19th International Congress on Modelling and Simulation, Perth, Australia, 12–16 December 2011
http://mssanz.org.au/modsim2011

Seasonal Streamflow Forecasting with a workflow-based dynamic hydrologic modelling approach

R. Laugesen[a], N.K. Tuteja[b], D. Shin[b], T. Chia[b], U. Khan[c]

[a]*Bureau of Meteorology, Sydney, NSW, Australia,* [b]*Bureau of Meteorology, Canberra, ACT, Australia,*
[c]*University of New South Wales, Sydney, NSW, Australia*
Email: r.laugesen@bom.gov.au

Abstract: As part of its new responsibilities under the Commonwealth *Water Act* 2007, the Bureau of Meteorology commenced a Seasonal Streamflow Forecasting (SSF) service in December 2010. This service delivers probabilistic 3-monthly streamflow forecasts derived from the statistical modelling approach at target locations and total inflows into water storages (www.bom.gov.au/water/ssf). In parallel, the Bureau has developed a new modelling system for generating seasonal streamflow forecasts using the dynamic hydrologic modelling approach to complement forecasts derived from the statistical approach. This new approach uses downscaled rainfall hindcasts and forecasts from the Bureau's seasonal climate model POAMA (Predictive Ocean Atmosphere Model for Australia) and lumped rainfall-runoff models to generate ensembles of streamflow.

This new system called the Dynamic Modelling System (DMS) allows hydrologists to calibrate, validate, hindcast, and forecast catchment streamflow in ensemble mode using a workflow-based process. This workflow-based approach was found to greatly improve productivity during experimental evaluation because it allowed investigations to be declared as a workflow and then ran in an automated way. Investigations frequently involved a large number of combinations of catchment input data, rainfall runoff models, numerical optimisers, objective functions, and accumulation schemes. Additionally, the unique characteristics of the DMS architecture and adoption of a standardised data format allowed development, testing, and modelling work between three collaborating groups to proceed in parallel because data and model parameterisations could be shared easily between the different systems due to identical components. The solution design resulted in many benefits in terms of implementation and application efficiencies.

DMS development leveraged the currently available fit-for-purpose hydrologic models and components from eWater CRC (Cooperative Research Centre), seasonal rainfall downscaling technologies developed by the Centre for Australian Weather and Climate Research (CAWCR), targeted research by the Commonwealth Scientific and Industrial Research Organisation (CSIRO) through the Water Information Research and Development Alliance (WIRADA) program and the university sector in the context of seasonal streamflow forecasting, particularly focused on hydrologic modelling and predictive uncertainty analysis techniques.

A functional and technical specification document was created following a requirements elicitation and stakeholder engagement process. A solution plan was then designed which satisfied those requirements. Implementation proceeded following a guiding principle of pragmatism which resulted in a system leveraging a number of different software development methodologies, languages, and libraries, each selected for it's strengths to satisfy the requirements of that component. This paper will discuss the specific technical considerations used in development of the DMS and the costs and benefits of the solution architecture.

Experimental evaluation of the dynamic modelling approach using this system has demonstrated significant skills in the seasonal streamflow forecasting capability. Based on the results, this system will transition into an operational component for delivering inputs to the SSF.

Keywords: *Streamflow, forecast, model, system, workflow, Bureau of Meteorology*

1. INTRODUCTION

Within the Climate and Water Division of the Australian Bureau of Meteorology and its research partners, a need was identified to evaluate seasonal streamflow prediction using a dynamic modelling approach. The dynamic approach relies on the assessment of streamflow forecasting skills using downscaled climate forecasts from seasonal global climate models in conjunction with hydrologic models of varying levels of complexity (Tuteja et al., 2009). This approach will complement the existing statistical approach used in the operational Seasonal Streamflow Forecasting service (SSF) (Plummer et al., 2009).

The statistical approach is based on robust science and is implemented in a well designed and operationally reliable system called Wafari (Wang and Robertson, 2011 and Shin et al., 2011). A new system was required to generate the outputs for evaluating a dynamic modelling approach because it has different requirements to the current statistical approach. The resulting system has a flexible architecture and an efficient implementation with different characteristics to Wafari.

