*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*

A data model for a sustainable management of parameter sets and optimization results

Jean-Michel Perraud, Biao Wang, Jai Vaze
Water for a Healthy Country Flagship, CSIRO Land and Water, Canberra ACT 2601, AUSTRALIA, firstname.lastname@csiro.au

Abstract: The Catchment Water Yield Estimation Toolset (CWYET) is a software toolset for estimating daily catchment water yield and runoff characteristics in regulated and unregulated catchments. It is used to estimate water yield typically for hundreds of catchments, featuring capabilities for calibration, catchment cross-verification, multiple model structures, and scenario modelling such as impact of climate change. Due to the large number of combinations of these ensembles, using simplistic text files to store model parameterization can become at the very least logistically tedious. Of more concern, this is a brittle storage system that is inadequate to underpin provenance tracking and reproducibility. The issue is not unique to CWYET, and there are substantial efforts in modelling software products to use state of the art Object Relational Model (ORM) tools to store on disk model structure and parameterisation. In this paper we present how we used the Microsoft Entity Framework version 4.1 to implement a database schema to store and manage a large number of model parameterizations. We summarise the main use cases for these model parameterisations. The data store is decoupled from a particular modelling framework or tool, and not limited to CWYET. We derive the schema of the database from the characteristics of the results of optimization tools, and the information that is determined as necessary from the use cases. We illustrate how the library of optimization results is accessed to assess visually the performance of model calibration on a large number of catchments. We demonstrate the use of this repository of parameter sets from a Python interpreter and from the scientific workflow software Hydrologist's Workbench.

***Keywords**: Entity Relationship Model; optimization; logging; parameterization*

1 INTRODUCTION

The Catchment Water Yield Estimation Tools (CWYET) is a modelling framework for estimating daily catchment water yield and runoff characteristics in regulated and unregulated catchments (Vaze et al. [2011a] and Vaze et al. [2011b]). One background motivation for this toolset is a need to develop a modelling framework which can be used by different water management and research agencies across Australia that allows them to undertake the modelling in an objective, consistent and reproducible manner.

CWYET has been applied in research and decision support projects, some with a substantial requirement for reproducibility and an audit trail. This can be a challenge in a context where the toolset will still need rapid evolution for the research purpose. The computational load required by the tool, due to the large number of combinations of alternate inputs, catchment models, calibration techniques, etc. often requires distributed computation on high performance computing facilities. These contexts have some bearing on how the data and model configurations are structured.

Perraud et al. [2012] describes a data layer for the management of models and data associated with CWYET. Model parameterization is an integral part of the overall model configuration, and is covered as one configuration element. However, the storage and management of parameter sets has broader needs, and could not be covered in that paper. Besides, CWYET models are calibrated using an optimization software framework that is purposely not coupled to CWYET toolsets, and the data layer capturing these parameter sets in a calibration context is distinct from CWYET.

We propose a software solution in the form of a data layer using current or recent technologies, and the current recommended practices for architecting it. A background motivation, but an important one, is the aim to access libraries of parameter sets from a scientific workflow software tools, the Hydrologist's Workbench (Cuddy and Fitch [2012]). The case studies we use in this paper derive from the design and implementation of calibration workflows (Perraud et al. [2010]).

We will end this section with a note on terminology. The term "parameter set" is usually understood in the hydrologic modelling domain as a set of continuous numeric values describing some particular states of a model controlling its behaviour. This is actually a subset of a broader concept, let us call it "system configuration", where states may be discrete values (logical, categories, integers) or even mathematical functions. In this paper, for the sake of readability, we will mostly use the term "parameter set", even to cover the potentially larger scope.

2 NEEDS

An anecdotal way of summarising the needs is by reporting a not so hypothetical question: "Do you remember the calibrations that we did for model XYZ in spring 2007 on the 240 catchments? Where are the parameter sets? We need them for 60 of those catchments". Of course, this was one of many calibrations performed around that time, and organisational changes since meant the data had moved location on the file system, not to mention staff moving on to other projects. In the rest of this paper we will mostly consider the case study of managing results and logging information from a calibration process.

More formally and more generally, the main specifications of a manageable repository of parameter set are as follow:

1. The software entities capturing the parameterization information should be independent of specific modelling toolsets, notably to facilitate the transfer of parameters between different model implementations.

