A Framework for Modelling Multiple Resource Management Issues - an Open Modelling Approach

Michael Reed*, Susan M. Cuddy*, Andrea E. Rizzoli†
* CSIRO Land and Water, Canberra, Australia
{Michael.Reed, Susan.Cuddy@cbr.clw.csiro.au}
† IDSIA, Instituto Dalle Molle di Studi sull'Intelligenza Artificiale
Lugano, Switzerland
{andrea@idsia.ch}

Abstract Our experience with development and distribution of environmental decision support systems (EDSSs) for delivering research outcomes to catchment managers has identified major impediments to their adoption. These include protracted development time, focus on single impacts, difficulty in combining results with other EDSSs, and no attention to socio-economic factors. Based on the premise that decision support tools can play a major role in the integration and adoption of research outcomes, we are developing a software tool for rapid building of EDSSs which can handle multiple issues across different scales. The prototype is called the Open Modelling Engine (OME). This paper describes the background to the development of the OME, its basic architecture, an OME-built EDSS for catchment nutrient management, and concludes with a discussion on research direction and opportunity.

1. INTRODUCTION

During the past few years, we have been developing a tool to build Environmental Decision Support Systems (EDSSs) to simulate multiple processes in catchments. The need for such a tool, currently called the Open Modelling Engine (OME), emerged from our experience with the Catchment Management Support System (CMSS) software (Davis and Farley, 1997). This software predicts the likely impact on water quality from changes in land use and/or land management in a catchment. The impact is predicted as a change in average annual loads of nutrients, usually total phosphorus and total nitrogen, reaching the catchment's streams. The EDSS contains two very simple models: one to predict nutrient generation from land uses and one to predict instream assimilation. Such simple models have proved to be most appropriate when data quality and data coverage is poor.

This has been appreciated by water resource managers in Australia and CMSS has been adopted by several state agencies as part of their water management planning process. In particular, the New South Wales Department of Water Resources (now the Department of Land and Water Conservation) conducts a CMSS catchment study as one of the steps in assisting catchment communities develop their nutrient management plans. CMSS studies have now been completed for most of the inland basins in New South Wales. (The study for the Murrumbidgee catchment is documented in Cuddy *et al.*, 1997). However, things have changed since CMSS was adopted in the early 90s.

Firstly, our understanding of environmental processes and responses is constantly being improved by new research findings. EDSSs must be able to incorporate these new research results. Secondly, the interactions between different processes in a catchment cannot be ignored and catchment managers are demanding a holistic view of the world to assist them identify potential outcomes of management decisions. Finally, EDSSs have tended to focus on single impacts (in the case of CMSS, water quality of conservative nutrients). Integrated management requires integrated decision making. For this, EDSSs require the ability to handle multiple models and the ability to couple or integrate their predictions.

Thus the purpose of the OME is to provide a generic framework for developing spatial modelling systems that address single or multiple resource management issues.

2. THE OPEN MODELLING ENGINE

2.1 Core Architecture

The OME is a software architecture for model management (Rizzoli, 1994a). It is highly influenced by Systems Theory concepts (some pertinent references are Guariso and Werthner [1989] and Zeigler [1989]). A model of a system is

characterised by its input, state, and output variables, and by its parameters. Input, state and output variables are related by functional relationships which map the input variables to the outputs. This modelling paradigm is widely used and is particularly suited to represent environmental catchment processes. Many classes of models (functional, declarative, constraint-based, see for instance Fishwick, 1995) lend themselves to this structure. These definitions can be implemented using Object Technology to facilitate essential functions such as model prototyping, data access and model integration.

The core elements of the OME structure are:

- **Domain Objects (DObjects)**: Generic components which represent a class of object. These will be more fully described after the other core elements are introduced.
- **Models**: Mathematical equations whose variables are classified as output, input or state variables. The equations transform the inputs into outputs, using states and parameters (written using a reduced-set programming language similar to C).
- **Class templates**: Definitions of the structure of DObjects, including data and models (similar concept to Minsky's [1975] frames).
- **Data templates**: Definitions of the structure of DObject data. There are usually many data templates for each class template.
- **Data instances**: The data for each DObject, based on the data templates. These can be scalar or matrix in size.
- **Model-to-DObject links**: Connections between model variables and DObject data instances. These links allow model exchangeability (see Section 2.3).
- **DObject-to-DObject links**: Connections between data instances in different DObjects. These links allow DObjects to be connected together in a hierarchy.
- **Parser**: A powerful interpreter which processes each model and the data links in the hierarchy. Each model is executed and the results saved in a database for future display or processing.

