Towards a Software Architecture to Facilitate Multiple Runs of Simulation Models

Perraud, J.-M. [1, 3], G. Kuczera [2, 3] and R.J. Bridgart [1,3]

[1] Commonwealth Scientific and Industrial Research Organisation, Land and Water, Australian Capital Territory
[2] School of Engineering, University of Newcastle, New South Wales
[3] eWater Cooperative Research Centre
Email: jean-michel.perraud@csiro.au

Keywords: *Monte-Carlo simulations, uncertainty analysis, modelling engine*

EXTENDED ABSTRACT

A variety of techniques in environmental modelling require multiple runs of simulation models. These include for instance uncertainty and sensitivity analysis, using stochastically generated replicate climate inputs, calibration and optimisation, and straight batch running of multiple scenarios. These techniques are in theory applicable to the vast majority of spatial and temporal simulation models in use, and fairly generic tools exist that implement them. Yet the overhead associated with applying these tools to the simulation model at hand is often still significant in practice, with limited prospect for economy of scale.

This paper illustrates the architectural considerations to have in order to facilitate these techniques from the ground up in modelling frameworks, while allowing the use of existing analysis tools. One of the aforementioned techniques, uncertainty analysis, is presented and is used as a driver to determine an object oriented architecture for modelling engines, suitable to underpin uncertainty analysis.

First, a general conceptual framework for uncertainty estimation of simulation models driven by point time series is presented. This framework does not require knowing the inner workings of the model, and allows for a generalised description of outputs as a random vector with a probability distribution. To illustrate the importance of accounting for uncertainty in decision support, this framework is applied to a simple example of an urban water supply with one reservoir, embodied in a temporal model running at an annual time scale. It demonstrates that the reliability of the water supply is sensitive to the value of the mean annual streamflow, and that the uncertainty of this parameter must be accounted for in the modelling process to avoid an excessively optimistic reliability assessment.

The second part of the paper takes this general uncertainty framework and the urban water supply example as a use case. They are analysed, and a summarized design process is presented. The aim of the process is to derive, from the mathematical description of the conceptual framework, an object oriented software architecture for organising simulation models and modelling engines, such that they are more amenable to being subject to uncertainty analysis. To foster a layered design, the design process purposely makes no assumption about the details of how model runs are distributed. This paper concludes with an outline of the computing aspects and the current thinking about how multiple model runs can be parallelised and distributed with different distributed computing tools.

1. INTRODUCTION

Mathematical simulation models play a pivotal role in decision support for water resources planning and operation. These models need to simulate temporal and spatial processes that are complex, often insufficiently understood and subject to stochastic climate forcing. A defining feature of these models is uncertainty. Typically complexity is accommodated by using conceptual models that seek to simulate the dominant dynamics of the system. As a result, such models have parameters that require calibration and produce predictions that are subject to considerable uncertainty. Because future climate ranging from short to long time scales cannot be predicted with confidence, it is necessary to use long historic records or preferably long records generated by stochastic models to evaluate the behaviour of the water resource system. These uncertainties complicate matters because metrics of system performance will themselves be random variables. Moreover, these random variables may change over time as a result of underlying changes in climate forcing and demand and through interventions by managers.

Modelling tools and frameworks must handle the uncertainties that are inherent in water resource planning and operation, to provide credible decision support. Fundamental to this challenge is the capability to supervise multiple runs where a run is defined as a simulation conducted over some interval of time. This capability is needed in calibration, evaluation of system performance, sensitivity analysis and in scenario evaluation.

This paper reports on the current state of the thinking in activities undertaken in the eWater Cooperative Research Centre (www.ewatercrc.com.au), primarily to facilitate the development of modelling systems able to support or facilitate uncertainty analysis from the ground up. It first describes a general conceptual framework for evaluating system performance, the forward problem, and calibration, the inverse problem. A simple case study of a water supply at an annual time scale is presented. Then, this framework is analysed in order to derive suitable software architecture. While this analysis is applied to uncertainty analysis in this paper, it is hoped that a similar process applied to other modelling techniques requiring multiple model runs would yield similar software architectures, thus allowing for a common approach to better handle operationally multiple model runs.