In this paper we will outline the system requirements, describe the adopted solution design, outline the implementation of that solution, consider the key strengths and weaknesses of the chosen solution design, and finally discuss the method for using the system.

2. SYSTEM REQUIREMENTS

A project plan outlining 'hydrologic scope' of the experimental evaluation of the dynamic approach including key milestones and deliverables was developed and approved by the Bureau (Tuteja et al., 2009), which identified the need and high level guiding principles for development of an experimental dynamic modelling system. A process of stakeholder engagement and requirements elicitation was then conducted to capture the functional and technical requirements of the required system in a specifications document (Laugesen et al., 2009).

Broadly, the system was required to provide the capability of hydrologic modelling at catchment scale in calibration, validation, forecast, and hindcast modes using a number of different rainfall-runoff models, numerical optimisers, objective functions, accumulation methods, and diagnostic statistics and plots. Investigation of a large number of combinations of these options was required which necessitated both a simple way to define the steps to be performed and an automated way to execute them.

Five key issues were identified from the requirements which constrained the possible solutions designs:

1. An existing framework should be utilized for all core rainfall-runoff modelling functionalities.
2. Accommodating new functionality into the system should be possible without changing the existing architecture or components.
3. The team has experience in the Python and R programming languages and this should be utilized, especially for the implementation of visualisation and analysis of results.
4. The system should operate by executing workflows of tasks in an automated way which may require days of computer time to complete.
5. End users of the system are hydrologists who are familiar working with simple ASCII format files for input data and output results analysis and are not proficient in software engineering or information technology.

The Invisible Modelling Environment (TIME), developed by the eWater CRC was available and strongly recommended to the Bureau for use in this system (Rahman et al., 2003). This mature library has had a significant investment from many stakeholders in the Australian water community over the last decade and by leveraging it the time required for system implementation is significantly reduced and confidence in the generated results is enhanced.

3. SOLUTION DESIGN

Naturally, the primary objective of the solution design was to address the identified requirements. A secondary objective of the design was to separate the concerns of the various system stakeholder types as much as possible to remove complexity introduced through interaction between them. This secondary objective was considered important since the system stakeholders; managers, hydrologists, developers, and information technologists approached the system with different contexts, expectations, and specialist language styles (O'Toole and Laugesen, 2011). The resulting solution design allowed managers to exercise project and resource oversight, hydrologists to develop workflows and analyse results, developers to add components, and information technologists execute workflows and monitor the system. Each stakeholder was

then able to work more effectively with reduced communication overhead because the system architecture did not require each user to have knowledge of the tasks performed by other users.

Because no off-the-shelf system that satisfied the identified requirements existed, a solution plan was designed for in-house implementation called the Dynamic Modelling System (DMS) (Laugesen, 2009). The DMS is based around an architecture composed of loosely-coupled components driven by declarative workflows (Gamma, 1995). The standard use-case of a modeller using the system is illustrated in Figure 1. This figure illustrates the idealised flow of information (blue arrows) and control (red arrows) through time (left to right) between the modeller, the DMS, and the filesystem. The specifics of the file types are addressed later in the paper.