This approach to the design of the OME was taken after consideration of the requirements of the domain of interest, namely modelling water quality at catchment scale.

Thus, each DObject contains data instances which describe the DObject's parameters, inputs and outputs. These data are mapped via Model-to-DObject links to model variables (see Figure 1).

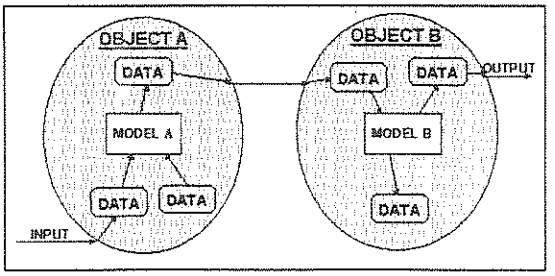

Figure 1, DObjects showing object, data, model and link relationships

This structure allows the problem to be defined by using DObjects to represent different parts of the catchment domain (e.g. streams, sources, sinks).

Models are described in the OME as symbolic equations. A model to simulate the DObject's operation can be chosen from a library, and links between the model's symbols and the DObject's data instances are made to provide the model with real data.

2.2 Model Connectivity

Connectivity between models is central to the OME. DObjects may be linked to each other (parent/child relationship), data instances may be linked to each other (data transfer between DObjects in a many-to-many relationship), and data instances may be linked to models (connecting the model to the DObject). This connectivity allows the results of one model (associated with a DObject) to be input into the next model.

A simple example would be two DObjects which describe different sub-catchments. These sub-catchments are different in terms of landscape and transport processes. Different models (e.g. rainfall/runoff models) can be associated with each DObject. The residual runoff from one DObject can be passed as input to the next DObject.

2.3 Model Interchangeability

One advantage of using the OME is the ability to interchange models. Different models may be linked to the same DObject and associated data, allowing the modeller to begin with a simple model, and expand that model as more data become available. The previous model versions can be stored away in a library, ready for reuse at any time. An example of interchangeability is given in Section 3.3.

Having described the domain and the models, the next step is to describe how the OME is executed to produce model outputs.

2.4 The Interpreter

The OME has a built-in parser which is capable of processing most mathematical expressions and supports C-like syntax for the definition of model equations. Features such as programming control structures and matrix operations are available. The parser processes each DObject, executing its associated model using the values stored in the data instances, and performs data connections between DObjects. The results are then saved for later processing and reporting. The current implementation of the parser only handles static models, but methods for simulating time-varying models are planned for implementation.

The interpreter used in OME is a MKS [1997] Lex & Yacc™ configuration with a partially customized C grammar. Models are defined as mathematical functions, such as:

$$\text{Output} = \text{Local} * \text{Factor} + \text{Input} \qquad (1)$$

The interpreter processes each token (e.g. "Output", "=", "Input") and builds a parse-tree to be executed. The above example would produce the following parse tree (Figure 2):

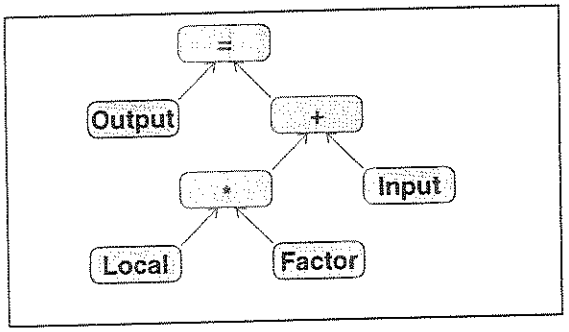

Figure 2, Interpreter Parse Tree

The interpreter moves through the tree, from bottom to top, calculating the results at each step, before calculating the value of "Output". When the interpreter is finished with each DObject, the results are written back to the relevant data instances, and the next DObject processed.

3. WINCMSS - AN OME APPLICATION

To test its usefulness, we have used the OME to develop the Windows® version of CMSS, called WinCMSS. This development is completed and this section describes this implementation. We also describe the changes that were made to the original architecture, mainly to improve performance.

CMSS predictions are expressed as average annual loads which are calculated by assigning a generation rate per land use class per nutrient. The extent (area or frequency) of these land uses is recorded. Base loads are then the summation of land use extents multiplied by their generation rates.

Scenarios can then be developed which represent changes in land use and/or land management practices. These scenarios modulate the inputs to the model.

