A New Rainfall Runoff Software Library

J.-M. Perraud[a], G. M. Podger[b], J. M. Rahman[a], R. A. Vertessy[a]

[a]Cooperative Research Centre for Catchment Hydrology, CSIRO Land and Water, Canberra, jean-michel.perraud@csiro.au

[b]Department of Land and Water Conservation, Sydney

Abstract: Lumped-conceptual rainfall-runoff models are a 'stock in trade' tool for those working in the water industry and allied fields. However, there are many alternative models available, and sometimes multiple implementations of the same model. The consequence of this is a plethora of models with different operational features and different access arrangements. Similarly, there are a variety of optimisation tools being used to calibrate such models, and they too vary greatly in terms of their style and access arrangements. To address this issue we have developed a software application currently called the 'rainfall-runoff library' (RRL) and have made this available to end-users via the World Wide Web. The RRL includes some popular lumped conceptual rainfall-runoff models, including AWBM, SimHyd and Sacramento. It also includes popular model-independent and multi-objective parameter optimisation routines, including the Pattern Search, Shuffled Complex Evolution and Genetic Algorithm methods. In addition, a custom parameter optimisation method for AWBM is provided. The RRL has powerful visualisation and data handling features. It is expected that end-users will be more willing to trial a range of models and parameter estimation methods on their particular modelling problem as the 'cost' of moving from one method to another is relatively minor given the common interface elements. The tools embedded within the current version of the RRL are a sample only and we expect the software to grow significantly over the coming years. For this reason, we have designed the RRL as an extensible system that simplifies the task of adding models or optimisation methods, by using object introspection and metadata. This paper explores the functional analysis, technical design and implementation of the RRL.

Keywords: *Rainfall-runoff models; Calibration; Metadata*

1. INTRODUCTION

Modelling of rainfall-runoff relationships has been studied for several decades. One consequence is the proliferation of a large number of models implemented with different access arrangements, software requirements, and data formats. Despite this proliferation, or maybe because of it, most engineers will use primarily one model for any given task. However, papers like those of Perrin (2001) suggest that while some models may be identified as more robust, the variability in the behaviour of catchments ensures that no model will be the best in all cases.

The Cooperative Research Centre for Catchment Hydrology (CRCCH) has built a library of lumped conceptual rainfall-runoff models for inclusion in its Catchment Modelling Toolkit. The aim is to unify the tools commonly found in rainfall-runoff model software implementations, while still allowing for model specificities. The library is implemented within the CRCCH modelling framework TIME (The Invisible Modelling Environment), and adopts many of its software engineering paradigms.

One purpose of this library is to offer the water engineering community a set of models that is accessible through a common, user-friendly interface. While this library is not meant to be a tool for model development, it is structured such that the effort required to add a new model or optimisation method is minimal. Another goal of the project is to develop hydrologic software components which can be re-used within other applications resident in the CRCCH Catchment Modelling Toolkit. This paper presents the rationale and the design of the Rainfall Runoff Library (RRL), and how needs driven by this project can lead to the production of generic re-usable components in an integrated modelling environment.

2. ANALYSIS OF NEEDS

The main population of users of the RRL are found in the water engineering community. A large number of model structures were reviewed

to determine the common patterns found in them. As part of the analysis of likely needs of this group, a previous implementation of AWBM from Boughton et al. (1996) was carefully examined, as this is one of the most widely-used models in the Australian water engineering community. Interestingly enough there were very few properties found to be truly common to all models, aside from the rainfall input data, which indicated the need for a framework which could accommodate idiosyncratic model structures.

The water engineering community's major needs in terms of an integrated RRL were identified as follows:

- The ability to read datasets from existing software implementations and environments,
- A unified software interface containing generic visualisation features found in the majority of model implementations , as well as more advanced visualisation capabilities,
- The ability to easily trial and compare several models to avoid the one-model-fits-all syndrome,
- A facility to add new rainfall-runoff models with a minimum of development effort,
- The ability to track and archive the tasks performed leading to the calibration of a model on a particular catchment and the results.

Although most of lumped rainfall-runoff models use a daily time step, our implementation of these in TIME does not make any assumption about the time step.

Most required tools, with the help of object-oriented programming, are rather naturally leading to the design of re-usable hydrologic software components. Examples of such components might include a Unit Hydrograph tool, elementary "bucket" objects, or the discrete probability density function generators used in several calibration tools based on evolutionary principles. In other cases the specific needs, provided they are looked at within a broader context, lead to the definition of more generic and powerful mechanisms. Such an example might be the code needed for axis transformations required for the display of duration curves on a log-normal scale.