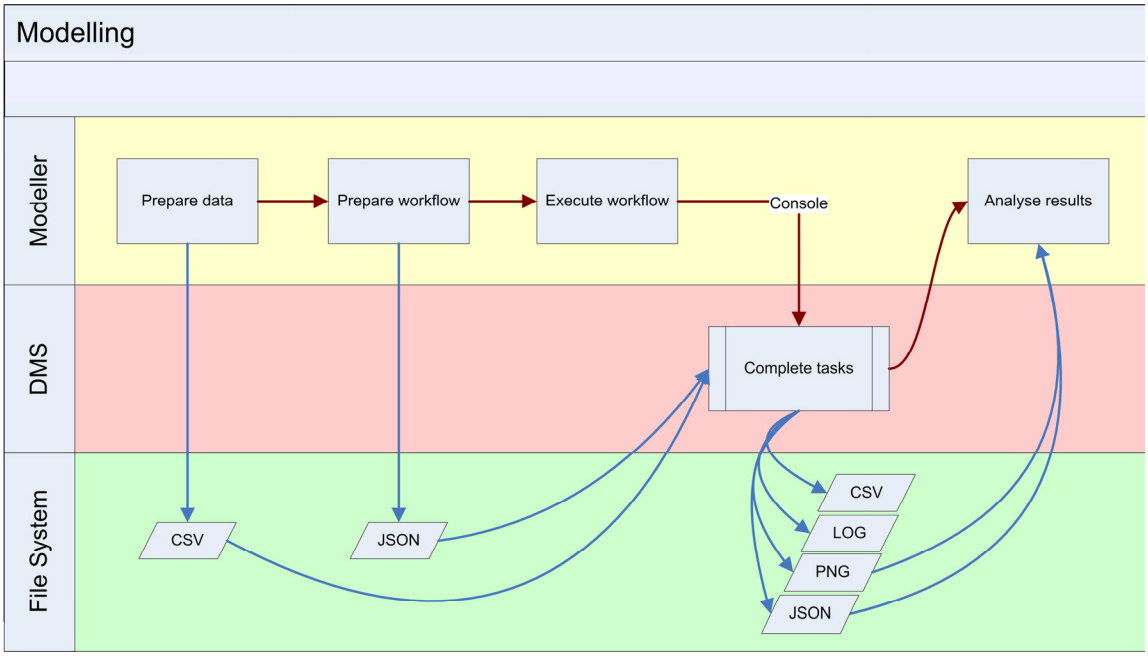

Figure 1: Cross functional flow chart of common use-case of a modeller using the proposed Dynamic Modelling System

The components of the DMS are illustrated in Figure 2 and consist of Dynamic Modelling Controller (DMC), Workflow Manager (WFM), and a collection of Utility Tools (UT). The DMC is responsible for modelling and simulation of the rainfall-runoff process, accumulation of timeseries, and calculation of simple model performance statistics. The WFM is responsible for parsing sequential workflows and executing the resulting tasks in the correct sequence. Finally, the UT are responsible for post-processing, analysis, and visualisation of the results generated by the DMC.

Workflows may contain any number of tasks and each task may have any number of dependencies on other tasks. A