2. A GENERAL UNCERTAINTY FRAMEWORK BASED ON MULTIPLE RUNS

The starting point for a framework capable of handling the various forms of uncertainty inherent in water resource planning and operation is a careful, robust definition of the system and its attendant uncertainties. In general, the properties of a simulation model can be categorised as:

- *Forcing (inputs)*: Variables that drive or force the system to respond and may change at every time step.
- *Parameters*. Variables that characterize processes within a model and that typically do not vary during a run of the system. However, if the parameters are subject to uncertainty, they may vary between runs.
- *States*: Variables of the model that are set by the model during a time step. Some may affect the model at a given simulation time by their antecedent conditions in previous time steps, and may thus need to be set at the start of a simulation run, sharing some characteristics of an input variable.
- *Outputs*. The distinction between a state variable and an output is somewhat arbitrary: outputs are the state variables of prime interest in a given modelling context.

In this paper we limit the scope of our analysis to simulation models that simulate time series of one or more output variables at discrete times and discrete spatial locations. This class of model is very general and is likely to encompass most of the eWater model applications, and *a priori* the future extension to other types of simulation models is not excluded.

Such models can be reduced to the canonical form

$$z_t = f(x_t) \quad (1)$$

where f() is the vector-valued function which represents the simulation model that maps the input p-vector x_t into the q-output vector z_t at time t.

It is important to note that there is no need to know the inner workings of the simulation model – all that is required is the capability to receive the output z_t on submission of the input x_t. The key insight stems from the following observation: If one or more elements of the input vector x_t are random variables, then the output time series z_t

will be a random vector with a probability distribution.

The object of the exercise is to infer the probability distribution of z_t. The input x_t consists of forcing, state and parameter values which have meaning within the context of the simulation model. However, from the perspective of uncertainty propagation, this distinction is unimportant. The only requirement is that x_t be partitioned into subvectors or groups where all the elements within the group are sampled at the same time. More formally, partition x_t into m groups or subvectors $\{x_{jt}, j=1,..,m\}$.

During the simulation run, sampling of the variables within the groups is conducted at defined times defined by the n-vector of sampling times $t = \{t_i, i=1,..,n\}$ where t_i is the actual time a random sample takes place. To retain maximum flexibility it is important to allow groups to be sampled at different times. This can be accomplished by defining a group sampling indicator function

$$I_s(j,i) = \begin{cases} 1 \text{ if group j variables are sampled at } t_i \\ 0 \text{ otherwise} \end{cases} \quad (2)$$

Once it is known that group j is to be sampled at time t_i, the vector x_{jt_i} is randomly sampled from the probability model associated with the j^{th} group; that is,

$$x_{jt_i} \leftarrow p_j(x \mid x_{t_1},..,x_{t_{i-1}}, x_{1t_i},..,x_{j-1,t_i}) \quad (3)$$

The random variable x_{jt_i} may be conditionally dependent on input vectors sampled at previous times $x_{t_1},..,x_{t_{i-1}}$ and on inputs sampled in earlier groups at the current time, $x_{1t_i},..,x_{j-1,t_i}$. This conditional dependence offers a rich range of possibilities.

2.1. Forward Problem

In the forward problem, the group probability distributions are known. The objective is to derive the probability distribution of z_t associated with the system (1). The Monte Carlo method offers an intuitive and general approach for solving this problem. It involves two steps:

1. Random sampling from the probability models for each group at each time step;
2. Replication of the time series a sufficient number of times to enable meaningful estimation of the probability distributions of the outputs z_t, $t = \{t_i, i=1,..,n\}$.

It is noted that Monte Carlo simulation only yields results of acceptable accuracy when an adequate number of runs are sampled. This is because accuracy, as measured by the standard error, is inversely proportional to the square root of the number of runs. This is the Achilles heel of the Monte Carlo method and needs careful handling to ensure acceptable computational times.

2.2. Inverse Problem or Calibration

In the forward problem, multiple runs are fundamental to the Monte Carlo method. The same applies to the inverse problem which involves calibrating the model.

In calibration the inputs and outputs are observed (usually with error), while the parameters are unknown and need to be inferred. The objective of the calibration is to identify parameter values, or more generally distributions, which are most consistent with the observed data.

There are several general approaches to accomplishing this. One approach seeks to optimize an objective function such as in least squares estimation. This involves conducting a search involving many trial parameters. For each trial parameter set, the model is run using observed inputs and the simulated outputs are matched against observed outputs. In the Bayesian approach the posterior distribution of the parameters is reconstructed typically using Markov Chain Monte Carlo methods [see Gelman et al., 1995]. This involves many runs with different parameters.