The aspect of the RRL that required the most development effort was the model calibration tools. The optimisation methods we implemented were categorised in three broad types: generic, custom and manual. The generic optimisers treat equally all parameters of a model, whereas custom ones have an intrinsic knowledge of a given model and its parameters. Tools based on well-known procedures have been included, like the Pattern Search from Hookes et al. (1961), the method described by Rosenbrock (1960), Genetic Algorithm (Wang, 1991), and the Shuffled Complex Evolution, based on Duan et al. (1992) and (1994). Although optimisation tasks in the RRL are focussed on runoff time series, we have implemented these various optimisation methods such that they can be applied to other data types such as rasters. In this way, they serve a more general purpose in the Catchment Modelling Toolkit.

The quality of the calibration of a model for a given purpose depends a lot on the choice of an appropriate objective function. When surveyed, end-users declared a clear need to choose from a range of objective functions in the calibration process. They also stated the need to be able to calibrate on the flow duration curve as well as flow series. One of the reasons given for this was to overcome time mis-matches between input rainfall and flow data. One such mis-match problem arises when daily rainfall records start and end at 9am whereas daily flow records start and end at 12 am.

3. DESIGN

3.1. System Architecture

The overall architecture of RRL follows principles that are fairly standard in modern software engineering practice. The application is broken into four main parts (Figure 1). The data Input Manager allows for the transparent addition of new readers and writers for new time series file formats. The Model Manager references all of the rainfall-runoff models implemented in the RRL and includes a collection of tools necessary for model introspection (i.e. the discovery of model variables relying on metadata; see next section). The Calibration Manager acts as the coordinator of all the objects involved in an optimisation process, and as a single point of entry in this regard for the code in the user interface layer. Results and graphs are presented via the Display Manager. These layers are discussed in more detail in the following sections.

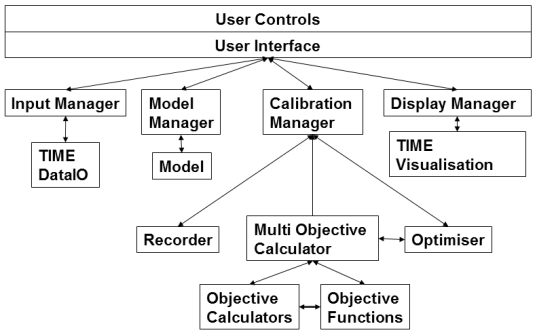

Figure 1 System architecture

3.2. Metadata attributes

The use of metadata attributes, as described by Rahman (2003), is a prominent aspect of the software design for the RRL and permits the development of powerful and flexible model processing tools and user controls. Metadata attributes carrying the characteristics of model parameters, such as Minimum, Maximum, and Default Value, is of particular interest in the RRL. For instance, the quality of model calibration is very dependant on the specification of the parameter space boundaries, hence the importance of the correct definition of these attribute. Most of the flexibility in the RRL is achieved through attributes and their detection at run-time. Examples of the use of attributes to address specific needs associated with the analysis of the RRL are discussed in the next section.

3.3. Objects and Components

All rainfall runoff models in the RRL inherit from the same parent abstract class 'RainfallRunoffModel'. This class plays a key unifying role to detect the available models at run-time. As indicated in the section of needs analysis, it appeared that the vast majority of properties and methods of the models could not be deferred to a parent class. Hence, it was decided that this class had to be kept very small in order to leave room for the idiosyncrasies of each model, such as the definition of its inputs, outputs and parameters.

Following the design philosophy of TIME, the individual models in the RRL do not explicitly handle time. Instead, they utilise a simple *runTimeStep()* method and the declaration of inputs, outputs, and parameters as doubles or integers. Keeping the models small is one important reason which permits the easy addition of a new model to the RRL, since no complex time handling is necessary.

The Input Manager is a context-sensitive component, which relies solely on the discovery of model properties tagged with the Input attribute. It also stores the definition of *input modifiers* (e.g. monthly pan factors), which is a generalisation of a feature found in the Sacramento model implemented by the New South Wales Department of Land and Water Resource Conservation. Input data is thus left in its original state while the correcting factors applied are explicit.

