International Environmental Modelling and Software Society (iEMSs)
2012 International Congress on Environmental Modelling and Software
Managing Resources of a Limited Planet, Sixth Biennial Meeting, Leipzig, Germany
R. Seppelt, A.A. Voinov, S. Lange, D. Bankamp (Eds.)
http://www.iemss.org/society/index.php/iemss-2012-proceedings

Integrating OpenMI and UncertWeb: Managing Uncertainty in OpenMI models

Tushar Gupta [a], **Richard Jones**[b], **Lucy Bastin**[b] **and Dan Cornford**[b]

[a] *Computer Science Department, Indian Institute of Technology, Ropar, India (tushargupta51@gmail.com)*

[b] *Computer Science, Aston University, Birmingham, UK (jonesrm1@aston.ac.uk, l.bastin@aston.ac.uk, d.cornford@aston.ac.uk)*

Abstract: OpenMI is a widely used standard allowing exchange of data between integrated models, which has mostly been applied to dynamic, deterministic models. Within the FP7 UncertWeb project we are developing mechanisms and tools to support the management of uncertainty in environmental models. In this paper we explore the integration of the UncertWeb framework with OpenMI, to assess the issues that arise when propagating uncertainty in OpenMI model compositions, and the degree of integration possible with UncertWeb tools. In particular we develop an uncertainty-enabled model for a simple Lotka-Volterra system with an interface conforming to the OpenMI standard, exploring uncertainty in the initial predator and prey levels, and the parameters of the model equations. We use the Elicitator tool developed within UncertWeb to identify the initial condition uncertainties, and show how these can be integrated, using UncertML, with simple Monte Carlo propagation mechanisms. The mediators we develop for OpenMI models are generic and produce standard Web services that expose the OpenMI models to a Web based framework. We discuss what further work is needed to allow a more complete system to be developed and show how this might be used practically.

Keywords: OpenMI; Web services; uncertainty; visualisation; models

1 BACKGROUND

UncertWeb is an European Commission (EC) funded project aiming to build the uncertainty-enabled Model Web[1]. The Model Web is a developing concept for a dynamic network of computer models that, together, can answer more questions than the individual models operating alone [Geller and Turner, 2007]. It is based on a philosophy that encourages modellers to provide access to their models and associated outputs through standard Web service interfaces, making it easier to discover and use models, and also enabling models to exchange information [Pebesma et al., 2010]. Models which are exposed as Web services can be re-used and integrated to provide new composite models, known as service chains or workflows. Within these workflows, component models can be modified and replaced providing that the interfaces remain fixed. There are many sources of uncertainty in such workflows which can affect the reliability and usability of the ultimate results. The UncertWeb project addresses how to compose these workflows within a Global Earth Observation System of Systems (GEOSS)[2] or Web service context, accounting for these uncertainties. To achieve this objective, the project is developing various tools to support uncertainty handling.

[1] See `www.uncertweb.org`
[2] `www.earthobservations.org`

There are a number of existing modelling frameworks available to build environmental models and complex workflows [Bastin et al., 2012]. A number of these frameworks are able to include Web-based models if they conform to certain interface standards. Supporting these frameworks are a variety of tools and standards which can be used to expose processes (i.e., models) on the Web in formats which make them discoverable and re-usable. As part of the UncertWeb project, a generic Web service framework was developed to facilitate the exposure of processes on the Web [Jones et al., 2012]. In this work we evaluate an existing, generic modelling interface standard, and assess how far it might be used in combination with the tools and architectures developed in UncertWeb. The standard we chose to evaluate was OpenMI [Jagers, 2010].

The OpenMI association[3] has created a generic modelling framework which has strong potential to contribute to the development of the Model Web. OpenMI is described at two levels. On the user level, OpenMI provides a standard interface allowing models to exchange data with each other and other modeling tools on a time–step–by–time–step basis, thus facilitating the modelling of process interactions. On a technical level, the OpenMI standard is a software interface definition for the computational core (the engine) of the models. Currently, the majority of models employing the OpenMI interface are in the hydrology domain. Model components that comply with this standard can, without any programming, be configured to exchange data during execution. The standardised interface is used to define, describe and transfer data between software components that run simultaneously, thus supporting systems where feedback between the modelled processes is necessary in order to achieve physically sound results [Gregersen et al., 2007]. This feedback mechanism provided a motivation for adopting OpenMI within UncertWeb, as the interface promised to allow bi-directional data exchange between participating models. Models previously considered within UncertWeb were primarily chained models with data flow occurring in only one direction. The OpenMI framework is realised in Java and C# interfaces available under an open-source MIT license. Our experiments were implemented using Java interfaces for OpenMI version 1.4.