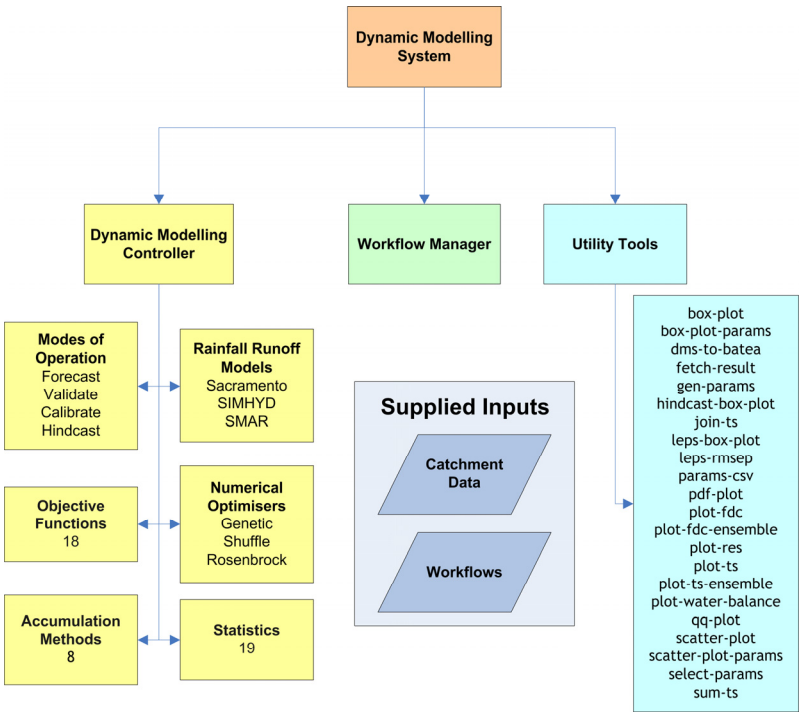

Figure 2: Structure diagram for the Dynamic Modelling System

two-step approach to defining workflows was adopted: Step Declaration – a sequence of steps describing an abstract scenario is defined, and Process Definition – the Step Declarations are applied over a large number of concrete scenarios. This approach was beneficial because it made the purpose of each workflow clear and made it easy verify it was working as expected by simply executing the Step Declaration over a single concrete scenario. Additionally, either the Step Declaration or Process Definition could be reused in a new workflow independent of the other. The process of running an individual Step Declaration is illustrated in Figure 3 and is essentially an exploded view of the single "Complete Tasks" process in Figure 1. The WFM processes a Step Declaration file by working through all the defined tasks, each task executes a component which reads input data from a specific filesystem location, processes that data, and then writes output data to a different location. The output data of one component may potentially become the input of another. The Process Definition file simply repeats the Step Declaration for all required concrete scenarios (not shown in Figure 1).

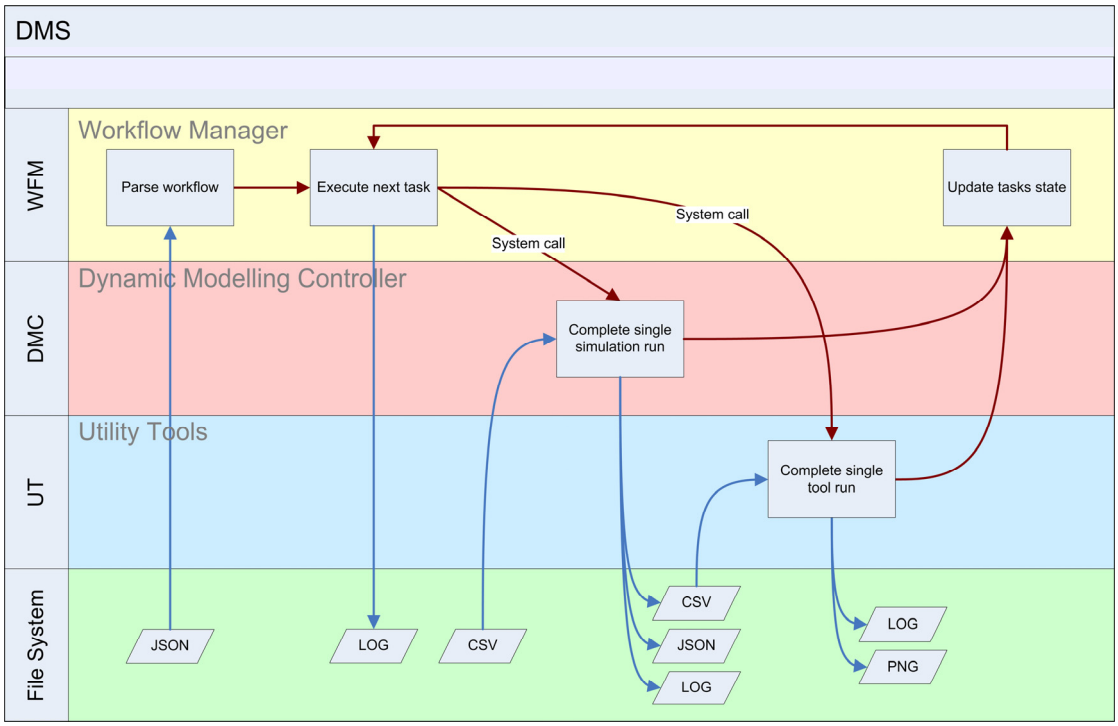

Figure 3: Cross functional flow chart of flow of information and control between components of the proposed Dynamic Modelling System

All components are executed through a single operating system call with arguments specifying the input and output data locations and various component specific arguments. A simple comma-separate-value (CSV) data file format was used for all input and output data transfer. The specific structure of this CSV format was carefully designed and standardised to allow components to develop independently and enable easy sharing of data with partners. To simplify the implementation of concurrency these files are considered immutable; once created they can't be written to again. The decision to use operating system calls and simple files instead of a more sophisticated approach was to simultaneously address all five of the identified constraints on the system in the most parsimonious way. Although it was possible to identify alternative approaches that were more efficient in terms of computing resources when evaluating each constraint in isolation, it was not possible to find an approach which simultaneously satisfied them all. The overhead introduced by this approach was found to be negligible and will be discussed later in this paper.