2.3. An example: accounting for uncertainty in performance evaluation

To illustrate the importance of full accounting of uncertainty in decision support, we consider a simple example involving drought security assessment for an urban water supply system. The system, illustrated in Figure 2, consists of a single reservoir with capacity 1000 units receiving an annual streamflow which is uncertain and is described by a normal distribution with mean m_q and standard deviation 200. We shall use the shorthand notation $N(\mu, \sigma^2)$ to denote a normal distribution with mean μ and standard deviation σ. The reservoir supplies an urban zone with known demand of 500 units. The planner wishes to assess the probability of the reservoir running out of water in any year – this will be denoted by P(empty).

There are two sources of uncertainty: Future annual streamflow cannot be predicted using current forecasting techniques and, as a result, its variability is best described by a probability model. The second source arises from uncertainty in the mean of the annual streamflow, m_q. Due to short historic flow records, the true value of m_q is not known. From a statistical analysis of the data, suppose the uncertainty about m_q can be summarized by the normal distribution $N(\mu_m, \sigma_m^2)$. As a result of this uncertainty in m_q, there is no single value of P(empty). Rather P(empty) itself is a random variable with a mean and standard deviation.

The framework described in Section 2 is used to evaluate the distribution of P(empty). With reference to Section 2, the input vector x_t has two groups:

- The group 1 random variable m_q is sampled at the start of the simulation run according to

  $$m_q \leftarrow N(\mu_m, \sigma_m^2) \quad (4)$$

- The group 2 random variable, the annual streamflow q, is sampled each year t according to

  $$q_t \leftarrow N(m_q, 200^2) \quad (5)$$

  Note that q_t is conditionally dependent on the mean streamflow m_q which is sampled at the start of the run.

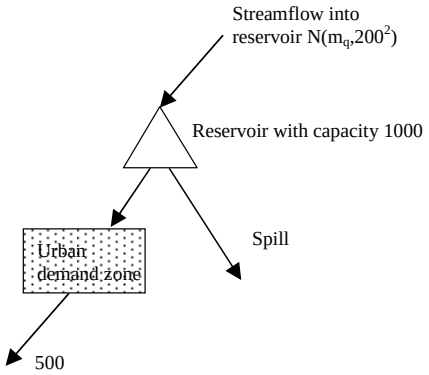
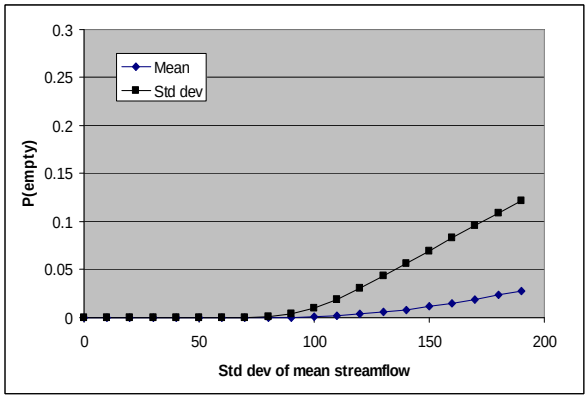


Figure 1 Schematic of reservoir system

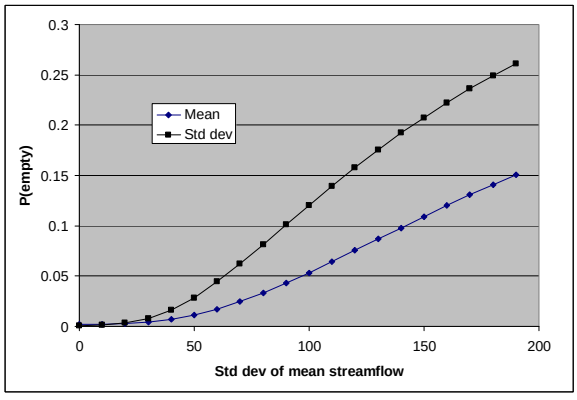
Suppose m_q is 800. In Figure 3a the mean and standard deviation of P(empty) are plotted as a function of σ_m. One observes that both the mean and standard deviation of P(empty) are less than 0.001 until σ_m exceeds 100.

In stark contrast, Figure 3b shows that when m_q equals 600, the mean and standard deviation of P(empty) grow alarmingly with σ_m. Even when σ_m equals 50, the probability of zero water supply to the urban area is of the order of a few percent and would render the system unacceptable and in need of urgent attention.