2. The data model must help to support the capture of the provenance information in the overall modelling workflows. Metadata must be an integral part of the data model

3. The state of the art software patterns used in persistence and data layers should be considered

4. The repositories of parameter sets should be easily searchable. The search queries should be structured rather than full text.

5. The data layer must be extensible and the structure may change. It must be extensible to types of model parameterisation information which is other than in the form of a hypercube (i.e. several numeric variables with bounds), as is typically the case for most calibration algorithms used in the hydrology domain. The need to be able to change the structure is recognition that more often than not updating design specifications will require a change to the data model that requires more than extensibility, raising the issue of data migration and backward compatibility.

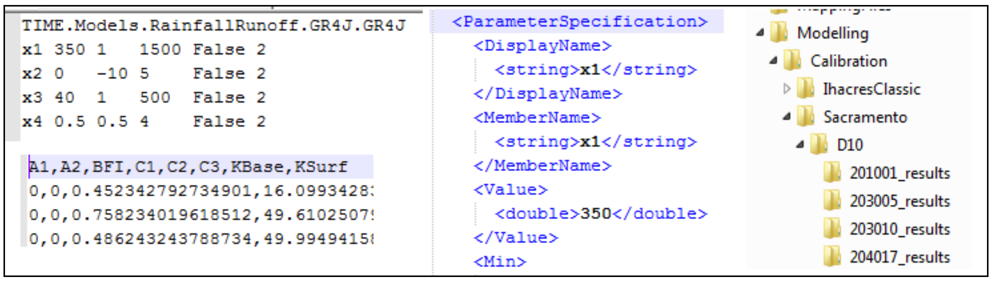
6. The capture of parameter sets should be usable not only for final results from a calibration process (parameter set and model fitness statistics), but also to capture a detailed log. In other words, it should scale up well to handle several orders of magnitude more information than 'just' the results of calibration processes.

7. The software tasks arising from the persistence mechanism itself should be reduced to a minimum.

3 RELATED WORK

The area of modelling and management of observational data is very active (see for instance http://www.opengeospatial.org/projects/groups/waterml2.0swg, accessed 2012-03). Comparatively, literature on the management of model configurations and in particular model parameterisation appears sparser. Marsh et al. [2006] states that "the parameter sets resulting from modelling activities are poorly reported and as a consequence they are undervalued". It proposes a software package called the Catchment Modelling Parameter Library. The paper identifies the needs to capture parameterization in the domain of environmental modelling. The application comprises a user interface, search capabilities, database and reporting. The clear intent is to capture the information with the modeller in mind, allowing for many forms of ancillary information for each parameter. The capture of calibration log information or ensemble of parameter sets is not explicitly considered in the scope of the Catchment Modelling Parameter Library.

The past few years have seen the emergence of several metaheuristics software frameworks (see Lukasiewycz et al. [2011a] and its references). These frameworks are largely oriented towards research needs, with an emphasis on the "white box" capabilities of the engines, giving access to many algorithmic options to investigate optimization. The storage and management of parameter sets is not put forward as a core capability, although significant capabilities are of course present to investigate the log and output of optimization processes. jMetal (Nebro and Durillo [2011], Durillo and Nebro [2011]) uses text files to store the results of an optimization, and opt4J 4.5 (Lukasiewycz et al. [2011b]) includes facilities to log to a tab-separated values file format. CWYET is built on The Invisible Modelling Environment (TIME – Rahman et al. [2005]) which serialises parameter sets as text files (XML or plain text), and post processing tools help to collate parameter sets to comma-separated value files (Figure 1).



```
TIME.Models.RainfallRunoff.GR4J.GR4J
x1 350 1   1500 False 2
x2 0   -10 5    False 2
x3 40  1   500  False 2
x4 0.5 0.5 4    False 2

A1,A2,BFI,C1,C2,C3,KBase,KSurf
0,0,0.452342792734901,16.0993428
0,0,0.758234019618512,49.6102507
0,0,0.486243243788734,49.9949415
```

```
<ParameterSpecification>
  <DisplayName>
    <string>x1</string>
  </DisplayName>
  <MemberName>
    <string>x1</string>
  </MemberName>
  <Value>
    <double>350</double>
  </Value>
  <Min>
```

Figure 1 Typical text format for parameterization

Such text formats are mostly adequate to store the definition of parameter sets. There are several difficulties that become apparent when scaling up the usage to managing ensembles of parameter sets. The textual and file-based nature of the format introduces a performance penalty due to the parsing and input-output, but this is a lesser logistical concern. There is no consistent mechanism to group related parameter sets together, and at best an inflexible mechanism to relate the parameter sets to the metadata with the information that explains their origin (e.g. the objective scores obtained through the optimization process, the steps in the algorithm, etc.). Folder and file name conventions become ad-hoc ways to store the various metadata values as illustrated by Figure 1. The code to parse and generate these file paths is tedious, but of greater concern it is inflexible.