The UncertWeb Processing Service is a generic framework for exposing processing functionality on the Web [Jones et al., 2012], and is designed to develop the Open Geospatial Consortium Web Processing Service[4] to a more generic and usable standard. The framework aims to reduce the complexity of exposing a process on the Web by automatically selecting encoding formats, producing service descriptions, and handling requests. A developer is only required to specify the identifiers, Java classes, and multiplicity of the inputs and outputs of their process. This allows them to focus on process functionality, rather than the mechanisms for exposure on the Web. Processes exposed using the framework are available through both a SOAP/WSDL (Simple Object Access Protocol/ Web Services Description Language) and JSON-based (JavaScript Object Notation) interface, allowing for flexibility when creating client applications. In addition to providing a WSDL document, the service uses a fixed message pattern for process requests and responses.

UncertWeb consists of a number of components, such as the processing service and related encoding profiles. The primary focus of the UncertWeb project is the management and use of uncertainty information in the Model Web. Model inputs are subject to various forms of uncertainty, which propagate through workflows and interact with further uncertainties inherent in the component models. This leads to uncertain outputs, whose reliability must be quantified and communicated to users in a usable form, if any robust decision-making is to be based on those outputs. Therefore, the framework has built-in support for UncertML [5], a dictionary and set of encodings designed to represent

[3] `www.openmi.org`
[4] `www.opengeospatial.org/standards/wps`
[5] `www.uncertml.org`

quantitative uncertainty in an interoperable manner. For describing geospatial data sets, the UncertWeb Geography Markup Language (GML) and Observations and Measurements (O&M) profiles [Stasch, 2011] are supported. These profiles restrict the elements available from the base GML and O&M schema. Without these sets of restrictions, it is possible to specify an O&M observation as being of any type, making client and service development difficult.

This paper discusses how the OpenMI modelling interface was integrated into the UncertWeb Processing Service. The existing OpenMI standard has no explicit means of representing uncertainty on model inputs and outputs, and so this was the primary challenge in integrating the two. The models considered were simple models, both time-stepping and non-time stepping, with numerically–characterised uncertainty on at least some of their inputs.

2 OPENMI AND UNCERTWEB

OpenMI v1.4 is based on the request–and–reply mechanism. The pull-based pipe-and-filter architecture consists of components (source components and target components) that can be linked together to exchange memory-based data in a predefined way and in a predefined format. OpenMI defines the component interfaces, as well as the way in which the data is exchanged. The exchange takes place in a single-threaded manner where a component can handle only one data request at a time. The transfer of information is triggered by the target components of the chain. Once started, the data is transferred through the chained components, some of which may perform their own computation on the data before forwarding the requested results. The main interfaces in the OpenMI standard are shown in Table 1.

Table 1: OpenMI interface descriptions.

LinkableComponent	standard interface for engine components that OpenMI-compliant engine components must implement.
Link	holds reference to the two linked components, contains information about what is requested, where the requested values apply and how the requested data should be calculated.
Quantity	defines what should be retrieved, represented as a text string.
ElementSet	defines where the retrieved values must be used.
ExchangeItem	each exchange item contains a Quantity and an ElementSet describing what can be accepted/provided at which location.
GetValues	one LinkableComponent invokes the GetValues method of another LinkableComponent, the source LinkableComponent must return the values for the specified quantity, at the specified time stamp or time span and at the specified location.

The UncertWeb Processing Service framework exposes processes on the Web as 'Web methods'. Each process must implement the `AbstractProcess` class, enabling it to be parsed and exposed by the framework. The methods defined for each process are listed in Table 2.