If the uncertainty in the parameter m_q had been ignored, an excessively optimistic, indeed dangerous, assessment of drought security would have resulted. This highlights the importance of a modelling framework that naturally supports the characterization and propagation of uncertainty.



(a) Expected value for m_q= 800.



(b) Expected value for m_q= 600.

Figure 2. Mean and standard deviation of P(empty) as a function of the standard deviation of m_q.

3. A SOFTWARE PATTERN FOR THE SUPERVISION OF MULTIPLE RUNS

3.1. Design goals

The paper is part of an effort to design an object oriented software architecture that can support techniques such as uncertainty analysis, sensitivity analysis and model calibration. The main purpose of this architecture is to suggest software patterns to organise modelling engines such that they are more amenable to batch running in a broad sense. As a first step this present section mostly analyses the water supply case study and the generalised uncertainty framework conceptual previously described. It then proposes a software architecture, largely focussing on identifying entities in a single simulation run such that multiple runs and alternate model inputs and outputs are facilitated.

Other goals are:

- The pattern proposed should be decoupled from the specific means of distributed computing. As such the question of whether the runs occur on a single machine, a cluster or a GRID infrastructure is left out of the scope of the present section.
- The architectural elements should allow for the use of external tools and libraries to instrument the model analysed, for instance the Data Uncertainty Engine (Brown *et al.* 2006) or PEST (Doherty 2002). The eWater CRC also plans to produce a set of probability models suited to the water management domain.
- The architectural elements should not be coupled to a particular modelling framework, platform or language. However it is acknowledged that previous work based on TIME (Rahman 2003) likely has an implicit influence on the present design process, as a proof of concept implementation is currently based on this modelling environment.

3.2. Uncertainty analysis as a use case

Based on the presentation of a general uncertainty framework in the previous section, the following observations are made:

1. From a software viewpoint, variables that are inputs or parameters are properties that can be set. Conversely outputs and most state variables are properties that are usually read. State variables whose antecedent conditions matter are an exception, since they can be considered as an input or an output, but need a differentiated treatment for the purpose of setting or getting values only when considered at different event times in the system: start of run, before a time step, after a time step.
2. There is no need to know the inner workings of the simulation model, at least not explicitly. The model needs not know where its inputs are coming from (resp. where its outputs are feeding into). In particular temporal simulation models need not know upfront in a simulation run about the whole input time series. This is actually paramount to allowing for conditional probability density functions (PDFs) feeding into the model, e.g. an input value conditional on a state value of the model at the previous time step(s).
3. Each simulation run is a priori independent of others. If there were a dependency it would be indirect and implicit, by way of configuring the PDFs attached to the inputs in subsequent runs.

The process in the uncertainty analysis of the water supply case study can be represented via a simplified sequence diagram (Figure 3). The application, here referred to as the "uncertainty analyser", delegates the handling of running multiple simulations to a multi-run supervisor. The application needs to pass only the high-level properties of the analysis to the supervisor, e.g. the uncertainty of the mean and standard deviations of the input PDFs for the annual reservoir inflow, while the supervisor initialises those PDFs for each simulation run. This supervisor may run replicates serially, in parallel, or a mix thereof, based on conditional dependencies between runs and the availability of multiple processors. The water supply example does not introduce dependencies between runs, but the general uncertainty framework and inverse problems can do so. Those dependencies can be difficult to deal with, but mostly from a software implementation viewpoint; all analyses with dependencies between runs conceptually reduce to a non-cyclic graph.

There are three entities in charge of each simulation run: the simulation model itself (the core algorithms for a time step), a model runner (in charge of stepping the simulation through time), and a "PropertiesHandler" responsible for getting and setting the properties of the model. Note that figure 3 focuses on identifying the generic events and associated actions for each simulation run. Clearly identifying this PropertiesHandler as a separate entity is central to facilitating multiple simulation runs. It allows for transparently using alternate sources of (resp. destination for) data, a need highlighted by observation (2) above.