4 DESIGN AND IMPLEMENTATION

We approach the design of the data layer by considering the nature of a collection of parameter sets in a population based, multi-objective optimization process (Talbi [2009]). In this section we reference in text the list of specifications in section 2, with the numbering of needs indicated between parentheses.

The highest level data entity is an ObjectiveResultsCollection (Figure 2). It can represent any group of related system parameter sets, for instance the population of parameter sets at a stage of a genetic algorithm. The elements of this collection are ObjectiveScoreCollection. In the context of a multi-objective optimization, each parameter set will be associated with one or more scores (e.g. sum of squares errors and bias), based on the formulation of the optimization problem (2,4). To address a key need for extensibility of the data model (1,5), the ObjectiveScoreCollection references a generic SystemConfiguration. An HyperCube, representation of the most traditional parameter set in hydrology, is thus not the only possible type of parameterisation stored.

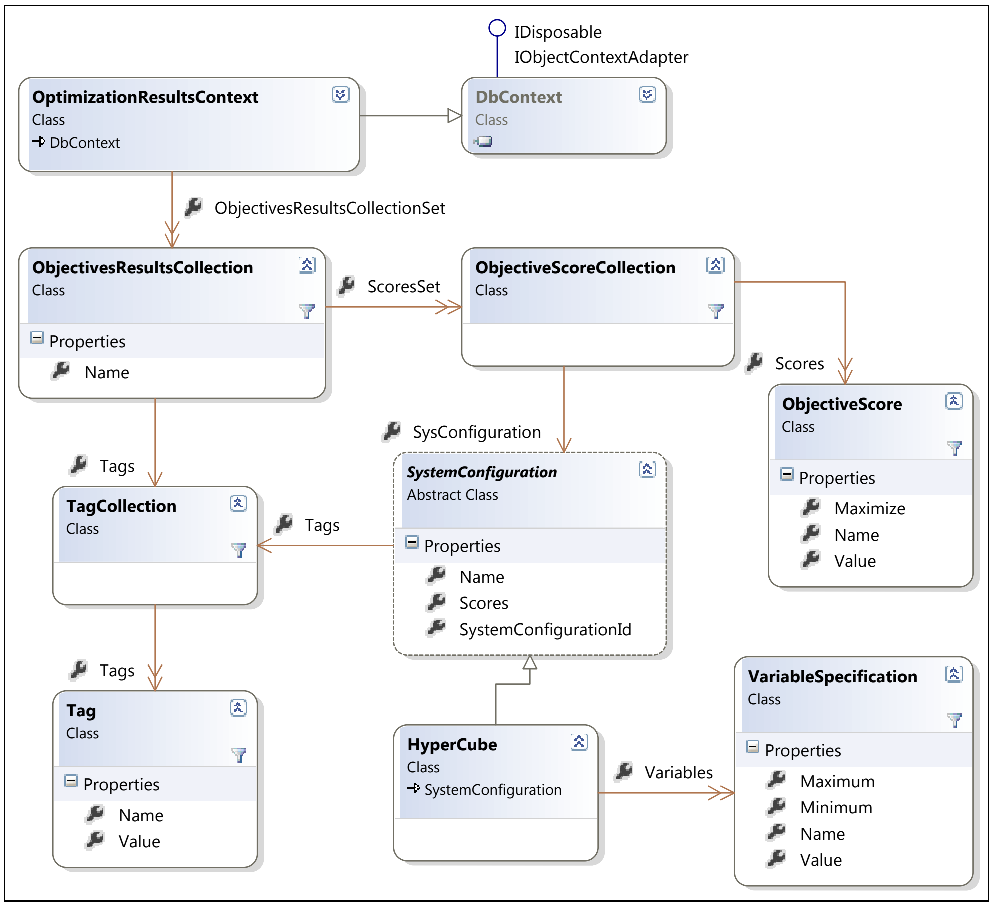


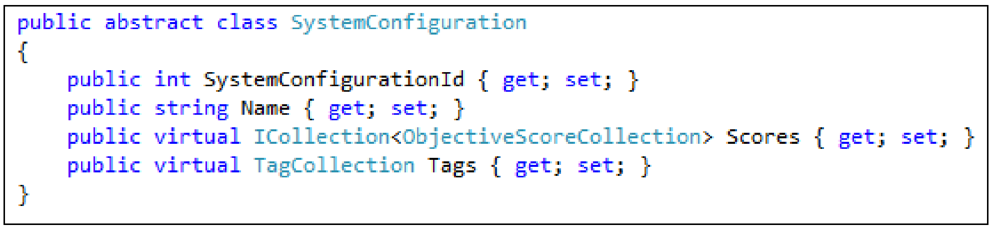
Figure 2 Entities and DB context of the data model

The metadata is captured in the name property of some of the entities, and more importantly by using key-value pairs as tags for the high-level ObjectiveResultsCollection and the SystemConfiguration. The presence of these key-value entities facilitates handling categorical information as metadata (2,4,6), and is much more flexible than the use of file path name conventions evoked in the previous section to manage text-based representations.

We implement the data model using Entity Framework v4.1. (EF) (http://msdn.microsoft.com/en-us/data/ef, last accessed 2012-03-05). Other technologies could be used; this choice of data access technology is based on a positive experience with its use for the CWYET model configuration (Perraud et al [2012]). In this present paper we use the Code First paradigm (Lerman and Miller [2012]) to implement the entities and derive the data persistence layer on a SQL Express database.

