Custom functionality and integrative approaches for hydrological modelling tools for water resources planning and management

Shaun S. H. Kim, Dushmanta Dutta, Chas A. Egan, Juernjakob Dugge, Ramneek Singh, Geoff P. Davis and Joel M. Rahman

ABSTRACT

This paper outlines the application and usefulness of a software platform that enables hydrologists to develop custom functionality in a new hydrological modelling tool, eWater Source, designed for water resources planning and management. The flexible architecture of the software allows incorporation of third-party components as plug-ins to add new capabilities that are not built in. Plug-ins can be developed to adapt the software to suit the needs of hydrologists with modest software development knowledge. This can result in an improvement in workflow and efficiencies. In addition, modellers can use plug-ins to integrate hydrological process and management models that may not be able to be built in the normal tool. The paper introduces the plug-ins functionality of the modelling tool, its design and applications with three example plug-ins to demonstrate. These are: (1) a data processing plug-in to upscale urban environment models; (2) a management rule plug-in to calculate loss allowances for the Pioneer Valley; and (3) a model plug-in to integrate into a river system model. For planning purposes, the use of plug-ins is thought to be critical for modelling management rules for various jurisdictions since these can vary significantly between jurisdictions and change over time.

Key words | custom functionality, eWater Source, hydrological modelling, plug-ins, river modelling, water resources management

Shaun S. H. Kim (corresponding author)
Dushmanta Dutta
Ramneek Singh
CSIRO Water for a Healthy Country National Research Flagship,
CSIRO Land and Water,
Clunies Ross Street,
Acton,
ACT 2601,
Australia
E-mail: *shaun.kim@csiro.au*

Chas A. Egan
Juernjakob Dugge
Geoff P. Davis
Joel M. Rahman
eWater, University of Canberra Innovation Centre,
Building 22,
University Drive South,
Bruce,
ACT 2617,
Australia

ABBREVIATIONS AND NOTATION

a	High security class A
b	High security class B
c1	Clipping threshold
c2	Days at end of month
CRC	Cooperative Research Centre
D	Diverted volume
DLL	Dynamic-link library
DNRW	Department of Natural Resources and Water
DSE	Department of Sustainability and Environment
GIS	Geographic information system
GR4J	Génie Rural à 4 paramètres Journalier
H	High class priority water allocations
IQQM	Integrated Quantity and Quality Model
kNN	*k*-Nearest Neighbour
L_n	Index percentage announced allocation for High Class B Pioneer
m	Index month
MDB	Murray–Darling Basin
MDBC	Murray–Darling Basin Commission
MI	Model integration
ML	Megalitre
MSM	Monthly Simulation Model
nsp	Non-stream flow period
P	Set of data points for the transmission operational allowance
R	Minimum flow rate
REALM	Resource Allocation Model
RR	Rainfall–runoff

doi: 10.2166/hydro.2014.125

S	Active storage
sp	Stream flow period
T	Transmission operational allowance
TIME	The Invisible Modelling Environment
UI	User Interface
VU	Victoria University
x_1	Maximum capacity of the production store
x_2	Groundwater exchange coefficient
x_3	Maximum capacity of the routing store
x_4	Base lag of the unit hydrograph

INTRODUCTION

In hydrological modelling, there is no single approach or algorithm that makes the most sense in all circumstances. Differing data sources across catchments limit the applicability of some approaches. Different solutions are needed to answer questions at very fine spatial and temporal scales as opposed to broader scales (Wurbs 1994; Wagener *et al.* 2010). Urban environments have different characteristics to rural environments, which themselves have very different hydrological characteristics depending on the amount of in-stream and catchment development. In water resources management (and particularly in Australia), the policy and operational rules governing the running of the system can be both complex and highly specific to a location. Water resource management has evolved over time to include more complex rules for allocation in attempts to adjust for increasing environmental and socio-economic challenges. In addition, Australia's river management, and therefore much of the hydrological modelling, was managed in jurisdictions, and so modelling tools were developed for specific regions. This caused an establishment of modelling approaches that were unique to the different regions. More recently, there has been a strong emphasis on integrated modelling using a holistic approach, both in modelling transboundary river systems where different parts of the basins fall under different political jurisdictions, and also in incorporating multiple processes and multiple management rules for defined regions (Welsh *et al.* 2013; Dutta *et al.* 2014). In this paper, a model shall be referred to as a generic conceptual representation of an environment or process that does not necessarily contain any site-specific data. Models take the form of equations, algorithms and/or computer code that may have a user interface (UI). The term framework is used to describe the underlying systems upon which models and other software products (applications) are developed.