The Calibration Manager has the role of storing information on the current state of calibration, which includes primarily the parameter values, the dates for calibration and verification, the type of optimiser, and the selected objective functions. It is thus hiding the complexity of the calibration process from the user interface layer. Reflecting the categorisation of the optimisers, two interfaces have been defined: IGenericOptimiser and ICustomOptimiser. As all automatic optimisation schemes rely in one way or another on the maximisation or minimisation of one or several objective functions, it is clearly necessary to standardise the behaviour of the optimisers in this respect. If not, it easily leads to a bug-prone situation. The convention adopted is to separate the task of optimising from the task of calculating the objective value. An IGenericOptimiser will always minimise the objective, while an Objective Calculator, which will be described in further details below, is in charge of returning a value which is to be minimised, relying on the inspection of the *Minimise* attribute on the method calculating the objective function. Another characteristic of many optimisation schemes is the parallel or successive calculations performed by objects of the same type. Consider for instance the case of a Pattern Search with multiple start searches, or the evolution of some ensembles or populations of points in a parameter space (Genetic Algorithms). Many properties will be transversal to all the objects of the same type, for instance the number of points per population and the maximum number of iterations. Also, optimisers of all types share a few properties and actions identical in purpose (e.g. notifying the user interface of the status of the ongoing optimisation). Due to these common properties an optimiser with consistent mechanisms has been structured into the framework of the RRL, where the optimisers (e.g. a collection of Pattern Search) and the individual search instances share the same parent class, although only the former will implement the interface IGenericOptimiser.

The Objective Calculator relieves the optimisers of the task of calculating the goodness of fit, while offering additional flexibility, such as the

ability to calibrate on a flow duration curve as opposed to a flow series. A generic mechanism has been designed, whereby the observed and calculated runoff time series are, from the point of view of the optimisers, transparently transformed to the appropriate form prior to the calculation. A Compound Objective Calculator class, with a customisable property 'compounding function', inherits from the Objective Calculator. A calibration on several objectives is thus again hidden from the optimiser. Any number of objectives can be set up, although the RRL itself currently uses a maximum of two.

Output time series are handled through the use of a Recorder object. The Recorder uses internal dictionary structures or 'Hashtables', as referred to in .NET terminology, where keys are the fields of the models, and the values the references to the time series in which to store the output. Combined with the run-time discovery of properties tagged with the Output and State attributes, this offers the option to record several model properties of interest.

The visualisation components have two important features worth mentioning here. First, an abstract class AxisTransform has been implemented, so that advanced visualisation capabilities required by the RRL are addressed. The graphical canvas has properties that can be set to any suitable transform (e.g. LogTransform and NormalTransform, Figure 2). Second, the use in TIME of the Observer-Subject design pattern permits dynamic updating of all graphs when model parameters are changed.

3.4. Languages

The RRL is implemented in TIME, which in turn is founded on the Microsoft .NET platform, which offers multi-language capabilities. Most of the RRL is written in C#, but Sacramento is written in Fortran for .NET. It is also expected that some more models will be implemented in Visual Basic. Other possible coding languages include J#, Eiffel, and Pascal, to cite a few. Hence, one of the real virtues of the RRL is that collaborators can contribute new models to it using the language of their choice.