The key aspect of this paper is the data model in Figure 2. The choice of technology is not coupled to the data model, following the usual best practices in software engineering regarding data persistence layer (3). Subsequent evolutions of this system may in fact move to other ORM tools than EF and SQL Express.

That being said, the current use of EF and SQL Express brings some acknowledged benefits. As shown in Figure 3, the code definition of the data model (properties of the entities, and relationship between entities) is very succinct, and has no dependency on EF classes. The most recent releases of EF also automatically take care of much of the tedium with respect to creating the back-end SQL database. Importantly, the upcoming releases will have improved capabilities to migrate (i.e. upgrade) databases in the likely event of a change in the structure of the data model (5, 7).



```
public abstract class SystemConfiguration
{
    public int SystemConfigurationId { get; set; }
    public string Name { get; set; }
    public virtual ICollection<ObjectiveScoreCollection> Scores { get; set; }
    public virtual TagCollection Tags { get; set; }
}
```

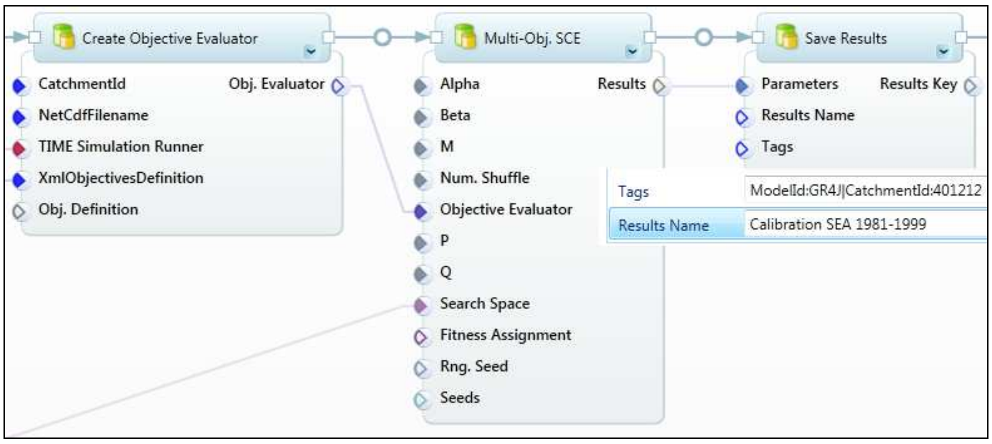
Figure 3 Definition of entities with the “Code First” approach

Database design and management requires skills and know-how sensibly different from object oriented programming

5 EXAMPLE OF APPLICATIONS

One central motivation for a consistent data model to manage parameter sets is to make them available to a scientific workflow tool, the Hydrologist’s Workbench (Cuddy et al. [2010]). We demonstrate in this section how the parameter set data model is used to log the behaviour of a calibration workflow. The log is queried to retrieve particular stages of the calibration process, to then visualise the behaviour.