Table 2: Method descriptions for the `AbstractProcess` class.

getIdentifiers	returns the name of the process.
getInputIdentifiers	returns a list of strings detailing the input identifiers.
getOutputIdentifiers	returns a list of strings detailing the output identifiers.
getInputDataDescription	returns a data description object for the input.
getOutputDataDescription	returns a data description object for the output.
run	execute the computation in this method

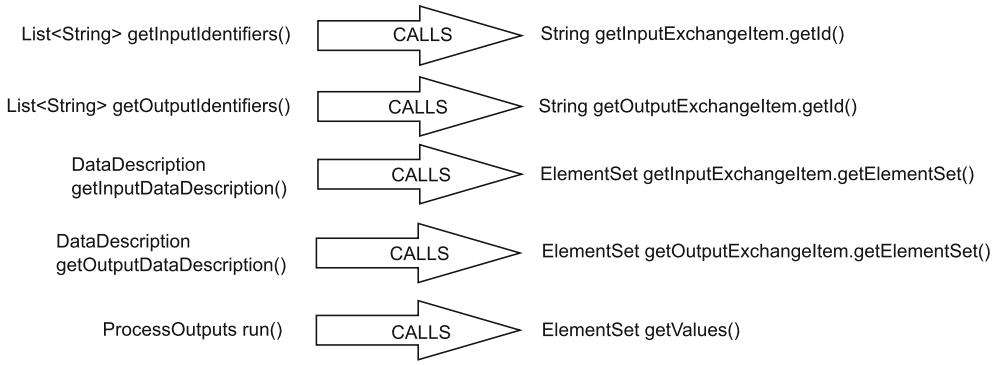

Figure 1: The mapping of the two frameworks.

In OpenMI, models exchange values through calls to the *getValues()* methods. One model can access the output identifiers of another model, and before the exchange can take place, it makes sure that they are compliant with the data type of its input identifiers. Subsequently the two models can proceed towards the exchange, using the link through which they are connected.

3 IMPLEMENTATION

`AbstractProcess` is the core class for creating processes in the UncertWeb Processing Service framework. All the processes which are exposed on the Web must implement the methods declared in the abstract class. In the OpenMI standard, the interfaces exchange information between each other. Ideally, the methods of both `AbstractProcess` and the OpenMI interfaces would be implemented in a single integrated model and process class. As the limitations of multiple inheritance in Java made this impossible, a wrapping approach was taken.

OpenMI components were created and linked inside the process itself. The process class is then responsible for communicating with the underlying model, and the processing service framework handles client interaction through the Web interface automatically. We defined a generic process class where the six abstract methods described in Table 2 were defined using the methods of the OpenMI interfaces. This mapping between methods of the OpenMI and UncertWeb components is shown in Figure 1. The code listing below demonstrates an example mapping[6] for `getInputIdentifiers`, where the relevant method in the OpenMI interface is called inside the method of the `AbstractProcess` class.

[6] Full source code available at http://www.github.com/tushargupta51

Listing 1: Code fragment for the `getInputIdentifiers()` method.

```
public List<String> getOutputIdentifiers() {
        int outputcnt = sum1.getOutputExchangeItemCount();
        String[] output_list = new String[outputcnt];
        for(int i=0;i<outputcnt;i++) {
                output_list[i] = sum1.getOutputExchangeItem(i).getID();
        }
        return Arrays.asList(output_list);
}
```

This approach required no changes to OpenMI model definitions or implementation, providing the potential for a 'plug and play' architecture, where an OpenMI model can be exposed as a Web service without any further configuration. The consistency among the data types of the methods defined in OpenMI models and UncertWeb Processing Service framework eased the process of mapping between the two, as follows:

- The input/output identifiers defined in the `AbstractProcess` were all strings and the corresponding identifiers defined in input/output `ExchangeItems` were also strings, hence a straight-forward mapping was possible;
- The `DataDescription` object returned by the data description methods contains information about the data type of inputs/outputs. The `ElementSet` contained in the input/output `ExchangeItems` describes the type of values exchanged and is passed as a parameter to the returning `DataDescription` object.
- The `run` method triggers the computation in a process. The corresponding method in OpenMI models is `getValues()` but the data types of both the methods are different. The `run` method returns the `ProcessOutputs` object which contains `Output` objects - these can be `singleOutput` or `multipleOutput`, and each contains an Object type member. This member corresponds to the `ElementSet` members returned by the `getValues` method. Thus inside the `run` method, a `getValues` method is called. This returns a `ValueSet` which is then used to create a Single/Multiple `Output` object. This `Output` object is then added to the `ProcessOutputs` object and returned by the `run` method.

