

David, O, Lloyd, W, Carlson, J, Leavesley, G H, Geter, F (2009) Use of Annotations for Component and Framework Interoperability *Eos Trans. AGU, 90(52)* Fall Meeting Abstracts Suppl., abstract #IN11A-1044

Dirksen P.W., Blind M.W., Bomhof T. and Srikrisnhudu Nagandla, (2006) Proof of Concept of OpenMI for Visual DSS Development, *Modsim 2006 International Congres on Modelling and Simulation, Melbourne, Australia*, pp.184-189

Donchyts, G., Hummel S., Vaneçek S., Groos J., Harper A., Knapen R., Gregersen J. Schade P., Antonelli A, Gijsbers P. (2010) OpenMI 2.0 What's New. *This conference, iEMSs 2010 Fifth Biennial Meeting, Ottawa, Canada*

Goodall, J. L.; Robinson, B. F.; Shatnawi, F. M. and Castronova, A. M. (2007) Linking hydrologic models and data: The OpenMI approach, *Eos Trans. AGU, 88(52)* Fall Meeting Abstracts Suppl., abstract #H13H-1682

Goodall, J.L., Robinson, B., and Castronova, A.M. (in press), Modeling complex water resource systems with service-oriented computing. *Submitted to Environmental Modelling & Software*

Gregersen, J.B. , P.J.A. Gijsbers and S.J.P. Westen (2007) OpenMI: Open modelling interface, *Journal of Hydroinformatics* Vol.9 No 3. pp 175–191

Knapen, M.J.R.; Verweij, P.J.F.M.; Wien, J.J.F.; Hummel, S. (2009a) OpenMI - The universal glue for integrated modelling? *In: 18th World IMACS Congress and MODSIM09 International Congress on Modelling and Simulation, Cairns, Australia*

Knapen, M.J.R.; Athanasiadis, I.N.; Jonsson, B.; Huber, D.; Wien, J.J.F.; Rizzoli, A.E.; Janssen, S. (2009b) Use of OpenMI in SEAMLESS. *In: Proceedings of the Conference on Integrated Assessment of Agriculture and Sustainable Development: Setting the Agenda for Science and Policy (AgSAP 2009). Egmond aan Zee, The Netherlands*

Peckham S. D., Hutton E. (2009) Componentizing, standardizing and visualizing: How CSDMS is building a new system for integrated modeling from open-source tools and standards *Eos Trans. AGU, 90(52)* Fall Meeting Abstracts Suppl., abstract #IN11A-1045

Schade, P.; Lang, G. and Jürges, J.; (2008) OpenMI Compliant Import of Initial and Boundary Data into a numerical 3D Model. *In: Proceedings of the iEMSs Fourth Biennial Meeting: International Congress on Environmental Modelling and Software (iEMSs 2008)* Vol. 2, pp 1086-1093

Seyoum S (2005), A generic software tool for OpenMI-compliant Ensemble Kalman Filtering, *M.Sc thesis, April 2005, UNESCO-IHE, Delft, The Netherlands*

Verweij, P.J.F.M.; Knapen, M.J.R.; Wien, J.J.F. (2007) The Use of OpenMI in Model Based Integrated Assessments *In: MODSIM 2007 International Congress on Modelling and Simulation,Christchurch, New Zealand, December 2007*

Wien, J.J.F.; Rizzoli, A.E.; Knapen, M.J.R.; Athanasiadis, I.N.; Janssen, S.; Ruinelli, L.; Villa, F.; Svensson, M.; Wallman, P.; Jonsson, B. (2009) Architecture of SEAMLESS-IF as framework *In: Integr. Assessm. of Agriculture and Sustainable Development; Setting the Agenda for Science and Policy (AgSAP 2009), Egmond aan Zee, The Netherlands*