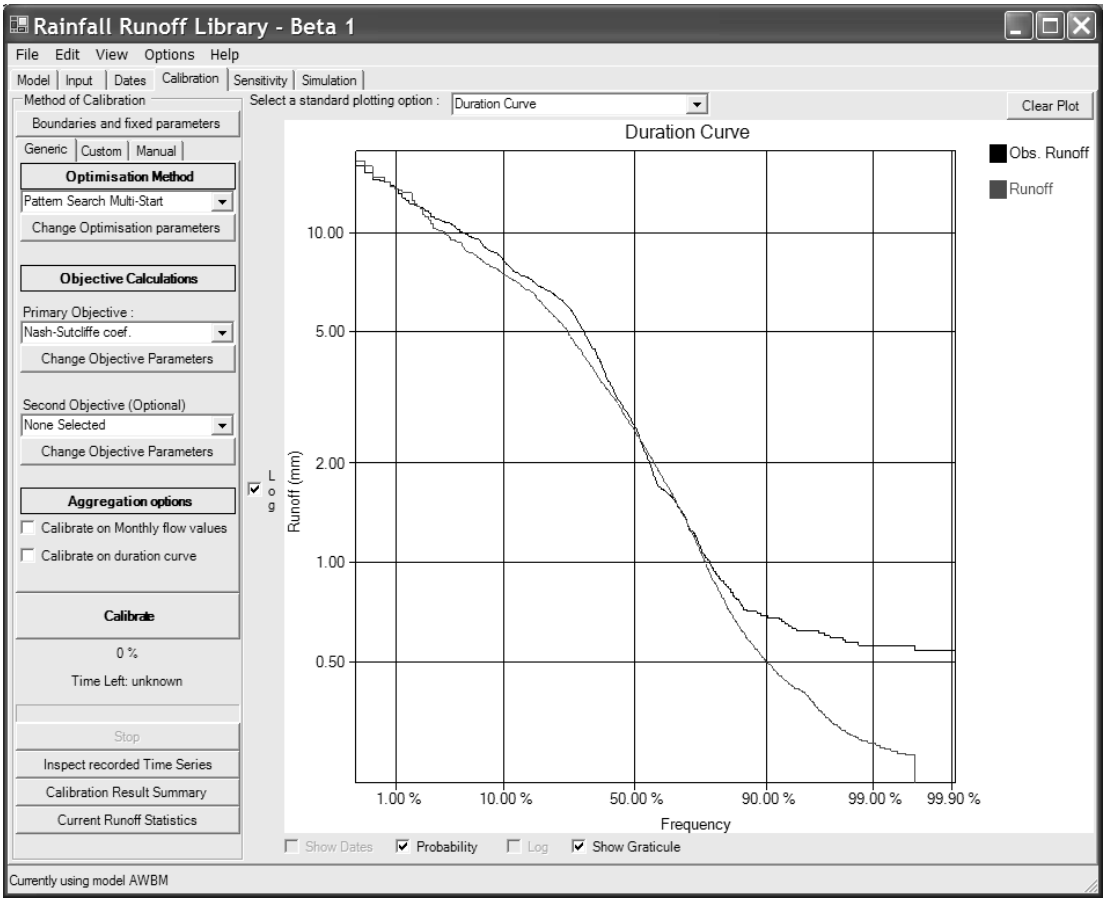

Figure 2. Duration curves of observed and calculated runoff, plotted on a log-normal scale.

4. CHALLENGES

4.1. Models with varying structure

Some rainfall-models are based on a varying structure, and hence contain varying numbers of parameters (for instance, the IHACRES model, Jakeman *et al.* (1993)). This is a clear challenge to the application of generic optimisation methods, since this means that the parameter space may change from one iteration to another in the calibration process. This can be also the case when models are using unit hydrographs and the number of parameters of the model is varying with the length of the memory of this unit hydrograph. While optimisation tools can be designed to handle varying structures and parameters, they are very difficult to implement, and the behaviour of the generic algorithms is unknown.

4.2. Optimisation under constraint

It is not rare to find in a model some of its parameters related by an equation or inequality. For instance, AWBM can have three parameters A_1, A_2 and A_3 (partial surfaces) related by the equations

$$A_1 + A_2 + A_3 = 1 \quad (1)$$

with $\forall i, A_i \geq 0$ (2)

The problem of varying freely the first two parameters in a truly generic optimiser while keeping this constraint *strictly* true is not trivial. There are many good analytical methods for optimisation subject to constraints (see for instance Arsham (2003) and references). The implementation of one of the generic numerical solution algorithms would here again require a significant amount of work, due mainly to the non-linearity of the response curve.

A more pragmatic approach to calibrating models with parameters subject to constraints and also dealing with models with a variable structure is probably to design custom optimisers with intrinsic knowledge of those models, and which in turn can call generic optimisers.

4.3. Performance and flexibility

Performance is not really an issue when we are considering running a rainfall-runoff model, even over centuries of daily data, given the processing power available these days on any personal computer. However, the nature of the calibration process is such that the complexity of the problem is exponentially related to the number of parameters in the model and the optimiser. Hence, attention needs to be paid to the trade-off between flexibility and performance. Most elementary operations which have a substantial computational overhead (e.g. involving a Hashtable, or using dynamic invocation), are performed once for each run over the data, and are not of major concern. However, as pointed out by Rahman (2003), the use of a Model Runner which relies at each step on introspection for running the models, a performance ratio of about 10 can be observed compared to the case of a model directly handling input and output time series. Note that the gain would however be less than this ratio for an overall calibration process. This is a clear example of a trade-off between flexibility and performance, since this is a consequence of the minimalist approach of the core models, and the reliance on discovery at run-time of model properties.