WinCMSS has the same functionality as CMSS, with a Windows® interface and all data handling performed by the OME.

3.1 Domain and Model Representation

WinCMSS uses a nested hierarchy to represent a catchment - catchment, catchment regions, sub-catchments and mapping units. The catchment defines the boundary of the application and the WinCMSS application is built at this level. Catchment regions are a convenient method for sub-dividing large catchments into more manageable areas. Each catchment region consists of a group of sub-catchments. Sub-catchments are hydrologically discrete sub-units of the catchment. WinCMSS produces results at these three levels. Mapping Units (MUs) are the finest spatial level. These represent an areal disaggregation of the sub-catchments and land use extents are mapped and stored at this level. Figure 3 is the (inverted) catchment hierarchy.

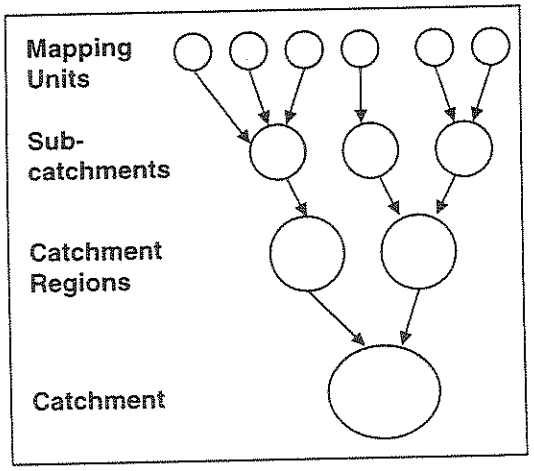

Figure 3, CMSS Catchment Domain Representation

DObjects are assigned at sub-catchment level. Each sub-catchment has its own set of data and associated model(s). A simplified representation of the sub-catchment DObject is shown in Figure 4.

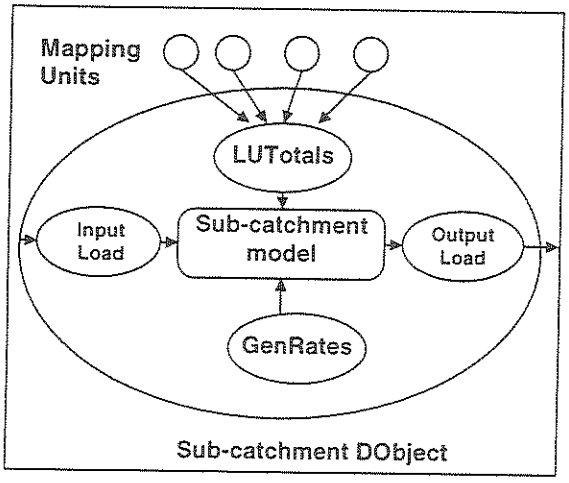

Figure 4, CMSS Sub-catchment DObject

The appeal of this method of object representation lies in the control over domain definition that it gives the user. DObjects represent the level at which the models operate. Relationships with other components (e.g. mapping units) can be represented by data links. Nesting (or scaling) relationships are then captured by the data flow through the links.

The data instance, LUTotals, is a summation of land use extents within the sub-catchment's Mapping Units and is a vector dimensioned by the number of land uses. The formula for calculating LUTotals for a sub-catchment$_{(i)}$ is:

$$\text{LuTotals}_i = \sum_{mu=1}^{mu=n} \text{LuArea}_{mu} \qquad (2)$$

where mu = mapping unit; n = number of mapping units within sub-catchment$_{(i)}$; and LuArea = area of each land use within mapping unit. However, equations are not used. Instead, the relationship is contained within the data flow which is expressed as a 1:m data link relationship between a sub-catchment and its component Mapping Units.

The data instance, GenRates, is the nutrient generation rate for each land use. It is expressed as a weight measure/extent with uncertainty (e.g. 1.5 ± 0.3 kg/hectare/yr. Rates can be uniform across DObjects (e.g. land use x has generation rate y, regardless of location).

The base model is represented as a static, linear model in the form of:

$$\text{OutputLoads} = \text{GenRates} * \text{LUTotals} + \text{InputLoad} \qquad (3)$$

In addition to the standard addition and multiplication of scalar and matrix variables, OME has several matrix operators which were written to improve processing time. These operators are prefixed with 'mat'. One example is the 'matmult' operator (see Figure 6) which produces a matrix with elements that are the multiplication of the corresponding elements of the source matrices.