In the 1970s and the 1980s, site-specific river basin models were developed by engineers for water management organisations (Zagona *et al.* 2001). Different organisations used different approaches, hardware and software technologies to develop models suitable to their needs at that time. These models, with limited representation of hydrological processes, could not be easily modified and were not flexible enough to apply in other sites and to incorporate new components for changing modelling needs. For example, within the Murray–Darling Basin (MDB) of Australia, different models have been used by different jurisdictions. The Queensland and New South Wales state governments use the Integrated Quantity and Quality Model (IQQM) (Simons *et al.* 1996; Podger 2004; Podger & Beecham 2004) at daily time scale. The Victorian state government uses the Resource Allocation Model (REALM) (Diment 1991; Perera *et al.* 2005) at monthly scale. The Murray–Darling Basin Authority uses the Monthly Simulation Model (MSM) with BigMod (Close 1996; MDBC 2002) to model river flow for the Murray. These models are hard to combine due to different modelling concepts and temporal and spatial resolutions.

The eWater Cooperative Research Centre (CRC) in partnership with several governmental and non-governmental organisations within Australia has recently developed an integrated modelling software called the eWater Source (shortened to Source from here on) that is designed to provide a common catchment and river modelling platform across Australia (Dutta *et al.* 2012, 2013; Welsh *et al.* 2013). Source has inherited many aspects from existing models, in particular, the IQQM, MSM-BigMod and REALM and it has addressed extensibility limitations making it a flexible platform for expanding uses.

The modelling tools that are mentioned above generally fall short in one aspect – since the models were hard-coded it was difficult to adapt new models of hydrological processes or incorporate complicated and changing management rules. Generally provided are 'rules engines'

that give some flexibility in defining flow, constraints and management decisions (Hameed & O'Neill 2005; VU & DSE 2005a); however, one would have to be creative with the available components of the software and create dummy systems in order to represent complicated resource assessment and sharing systems (e.g., VU & DSE 2005b). Wurbs (1994) observed that engineers and scientists naturally prefer the flexibility provided in programs they have developed themselves, however it was not practical for each water management agency to develop flexible in-house models. Therefore, in Australian water resources modelling, the advancement of hydrological models was limited by modellers' software development capabilities or the resources available to acquire the help of software professionals (Welsh *et al.* 2013).

Best practice modelling dictates that some level of complexity is required in hydrological modelling to obtain results which can be comparable to observed data (Black *et al.* 2011). To achieve this, the models need to incorporate the dominant processes that impact the water balance in a system. Where interactions between domain-specific models (e.g., MODFLOW (McDonald & Harbaugh 1988), REALM, IQQM, MSM-Bigmod, WATHNET (Kuczera 1997) and rainfall–runoff (RR) models are important, a manual approach called 'loose coupling' is often adopted for integrating the results from different domains (Cuddy & Fitch 2010; Gijsbers 2010; Yang & Podger 2010). Loose coupling in the context of this paper refers to completing a particular model run and then once complete, using the results as input for another model. The term integration is used to describe any interaction between different applications, models, processes and/or interfaces. Source providing flexibility for users to more easily implement integrative approaches is arguably a way the river system modelling community can improve the predictive capabilities of their existing models. However, this would also give users the responsibility to apply best practice modelling techniques since the flexibility allows users to create more complex models than the available data justifies (Silberstein 2006).

In the context of spatially focused environmental management, geographic information systems (GIS) were the first set of tools that provided integration functionality. They provided data management, process modelling, analysis and visualisation tools in single packages, otherwise known as software services. Now, GIS tools address problems from a range of research disciplines including atmosphere-surface systems, hydrology, forestry and biology. Argent (2004) explains that this was a successful approach to integration; however, it resulted in modelling concept inflexibility, since GIS tools are often built with specific technical and conceptual structures in mind, opposed to a generic platform for environmental modelling. Meanwhile, the integration of GIS and environmental modelling has advanced with the ability to build model functions into the GIS and vice versa using flexible scripting languages (e.g., DeVantier & Feldman 1993; Engel *et al.* 1993) and development of spatial and visualisation component-ware (e.g., Argent & Mitchell 1999; Rebolj & Sturm 1999). However, GIS are yet to integrate many well-known hydrological models, for example, MODFLOW.

From a software development point of view, there is a huge amount of flexibility available to the developers and modellers to build specialised software applications. Generally, the flexibility is available via the software development frameworks, where library and service components of the frameworks are used by the developers to build an application. From these building blocks, a conceptual modelling structure can be built.