In order to improve performances without compromising flexibility, a tool has been implemented to generate once at run-time a wrapper for any temporal model. This wrapper handles time and time series explicitly as opposed to the core model, and by construction has an intrinsic knowledge of the model structure (see Figure 4). This is possible on the .NET platform since it contains advanced tools in its 'System.Emit' namespace to generate additional code at run-time in an elegant fashion.

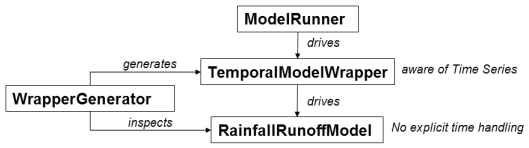

Figure 3. Model wrapper generation

The wrapper allows the model runner to avoid model introspection at run-time to set and read the values of its properties. The code generated can then be saved in a new assembly (DLL) and reloaded at next run-time. In a benchmark test where AWBM was run using the Pattern Search optimiser, run times were reduced by a factor of 2.5 by using this method..

5. CONCLUSION

The RRL has been developed to satisfy end-user needs in the water engineering community, but has also been designed to fit into a larger component-based modelling system known as the Catchment Modelling Toolkit. Its design has entailed the separation of features such as data I/O, visualisation, optimisation methods and the rainfall-runoff models themselves. In each case,

these features have been developed as re-usable components that can be adopted in different parts of the Toolkit, and as code which is extensible in itself. The use of the .NET environment and the TIME modelling framework in particular, aided the attainment of these objectives.

A beta version of the RRL is now available for download at *www.toolkit.net.au*. Further development of the RRL is envisaged, an will largely be driven by the feedback from end-users and the evolving requirements from other projects involved in the Catchment Modelling Toolkit initiative.

6. REFERENCES

Arsham, H., M. Gradisar, and M.I. Stemberger, Linearly constrained global optimization: a general solution algorithm with applications, Applied Mathematics and Computation, 134, 345–361, 2003.

Boughton, W., and R. Mein, AWBM catchment water balance model, calibration and operation manual, *www.catchment.crc.org.au,* 1996.

Chiew, F.H.S., P.J. Scanlon, R.A. Vertessy, and F.G.R. Watson, Catchment scale modelling of runoff, sediment and nutrient loads for the Southeast Queensland EMSS, Technical Report 02/1, *Cooperative Research Center for Catchment Hydrology*, 2002.

Duan, Q., S. Sorooshian, and V. Gupta, Effective and efficient global optimization for conceptual rainfall-runoff models, *Water Resources Research,* 28(4), 1015–1031, 1992.

Duan, Q., S. Sorooshian, and V. Gupta, Optimal use of the SCE-UA global optimization method for calibrating catchment models, *Journal of Hydrology,* 158, 265–284, 1994.

Hookes, R., and T. Jeeves, Direct Search solution of numerical and statistical problems, *J. Assoc. Comput. Mach.,* (8), 212–229, 1961.

Jakeman, A. J., and G. M. Hornberger, How much complexity is warranted in a rainfall-runoff model, *Water Resources Research,* 29(8), 2637–2649, 1993.

Perrin, C., C. Michel, and V. Andréassian, Does a large number of parameters enhance model performance? Comparative assessment of common catchment model structures on 429 catchments, *Journal of Hydrology,* 242, 275-301, 2001.

Rahman, J.M., S.P. Seaton, and S.M. Cuddy, Making Frameworks More Useable: Using Model Introspection and Metadata to Develop Model Processing Tools, *Environmental Modelling and Software*, in press, 2003.

Rosenbrock, H.H., An automatic method for finding the greatest of least value of a function, *Computer Journal,* 3, 175–184, 1960.

Sorooshian, S., Q. Duan, and V. Gupta, Application of global optimization to the Sacramento soil moisture accounting model, *Water Resources Research,* 29(4), 1185-1194, 1993.

Tan, B.Q., and K.M. O'Connor, Application of an empirical infiltration equation in the SMAR conceptual model, *Journal of Hydrology,* 185, 275-295, 1996.

Wang, Q.J., The genetic algorithm and its application to calibrating conceptual rainfall-runoff models, *Water Resources Research,* 27(9), 2467-2471, 1991.