3.2 Model Connectivity

Load routing between sub-catchments is implemented as DObject-to-DObject links. A very simple routing method is utilised using three variables:

- **From** sub-catchment DObject
- **To** sub-catchment DObject
- **Percentage** of load that is transported between From and To.

Multiple exits from and multiple entries to sub-catchments are catered for by establishing multiple links. For example, a sub-catchment can receive water from two upstream sub-catchments and, in turn, it can have multiple exits (depending on the boundary mapping of the sub-catchment). A visual representation is given in Figure 5.

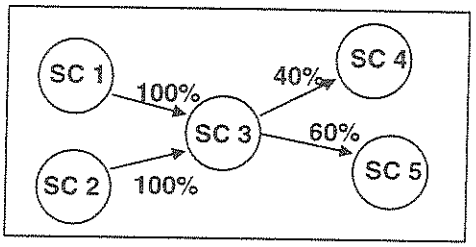

Figure 5, Sub-catchment Routing

To better approximate processes in river systems, a model to simulate instream assimilation is included. Although the chemical and physical processes which lead to instream attenuation of nutrients are complex (and their relationships not well understood), they can be lumped into one 'loss' equation. This is called the assimilation model. The model is attached to sub-catchments and is run on the output load from a sub-catchment DObject before that load is input (via its routing link) to its receiving sub-catchment(s).

3.3 Model Interchangeability

WinCMSS has two models. The **base** model (Figure 6) calculates the base nutrient load generated within a sub-catchment. This is a simple model, though it does carry some complexity in the calculation of uncertainty (a variance function) on the load predictions.

```
// base model
SCdata = invmult ( GenRates, SClocaldata );
SCdata = matmult ( SCdata, Factor );
SCdataout = matadd ( SCdata, SCdatain );
SCerrorout = matadd ( SCerror, SCerrorin );
SClanduseout = SClanduses + SClandusein;
```

Figure 6, OME Code for CMSS Base Model

The second model is the base model plus the assimilation model. This is referred to as the **assimilated base** model (Figure 7).

```
// base model
...
// assimilation component
SCdataout = exp ( -AssimFactor * TravelTime ) *
SCdataout;
SCerrorout = exp ( -AssimFactor * TravelTime ) *
SCerrorout;
```

Figure 7, OME Code for CMSS Assimilated Base Model

3.4 Design Problems

Initially all data in the OME were stored in databases, but when these databases became large, the run-times of WinCMSS expanded dramatically. Real-world applications turned out to be impractical due to poor performance. To overcome this, memory caching of the data was implemented, and performance was improved. Yet now the OME is limited in its capabilities by the amount of system memory available.

Another possible limitation of the design is the amount of data duplication. For instance, the data instance GenRates is used in every subcatchment DObject, yet all the values are the same. This results in extra data storage and slower run-times. The solution used in WinCMSS is to make the GenRates data instance a global data instance, so only one copy is stored. When the subcatchment models are executed, the GenRates data instance is reused each time. This is especially useful when the user changes the GenRates value - only one data instance needs to be updated.

4. THE FUTURE

This section builds on the lessons learnt in the WinCMSS development. It identifies design issues that must be explored and resolved, and proposes some changes to the OME architecture.

4.1 Complex Models

The OME has proved successful for executing, coupling and interchanging simple models with simple (and similar) data. It has yet to be tested with more complex systems, both in terms of complexity of models and, perhaps more importantly, in terms of complexities of data exchange. What is a meaningful and transparent way of coupling models which produce results at different scales, and which describe processes at different scales? The use of links in the OME architecture may be sufficient to encapsulate the data transformation and scaling functions that are required. However, we anticipate that implementation will identify shortcomings in the present design.

4.2 Global Data

The WinCMSS implementation identified the need to define "global" data instances. When the same data is required by many models, it is difficult and memory-expensive to maintain multiple copies of the same data. The original OME architecture (Rizzoli, 1994a) introduced the concept of compound models (an extension of "part-of" relationships in object-oriented programming). This facility was not incorporated in the prototype OME. The WinCMSS solution (of global data instances) is too limited. A better solution might be the use of default values for data instances, where the data template for a group of data instances contains the default value, and data instances can use the default value or create their own value.

4.3 Data Relationships

Data links are a means of capturing relationships between different components of the system being modelled. We see great potential in extending this capacity to explicitly represent data dependencies.