4. IMPLEMENTATION

The DMS solution design was implemented through the following broad structural components which are illustrated in Figure 2:

- The DMC was implemented using an object orientated methodology in C# to satisfy the requirement to leverage the TIME framework. A Model-View-Controller design pattern was used along with the Adapter and Bridge design patterns (Gamma et al., 1995). This is the only component which is dependent on Microsoft Windows; all other components are platform independent.

- The UT category consists of a large number of individual applications. Most were implemented using a structured procedural methodology in Python and leveraged the Numpy and Matplotlib libraries (Oliphant, 2007). Some were created in R
- The WFM was implemented as a finite-state machine using an object orientated methodology in Python. This repeats the simple rule for each task – wait a random and short amount of time, execute the task if all its dependencies have completed and a CPU core is available.

The two-step workflows were implemented in the following way using simple text-editable files:

- First, the Step Declaration is implemented using a schema defined in JSON (Java Script Object Notation), a lightweight text-based human-readable data format similar in scope to XML. This Step Declaration fully describes the sequence of steps required to complete an abstract scenario as a set of task blocks. Each task block first defines any dependencies it has on other tasks, then defines the system-call required to execute the required component, and finally defines any component specific information which is required.
- Secondly, the Process Definition is implemented using a simple Python script which iterates over various combinations of modelling scenarios (catchment, model, optimiser, etc). As each specific scenario is processed an instance of the WFM is launched and passed the Step Declaration JSON file along with any parameters describing the specific scenario, the WFM parses the JSON file and launches each component as required.

An example set of workflow declaration files are shown in Figure 4 which illustrates the case where a hydrologist wants to calibrate a rainfall-runoff model and generate a hydrograph for three catchments; in this case the data for each catchment is stored in a separate directory with the same name. This example demonstrates the concept of using a Workflow Variable to pass scenario specific information from the Process Definition to the Step Declaration; in this case the V1 variable is used to represent the catchment name. The Process Definition file loops through each catchment in the list and simply passes a new value for V1 (the current catchment name) to the WFM along with the Step Declaration file, the WFM then replaces every instance of $V1 with that value before executing the two tasks. The second task will always be executed after the first because of the dependency which has been declared. Up to fifteen separate Control Variables may be used which makes for a very flexible and general workflow definition process, this facilitates the creation of arbitrarily complex workflows which may be executed in a consistent and simple way.

Step Declaration

```
{
  "tasks": [
    {
      "id": 1,
      "dependant_ids": [],
      "title": "Calibrate",
      "program": "dmc/DMC.exe",
      "arguments": [
        {
          "mode": "calibrate",
          "model": "sacramento",
          "precipitation": "data/$V1/rainfall.csv",
          "observed_runoff": "data/$V1/runoff.csv",
          "pet": "data/$V1/pet.csv",
          "output": "data/$V1/results"
        }
      ]
    },
    {
      "id": 2,
      "dependant_ids": [1],
      "title": "Hydrograph",
      "program": "python ut/plot-ts.py",
      "arguments": [
        {
          "input_1": "data/$V1/results/runoff.csv",
          "input_2": "data/$V1/runoff.csv",
          "output": "data/$V1/results/hydrograph.png"
        }
      ]
    }
  ]
}
```

Process Definition

```
for catchment in ["Biggara", "Dohertys", "Gingera"]:
    cmd = "python WFM.py --workflow=steps.json --V1=" + V1

```

Figure 4: Example set of workflow files illustrating the cases where a hydrologist wants to calibrate a rainfall-runoff model and generate a hydrograph for three catchments.