Figure 4 shows a portion of a workflow where the results of a multi-objective version (Vrugt et al. [2003]) of the Shuffled Complex Evolution algorithm are captured and saved using the system we just described. The design of the activity “Save Results” is purposely Spartan and reflects the central role of the metadata of the results, most notably the additional tags a user may want to attach to these calibration results.



Tags	ModelId:GR4J|CatchmentId:401212
Results Name	Calibration SEA 1981-1999

Figure 4: Saving optimization results from a calibration workflow

Once saved to the database storage, the parameter sets can be retrieved by a variety of software tools. The advantage of Object Relational Mapping (ORM) tools is usually the ready availability of high-level querying capabilities, addressing need (4). Users not at ease with using SQL statements can produce queries in their language of choice. Figure 5 shows a typical query performed to extract log information from an optimization algorithm. In this instance, IronPython (http://ironpython.net) is used as an interactive language. It is build on top of .NET and one advantage demonstrated in Figure 5 is the ready availability of the Language Integrated Query mechanism (LINQ). The query is expressed succinctly using statements in the python language. The ORM takes care of the generation of SQL statements.

```
clr.ImportExtensions(System.Linq)
nameStringMatch = 'LogSce'
fTag = {'CalibName': '20120119061229', 'Category' : 'Shuffl.*'}
dbContext = OptimizationResultsContext()
results = dbContext.ObjectivesResultsCollectionSet
pSets = results.Where(lambda x: x.Name.Contains(nameStringMatch)).ToArray()
pSets = pSets.Where(lambda x: hasTags(x, dict([fTag]))).ToArray()
createCalibrationLogPlot(pSets)
```
Figure 5 Example Python code to query calibration logs

The extraction of this information is then used to visualize the behaviour of the optimization process to assess its performance (Figure 6). As it happens, this visualization is done using the R software, and the extracted data is first passed between software applications as a CSV file. In passing, this illustrates two things. First, the *conceptual* data model and the retention of metadata information matters more than the details of the back end storage. Second, the use of CSV is only transient and a convenience to quickly get data into R. Direct access to the SQL database from R is possible but more complex than is required for the purpose at hand.

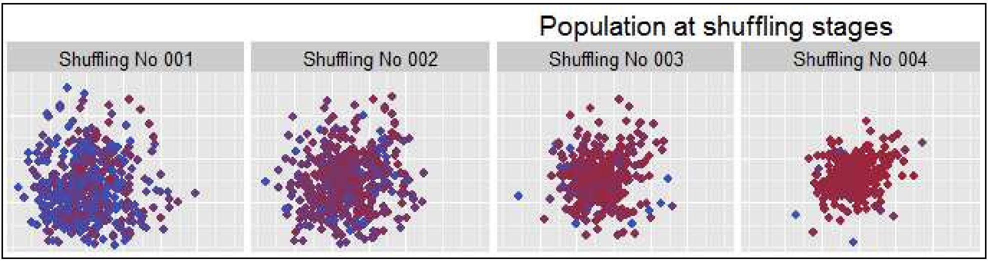


Figure 6 Example of visualization of parameter sets

6 DISCUSSION

The case study application of this paper is derived from the need to log the process of model calibration in a scientific workflow. The data model and current reference implementation with EF proposed in this paper successfully supported this need. Of course, there are still known technical shortcomings such as database performance, and the downsides of a highly framework-independent data model, etc. However, we will focus this discussion on one key aspect, metadata.

Which metadata tags to use on the parameter sets (or groups thereof) for optimization logs is mostly dictated by obvious algorithmic considerations (shuffling stage, etc.). Even then, the end goals requiring the persistence of the parameter sets do influence the choice of metadata tags. Conditional plots (also known as “facets”) such as Figure 6 rely on these metadata tags, so the final visualisation aimed for can require the use of additional tags during the logging operation of the calibration process. Insufficient foresight in the choice of metadata can mean that some data queries and visualisation become infeasible.

Choosing appropriate metadata tags is even more difficult in the context of transfer of model parameters to ungauged catchments. The choice is much more dependent on the purpose of the modelling than the relatively self-evident tags from the log of an optimization algorithm. Marsh et al [2006] describes a database scheme where the information on the parameter sets is free-form, and indeed can even be pictures.