4 EXAMPLE APPLICATION

With the OpenMI models exposed on the Web using the UncertWeb Processing Service framework, a set of test simulations could be executed. For this, we developed a client to validate the implementation. The two example applications developed were a simple summing component and a more complicated component implementing Lotka-Volterra equations on a time-step basis. The Lotka-Volterra model, used to model predator-prey relationships in terms of population numbers, involves two simultaneous differential equations [Brauer and Castillo-Chavez, 2001]. The equations include four parameters indicating the relationships between the two species:

$$\frac{dx}{dt} = x(a - by) \tag{1}$$

$$\frac{dy}{dt} = -y(c - dx) \tag{2}$$

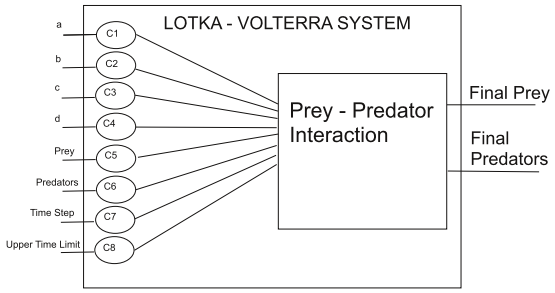

Figure 2: Lotka-Volterra system.

where x denotes the number of prey individuals, y the number of predator individuals, a the prey population growth rate, b the prey death rate due to predators, c the predator death rate and d predator growth due to feeding on prey.

In terms of uncertainty, we looked at several scenarios, including: uncertain initial conditions only (fixed parameters); fixed initial conditions and uncertain parameters (sampled from a distribution at the start of an integration, but then held fixed for the duration of the simulation); uncertainty on both (the most realistic scenario).

The Lotka-Volterra system was modelled as a single component. The input parameters and initial numbers of prey and predators were fed into the component. These values were used to solve the above equations, using the Runge-Kutta method [Butcher, 2003]. For this example, the model chain consisted of the constant components linked to the main model component which was doing the computation. The inputs and outputs were all double-precision values and are shown in Figure 2.

Constant components are similar to model components except that they do not have an input `ExchangeItem`. They have a single output, the value of which is the one with which the components are initialised at construction. Model component have inputs which get values from other components through the links with which they are connected. Since the first component inputs are not connected to other components, constant components are used to initialise the model chain.

Table 3: Inputs to Lotka-Volterra System.

Parameter	**Quartiles(Lower, 0.25, 0.5,0.75, Upper)**	**Distribution.**
Initial Prey	(200, 305, 555, 830, 1000)	lognormal(6.21, 0.59).
Initial Predators	(10, 80, 220, 345, 350)	lognormal(5.11, 0.83).
a	(0, 0.04, 0.2, 0.35, 0.5)	lognormal(-1.57, 0.81).
b	(0, 0.1, 0.25, 0.46, 0.5)	lognormal(-1.64, 0.81).
c	(0, 0.01, 0.05, 0.09, 0.1)	lognormal(-3.17, 0.85).
d	(0, 0.01, 0.04, 0.06, 0.1)	lognormal(-4.23, 0.78).

A Web-based client was developed to control execution of the model chain. An interface is provided to enable the user to enter input values, including initial parameters and beliefs. If required, these values can be used to elicit the distributions. Once the inputs (including those sampled) have been specified, a SOAP request is sent asynchronously

to the processing service interface using jQuery[7]. Upon receiving a response, the output is extracted from the SOAP message and displayed using the simple visualisation client. The description of the distributions of the input parameters is shown in Table 3.

The result of one of the iterations inside the Lotka-Volterra system is depicted in Figures 3 and 4. From the figures, it is evident that the number of prey individuals decreases exponentially whereas predator number increases initially, and then decreases due to reduced numbers of prey to feed upon.