5. CONSIDERATIONS

One benefit of the solution design was that components could be implemented using the development methodology, programming language, and libraries which were best suited for the purpose of the component and the skills of the particular developer; effectively isolating specific requirements to specific system components and addressing a number of the identified system constraints. Implementation on each component could proceed in isolation from the larger system and the developer remain confident that the component could be launched as required at runtime as long as it conformed to the specified system interface

and standardised data file format. Automated testing of each component against its expected behaviour further increased developer confidence and productivity. Additionally, identifying which component was the cause of a problem during a large workflow run was simplified because all inputs were mutable files and each component wrote a log, taken together it became possible to isolate and reproduce any individual run of a component.

An additional benefit of the solution design is that it turns out to be trivial to parallelise the workload across multiple CPU cores through the WFM due to the considerations made for concurrency such as; immutable data, asynchronous task scheduling, and declarative workflows. Additionally, because system-calls are used for control flow, all the low-level task scheduling is handled by the operating system and may be assumed to be operating efficiently, especially if each task is relatively long lived and the computer has sufficient RAM. Parallelising the workload significantly decreased the time required to complete large workflows.

Although the solution design afforded many benefits it came at the expense of four main overheads:

- Using operating system files for data transfer between components resulted in a large number of IO operations for large workflows.
- Using system-calls for control requires a new operating system processes for each component execution. This was particularly problematic for UT written in Python due to the interpreter warm-up time.
- Deployment was made more difficult because of the multiple programming languages and libraries used, each requiring additional dependencies to be installed before the DMS could be used. This was particularly aggravated due to target computers being located in a managed desktop environment and incompatibilities between versions of Python and its libraries.
- Using CSV files for all data transfer and storage made interpretation of results difficult due to the large number of files required to inspect before an opinion could be formed. This was somewhat offset by the ensemble plots generated by the UT.

The computational overheads were found to be negligible because each component completes a fairly large slice of the problem and in most cases the time spent within the processing phase significantly exceeded the time spent in file IO and process launching. This is particularly true for the DMC where the time required for each run is in the order of minutes. These overheads may be a limiting factor on performance in workflows with a large number of discrete UT tasks but the ability to easily scale across multiple CPU cores may mitigate the impact. The overheads associated with deployment and interpretation were addressed through resource and project management.

6. APPLICATION

Two main workflows were created and ran using the system; the historical modelling workflow and hindcasting workflow (Tuteja et. al, 2011). Historical modelling used observation data as forcing to calibrate and validate the rainfall-runoff model with a split-sample methodology. The best five calibrated parameter sets from historical modelling were then used in the hindcasting workflow to simulate past streamflow using rainfall as forcing from the hindcast dataset of the POAMA seasonal global climate model to generate hindcast streamflow ensembles (Alves et al., 2003).

The Step Declarations for both workflows were complex, involving the execution of the DMC and multiple UT to generate timeseries, diagnostic plots, statistics, metadata, and log files. The workflow Process Definitions ran these Step Declarations over all combinations of a large number of catchments, rainfall-runoff models, numerical optimisers, objective functions, accumulation schemes, and alternative parameterisation approaches. For eight experimental catchments the system generated approximately 60,000 files in total for the two workflows occupying around 15GB in total, the run-time was in the order of 96 hours on a Intel Pentium Core 2 Duo 2.8 GHz with 3GB of RAM. Generating results for so many combinations of simulation scenarios without a workflow-driven automated system such as the DMS would have been very difficult.

7. CONCLUSION

A phased process of requirements elicitation, solution design, implementation, and deployment was performed to create a software system capable of generating the outputs required to determine if a dynamic modelling approach to seasonal streamflow forecasting could add value to the current Seasonal Forecasting Service of the Australian Bureau of Meteorology.

A solution was designed and subsequently implemented which satisfied the identified requirements and the key issues constraining the possible designs. The solution was based around a loosely-coupled component

architecture driven by a workflow based approach. Operating system-calls were used for system control and CSV files used for both data transfer and storage. Each component was able to leverage technologies ideally suited to its problem domain and undesirable requirements could consequently be confined to individual components rather than impacting the entire system.