Beyond the core purpose of describing data flow, innovative use of data links in WinCMSS has been minimal, confined to the summation of land use areas from one level in the catchment nested hierarchy to another. Data links could be significantly expanded to handle data validation (e.g. range and consistency checking), data transformation (e.g. change in measurement units, rounding), and inference of data instances from other data.

It is anticipated that data links will reduce the complexity of models by removing much of the (necessary) data processing from the model to the data descriptions.

4.4 Compiler or Interpreter

An important change to the OME architecture will be the replacement of the existing parser. Currently the parser "interprets" the models each time they are run. This is a laborious process for the computer, and the main reason for limited performance in WinCMSS. In contrast, a compiled version of the models will run much faster because the parser can be discarded.

The compiler design is very similar to the interpreter. A parse-tree is built and each node processed, but instead of producing numerical results, a sequence of instructions are produced. These instructions can be executed at run-time much faster than the interpreter can operate. For example, the model expressed in Equation (1) may be represented by:

```
Get Local
Multiply Factor
Add Input
AssignTo Output
```

Figure 8, Equation (1) re-expressed as a Sequence of Single Parameter Instructions

where **Get**, **Multiply**, **Add** and **AssignTo** are functions for manipulating data. By processing this representation even further, executable machine code may be produced, making the run-time even shorter. Speed increases of 10 to 100 times the speed of an interpreter are easily achievable with modern compilers.

4.5 Conclusions

OME development commenced in 1994. Since that time little has changed that would negate the effort thus far put into its development. There have been technological advances in the field of information delivery (e.g. the WWW) and this may prove to be an efficient and effective highway for delivery of EDSSs.

We believe that the ability to handle different models and to test modelling alternatives with minimal effort required by the end-user will be a major requirement for EDSS development in the near future. Modellers today tend to rewrite their own code, often wasting precious time in doing something that has already been done by someone, somewhere. New technologies, such as distributed computing and the Internet can provide the tools either to standardise model interfaces in order to be able to interchange them among different applications (see Abel *et al.*, 1997), and to browse and discover relevant and available information on environmental models (see Guariso *et al.*, 1997).

In this perspective, our work is aimed at studying and understanding the real problems in abstracting models from their interfaces and how domain specific knowledge can be used for this purpose.

5. ACKNOWLEDGEMENTS

The work was initiated by Dr J.R. Davis and funded by CSIRO Division of Water Resources. Further development is being funded by CSIRO Land and Water and the Land and Water Resources Research and Development Corporation (LWRRDC) through the LWRRDC Integration and Adoption Program.

6. REFERENCES

Abel, D.J., K.L. Taylor, and D. Kuo, Integrating Modelling Systems for Environmental Management Information Systems, *ACM* 26(1), 5-10, March 1997.

Cuddy, S., W.J. Young, J.R. Davis, and T.F.N. Farley, Trialing the Catchment Management Support System in the Murrumbidgee Catchment, New South Wales, in *Managing Algal Blooms*, edited by J.R. Davis, CSIRO Land and Water, Canberra, Australia,103-113, 1997.

Davis, J.R., and T.F.N. Farley, CMSS: Policy Analysis Software for Catchment Managers, *J Env Modelling and Software*, Elsevier Applied Science, 1997 (in press).

Fishwick, P.A., *Simulation, Model Design and Execution, Building Digital Worlds*, Prentice-Hall, London, 1995.

Guariso, G., and H Werthner, *Environmental Decision Support Systems*, Ellis Horwood Limited, Chichester, U.K., 1989.

Guariso, G., E. Tracanella, L. Piroddi, and A.E. Rizzoli, A web accessible environmental model base: a tool for natural resources management, In these proceedings, 1997.

Minsky, M., A framework for representing knowledge, in *The Psychology of Computer Vision*, edited by P.H. Winston, McGraw-Hill, New York, 1975.

Mortice Kern Systems Inc., MKS Lex & Yacc™, http://www.mks.com/solution/ly/, 1997.

Rizzoli, A., A Software Architecture for Model Management and Integration: Theoretical Background, Technical Memorandum 94/10, CSIRO Division of Water Resources, Canberra, 1994a.

Rizzoli, A., A Software Architecture for Model Management and Integration: An Application to Catchment Runoff Modelling, Technical Memorandum 94/9, CSIRO Division of Water Resources, Canberra, 1994b.

Zeigler, B.P., G. Klir, M. Elzas, and T.I. Oren, *Methodology in Systems Modelling and Simulation*, North Holland, Amsterdam, 1979.