It seems very worthwhile to bridge the use of the data model presented in this paper and its associated implementation, driven mostly by analytical needs for an optimization framework, with an application oriented towards end-user such as that in Marsh et al. [2006]. A priori this requires designing systems to offer a different viewpoint on the parameter sets obtained from a calibration process. Technically, relational databases management systems offer a mature set of tools to repurpose this data with additional metadata, for further use. Whatever the tools, the challenge of designing appropriate views remains, and it is intertwined with the process of data management and curation.

7 CONCLUSION

The data model and implementation proposed in this paper has been successfully used to investigate the behaviour of an optimization algorithm. While conceptually not too dissimilar from prior file- and text-based systems, the implementation with Entity Framework permits a significantly more manageable and versatile tool. The overarching goal of this data model is to address the shortcomings perceived over many years in managing multiple modelling scenarios and their model parameterisation. Coupled with user-oriented tools to add richer semantic information to these parameter sets, this data model has the potential to bring parameter set repositories as first-class curated data stores in the Hydrologist’s Workbench.

ACKNOWLEDGEMENTS

We gratefully acknowledge Nick Marsh and Ben Leighton for the helpful comments on this paper.

REFERENCES

Cuddy, S. & Fitch, P. “Hydrologists Workbench–a hydrological domain workflow toolkit”, International Congress on Environmental Modelling and Software, Ottawa, Ontario, Canada, July 5 - 8 2010

Durillo, J.J., Nebro, A.J., jMetal: A Java framework for multi-objective optimization, *Advances in Engineering Software*, Volume 42, Issue 10, 760–771, October 2011

Lerman, J., Miller, R., Programming Entity Framework: Code First, O'Reilly Media, Inc., ISBN-13: 978-1-4493-1294-7, 2011

Lukasiewycz, M., Glaß, M., Reimann, F., Teich, J., Opt4J – A Modular Framework for Meta-heuristic Optimization, GECCO '11 Proceedings of the 13th annual conference on Genetic and evolutionary computation, Dublin, Ireland, July 12-16 2011a

Lukasiewycz, M., Glaß, M., and Reimann, F., Opt4J Documentation, The Optimization Framework for Java – Version 2.5, Publication date 2011-12-22, 2011b

Marsh, N., Tennakoon, S., Arene, S., and Banti, F., Catchment Modelling Parameter Library: Development and Population, 30th Hydrology and Water Resources Symposium, Hobart, TAS, 4 - 7 December 2006

Nebro, A.J., Durillo, J.J., jMetal 4.0 User Manual, November 10, 2011

Perraud, J.-M., Bai, Q., & Hehir, D., On the appropriate granularity of activities in a scientific workflow applied to an optimization problem, International Congress on Environmental Modelling and Software, Ottawa, Ontario, Canada, http://www.iemss.org/iemss2010/proceedings.html, July 5 - 8 2010

Perraud, J.-M., Wang, B., Vaze, J., Building a data model for a water yield estimation software toolset, 2nd IAHR Europe Congress, Munich, 27-29 June 2012.

Rahman, J.M., J.-M. Perraud, S.P. Seaton, H. Hotham, N. Murray, B. Leighton, A. Freebairn, G. Davis & R. Bridgart, Evolution of TIME. 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, pp. 697-703. ISBN: 0-9758400-2-9, December 2005

Talbi E.-G., Metaheuristics: from design to implementation, ISBN: 978-0-470-27858-1, Wiley, 624 pp., 2009.

Vaze, J., Chiew, F. H. S., Perraud, JM., Viney, N., Post, D. A., Teng, J., Wang, B., Lerat, J., Goswami, M., Rainfall-runoff modelling across southeast Australia: datasets, models and results. *Australian Journal of Water Resources*, Vol 14, No 2, pp. 101-116, (2011a)

Vaze J, Perraud J, Teng J, Chiew F, Wang B, Yang Z., Catchment Water Yield Estimation Tools framework (CWYET). 34th IAHR World Congress 2011 - Balance and Uncertainty Water in a Changing World. Brisbane, Australia, (2011b).

Vrugt, J. A., Gupta H. V., Bastidas L. A., Bouten W., Sorooshian S., Effective and efficient algorithm for multiobjective optimization of hydrologic models, *Water Resour. Res.*, 39(8), 1214, doi:10.1029/2002WR001746, (2003)