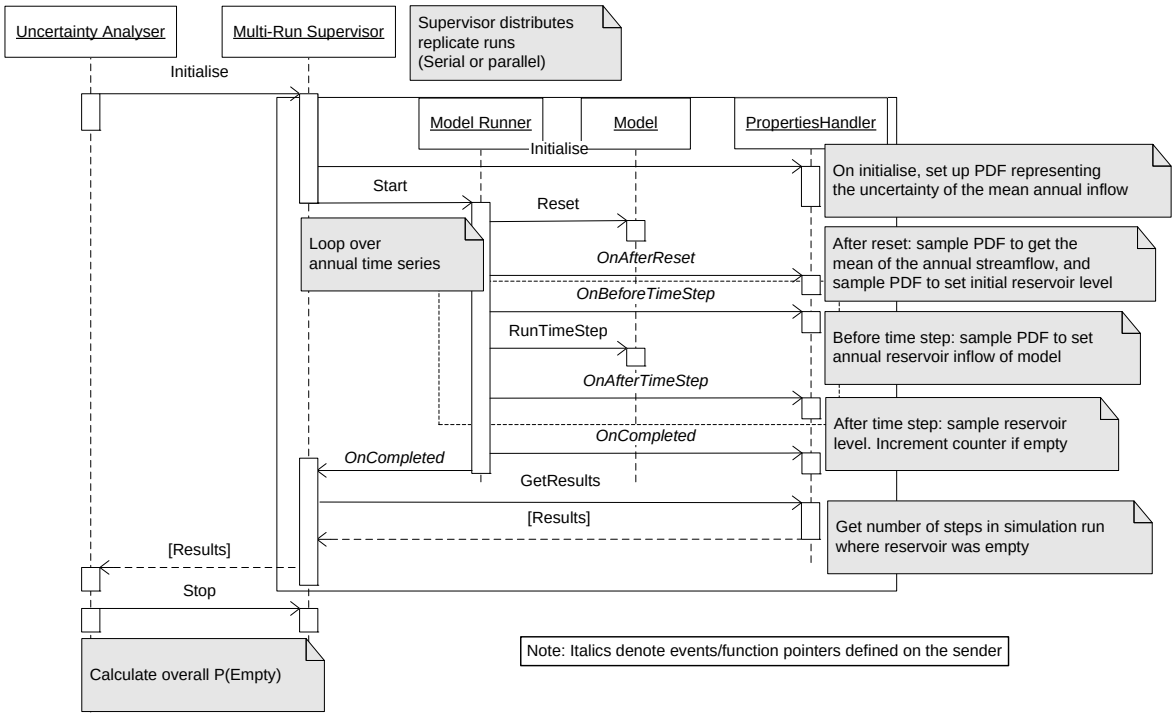


Figure 3: Simplified sequence diagram for the uncertainty analysis of the water supply system.

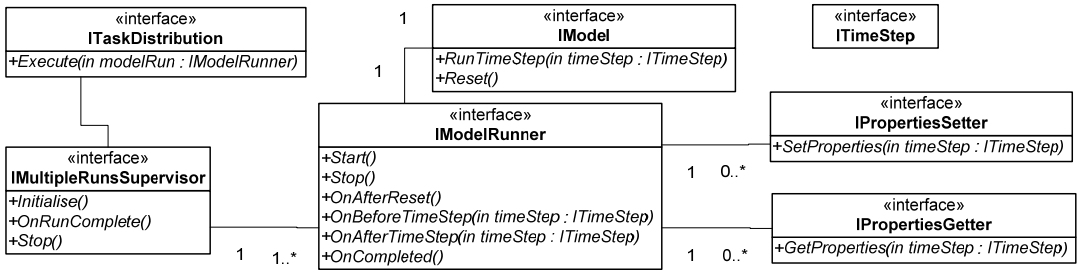


Figure 4: Simplified static structure for a multi-run architecture

Using a pattern with these three entities for the core modelling engine, one can easily scale it up, typically from a single simulation run on a set of observed climatic input time series, to using one or more PropertiesHandler with alternate back-end systems for setting or getting the model properties, e.g. random number generators from an external probability model or sets of stochastically generated time series.

3.3. An architectural pattern to facilitate multiple model runs

From the sequence diagram in Figure 3, a simple static structure can be derived as presented in Figure 4. This is an abstract static structure, in the sense that it is a generalisation inferred from the water supply use case dealt with so far.