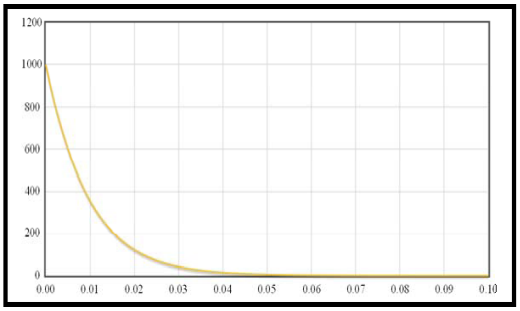

Figure 3: Example of a single simulation of prey: x-axis simulation time, y-axis prey numbers

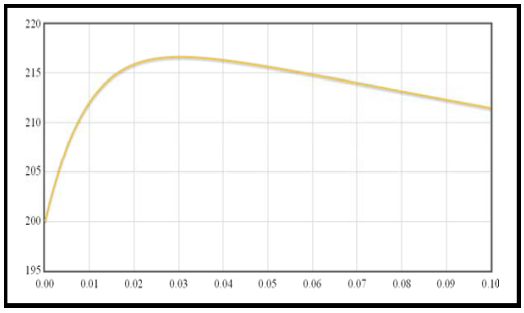

Figure 4: Example of a single simulation of predators: x-axis simulation time, y-axis predator numbers

5 Evaluation and Conclusions

This paper demonstrates the successful integration of the OpenMI and UncertWeb frameworks. The implementation maintains a generic nature, creating the potential for many other OpenMI models to be exposed on the Web in this manner. Each OpenMI component is only loosely coupled to a Web service process, making it possible to modify the model (although not its interface) without having to update the Web service interface. Once integrated, simulations based on sample OpenMI compositions were performed. To validate these, a client was developed to communicate with the Web service interface and execute component methods for the user. Values received from the Web service embedded in a SOAP message were as expected, indicating the valid integration of two frameworks. The wrappers developed are generic in terms that they can be easily updated to accommodate changes in either the OpenMI standard or the UncertWeb framework with minor modifications.

We also considered the issues that arose while propagating uncertainty through OpenMI models using tools developed within UncertWeb. An uncertainty–enabled example composition modelling the Lotka-Volterra system was developed, within which we ran several Monte-Carlo simulations with differing uncertainties in parameters and initial predator-prey values. By exposing the OpenMI models as Web services they are able to employ and benefit from the tools being developed in UncertWeb.

Future work needs to extend the range of mappings between data types in the two frameworks, and should also consider in more detail the computational issues of model integration using Web service technology and the associated communication overheads, which will be particularly significant for the time-stepping models typically deployed using

[7] http://jquery.com/

OpenMI. A logical next step would be to explore the use of the adapter with a larger range of more realistic and complex models.

ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° [248488].

REFERENCES

Bastin, L., D. Cornford, R. Jones, G. B. Heuvelink, C. Stasch, E. Pebesma, S. Nativi, P. Mazzetti, and M. Williams. Managing uncertainty in integrated environmental modelling frameworks: The UncertWeb framework. *Environmental Modelling and Software*, in press, 2012.

Brauer, F. and C. Castillo-Chavez. *Mathematical Models in Population Biology and Epidemiology*. Texts in Applied Mathematics v. 40. Springer-Verlag, 2001.

Butcher, J. C. *Numerical methods for ordinary differential equations*. John Wiley and Sons, 2003.

Geller, G. and W. Turner. The model web: a concept for ecological forecasting. In *Geoscience and Remote Sensing Symposium (IGARSS 2007). IEEE International*, pages 2469–2472, July 23-27 2007.

Gregersen, J., P. Gijsbers, and S. Westen. OpenMI : Open Modelling Interface. *Journal of Hydroinformatics*, 9:175–191, 2007.

Jagers, H. Linking data, models and tools: an overview. In *Proceedings of iEMSs (International Environmental Modelling and Software Society)*, July 5-8 2010.

Jones, R., D. Cornford, and L. Bastin. UncertWeb Processing Service: Making models easier to access on the web. *Transactions in GIS*, in press, 2012.

Pebesma, E., D. Cornford, S. Nativi, and C. Stasch. The uncertainty enabled model web (UncertWeb). In *Proceedings of EnviroInfo*, October 5-8 2010.

Stasch, C. UncertWeb O&M Profile. In *Proceedings of the OGC TC Meeting, SWE Domain Working Group*, March 1 2011.