The solution design enhanced both the productivity of developers implementing the system and hydrologists operating it by clearly separating and addressing the concerns of each. Computational overheads introduced by the architecture were found to be negligible in the context of long modelling runs and easily offset by the benefits gained.

Results generated by the DMS suggest that a dynamic modelling approach to seasonal streamflow forecasting could add significant value to the current SSF service and that the adopted solution design and implementation were appropriate for the identified system requirements. The DMS effectively transformed the limiting factor of the evaluation project from modelling and simulation to analysis and interpretation; an ideal outcome from the point of view of hydrologists. It will transition into an operational system following the identification of additional requirements and subsequent system modifications to address them.

ACKNOWLEDGMENTS

The DMS development is supported by the Water Information Research and Development Alliance (WIRADA) between the Bureau and the CSIRO Water for a Healthy Country Flagship Program. The authors acknowledge the support from Dasarath Jayasuriya, Neil Plummer, and Rob Vertessy in the Climate and Water Division of the Bureau, Enli Wang, Hongxing Zheng, Q. J. Wang, and David Robertson in CSIRO, and the BATEA team at The University of Newcastle and The University of Adelaide.

REFERENCES

Alves, O., G. Wang, A. Zhong, N. Smith, F. Tseitkin, G. Warren, A. Schiller, S. Godfrey and G. Meyers, (2003). POAMA: Bureau of Meteorology operational coupled model seasonal forecast system. National Drought Forum, Brisbane, 15 – 16 April 2003, Forum Proceedings, 49-56.

Gamma, E., Helm, R., Johnson, R., Vlissides, J., (1995). *Design patterns: Elements of reusable object-oriented software*. Reading, Mass: Addison-Wesley

Laugesen R., Shin, D., Tuteja N.K. (2009). Pilot Seasonal Dynamic Modelling System Specification and Proposed Solution, Water Forecasting Branch, Bureau of Meteorology.

Oliphant, T.E. (2007). Python for scientific computing. *Computing in Science & Engineering*, 9:10–20.

O'Toole, J.M., and Laugesen, R. (2011). *Developing Specialist Language Styles: Research and Application*. Seven Hills, NSW: Boraga Academic, ISBN: 978-1-74130-951-5

Plummer, N., Tuteja, N.K., Wang, Q.J., Wang, E., Robertson, D., Zhou, S., (2009). A Seasonal Water Availability Prediction Service: Opportunities and Challenges. 18th World IMACS / MODSIM Congress, Cairns, 13-17.

Rahman, J.M., Seaton, S.P., Perraud, J-M, Hotham, H., Verrelli, D.I., Coleman, J.R. (2003). It's TIME for a New Environmental Modelling Framework. Proceedings of MODSIM 2004 International Congress on Modelling and Simulation, Townsville, Australia, 14-17 July 2003. Modelling and Simulation Society of Australia and New Zealand Inc., Vol 4, pp 1727-1732

Shin, D., Schepen A, Peatey T., Zhou S., MacDonald A., Chia T., Perkins J., and Plummer N. (2011). WAFARi: A new modelling system for seasonal streamflow forecasting service of the Bureau of Meteorology, Australia, MODSIM 2011, Perth.

Tuteja N.K. and Extended Hydrological Prediction Team (2009). Development and Evaluation of Seasonal Prediction Capability using Dynamic Hydrologic Modelling Approach, Water Forecasting Branch, Bureau of Meteorology.

Tuteja N.K. and Extended Hydrological Prediction Team (2011). Experimental Evaluation of the Dynamic Seasonal Streamflow Forecasting Approach, Water Forecasting Branch, Bureau of Meteorology.

Wang, Q.J., and Robertson D.E. (2011). Multisite probabilistic forecasting of seasonal flows for streams with zero value occurrences. *Water Resources Research*, 47, W02546, doi:10.1029/2010WR009333.