The part of the pattern that deals with one simulation run comprises IModelRunner, IModel, IPropertyGetter and IPropertySetter, the last two composing "PropertiesHandler", setting or getting the model variables at appropriate times in the simulation. Many recent modelling engines could be abstracted as more or less following this pattern. One example can be found in (Perraud *et al.* 2005) to "play" (resp. "record") time series to (resp. from) a catchment modelling engine, using software reflection to decouple the model from the time series. OpenMI (Gisberg *et al.* 2005) could also conceivably be used to handle the exchanges of information between IModel, IPropertyGetter and IPropertySetter, although the "pull" paradigm in OpenMI may impose specific arrangements to the sequence of those exchanges.

While the abstract structure is very sparse, we infer that a wide array of techniques requiring multiple simulation runs can be cast to follow this pattern. A key aspect is the high level of abstraction of IPropertyGetter and IPropertySetter. The pattern makes no assumptions whatsoever as to the nature

of their back-end data (time series, random number generators, threshold recorders), nor the type of the model properties dealt with (parameters, inputs, etc.)

Though it can not be discussed in details in this paper, the entity dealing with the effective distribution of simulation runs (ITaskDistribution) is essential. How much flexibility it can achieve an important indicator of how operational the proposed system architecture can be. The following section gives initial elements of reflection on this topic, currently explored.

4. COMPUTING ASPECTS

Monte-Carlo based techniques, such as the one described above, are inherently computationally intensive. To handle this, strategies focus on the parallelisation of work and the utilisation of computer hardware beyond the single CPU. Specific implementations can vary in scope and complexity, ranging from multithreading, designed to take advantage of the new range of multi-core processors, to parallel architectures designed to utilise computational grids and clusters. The types of problems explored so far for use using the multi-run architecture are largely readily parallelisable. This is advantageous, in that minimal design changes tend to need to be made in order to harness the potentially significant gains afforded when using a computational grid. The example above pertaining to uncertainty evaluation is a prime example where multiple independent tasks could be derived and the workload shared across multiple machines.

Conceptually, many distribution frameworks operate on very similar principles. Different frameworks may operate on differing levels of parallelisation, ranging from coarse-grain systems that define tasks at the application level, to fine-grain systems that allow for tasks definition at the code level. Both approaches have their strengths and weaknesses but share a common need for the computational task to be expressed appropriately. A key task in this endeavour is to find a way to translate each replicate simulation run, as described in previous sections, in a form suitable for alternate distribution frameworks.

5. CONCLUSION

This paper presents a general uncertainty framework for mathematical simulation models, which entails running a large number of simulation runs. A case study of a water supply system illustrates the critical importance of taking uncertainty into account in decision support. This uncertainty framework and the case study are then analysed to determine the key desirable characteristics of an object-oriented architecture for the modelling engine, such that this modelling engine is more amenable to be run repeatedly on replicate input data sets. This paper is a first step in a process aiming to determine a consistent software approach to support a range of model analysis techniques requiring multiple model runs. It is hoped that a common conceptualisation will help decoupling the problem formulation from the distribution of computing task. This way different distributed computing platforms can be used for a given analysis technique requiring multiple simulation runs.

6. REFERENCES

Brown, J. D. and G. B. M. Heuvelink (2006) The Data Uncertainty Engine (DUE): A software tool for assessing and simulating uncertain environmental variables, Computers and Geosciences, (33), 172-190

Doherty, J. (2002), PEST, Model-independent parameter estimation, fourth edition (2002), User manual, Watermark Numerical Computing, pp. 279

Gelman, A., Carlin, J.B., Stern, H.S. and Rubin, D.B.(1995). Bayesian data analysis, Chapman and Hall.

Gijsbers, P.J.A. and J.B..Gregersen (2005), OpenMI A glue for model integration. In Zerger, A. and Argent, R.M. (eds) MODSIM 2005 International Congress on Modelling and Simulation. Modelling and Simulation Society of Australia and New Zealand, December 2005, pp. 648-654. ISBN: 0-9758400-2-9

Perraud, J.-M. , S. P. Seaton, J. M. Rahman, G. P. Davis, R. M. Argent and G. D. Podger (2005) The architecture of the E2 catchment modelling framework. In Zerger, A. and Argent, R.M. (eds) MODSIM 2005 International Congress on Modelling and Simulation. Modelling and Simulation Society of Australia and New Zealand, December 2005, pp. 690-696. ISBN: 0-9758400-2-9

Rahman, J.M., S.P. Seaton, J-M. Perraud, H. Hotham, D.I. Verrelli and J.R. Coleman, (2003), It's TIME for a New Environmental Modelling Framework, MODSIM 2003, pp. 1727-1732