The principles of component-based software development also apply to modelling applications; however, there are some important attributes of the modelling discipline that make it unique to others. First, process models (e.g., RR, routing, groundwater–surface water interactions) are the major components of the application. The process models are combined to form the conceptual structure of applications and, depending on the flexibility of the applications, models can be adjusted, deleted, added and/or replaced. Second, the users of such applications include scientists, who are always exploring and inventing new models. As such, it is desirable for applications used for modelling to be flexible, and have a wide range of available models, functions and tools for visualisation and analysis (Argent 2004).

Some basic applications may allow access to the software framework's components but do not declare a highly defined conceptual structure, such as VisualTIME (Stenson *et al.* 2011). Similarly, applications such as MATLAB

(MathWorks 2013) and R (R Core Team 2013) provide code-based modelling interfaces that have large and diverse ranges of tools within toolboxes and packages available for modelling, visualisation and analysis. These examples provide a large degree of flexibility, particularly with creating or adjusting conceptual structures of models. However, it is undesirable to develop a model from scratch if the conceptual structure replicates something already developed in the past. Conversely, highly developed software applications are completely inflexible in their conceptual structure, but usually provide more usability in their interface (e.g., MODFLOW, REALM, IQQM, MSM-Bigmod, WATHNET).

Generally, it is observed that flexibility decreases as the definition of conceptual structure increases as a software becomes more developed. Flexibility in these software applications comes from greater functionality, in the available modelling components, analysis tools and UI, requiring more effort on the part of the software developers to keep the applications up-to-date. Often the application can be more 'current' than the software framework as new and innovative methods are manually incorporated into the application instead of updating the software framework.

There are examples of concerted efforts to develop facilitations to integrate different legacy environmental modelling applications, such as GIBSI (Mailhot *et al.* 1997; Rousseau *et al.* 1997), which combines four legacy models with a database management system and a GIS. Another example is the IRSMF (Yang 2010), which is designed to link surface water and groundwater legacy models. Both examples are specialised software applications and therefore are limited to only use their respective specified legacy models, although there are applications that are designed to offer more generic approaches. For example, RAISON treats models as separate modules, only acting as a module management system, by managing the data handling, execution, visualisation and analysis functions (Argent 2004).

There is an increasing emphasis on using integrative approaches in environmental science. Argent (2004) summarises the software development approaches and concepts for integrating environmental models. More specific to hydrology, Wagener *et al.* (2010) describes cross-disciplinary integrations as a primary characteristic of future hydrologic research. A major challenge for environmental modellers is dealing with system complexity in circumstances that arise from non-linear, heterogeneous and dynamic processes involving hydrologic, biogeochemical, ecologic and human systems, with strong interactions and feedbacks. These are not easily derived from understanding the components of the system in isolation. To align with these challenges Source needs to be highly adaptable, to be able to easily implement integrative approaches. For integrative approaches to be used, Source needs to allow extension within the simulation. The extensions themselves must be specifically developed to depend on components of the host application.

One of the key features that Source provides is a platform to incorporate new components and functionalities. The philosophy of extensibility of hydrological applications is inherited from Source's predecessor, the E2 Catchment Modelling Framework (Argent *et al.* 2005, 2009; Argent 2007; Perraud *et al.* 2005), with which Source shares much of the conceptual structure and software implementation. Source gives users a platform to be able to add and develop features as plug-ins to the application. A plug-in is a set of software components that adds specific abilities to a larger software application. Plug-ins designed by the modelling community should help alleviate the pressure on developers on keeping applications up-to-date, increase the life of applications and enhance the flexibility of highly developed software. A particularly valuable feature of using plug-in architecture is that it allows plug-ins to effectively become part of the host application. This means there should not be any additional performance overhead compared to hard-coding since it enables intra-process communication between plug-in and host application (opposed to inter-process communication). This paper aims to determine how Source's plug-in functionality could be used to improve current river system modelling methods, and demonstrate that there is a negligible performance penalty involved in using Source plug-ins.

SOFTWARE DEVELOPMENT APPROACH

There are many examples of modern software designs that use plug-in architectures. Web browsers are well-known to utilise plug-in architectures since it is not possible to know all the media types that will be relevant when the web

browsers are initially developed. For example, Macromedia's plug-in allows their Shockwave Flash animations to be displayed in popular web browsers (Chatley *et al.* 2003). The Eclipse Platform is used for building integrated development environments, and is another successful example of a system designed for long-term evolution using reusable and extensible components (Wermelinger & Yu 2008).

Source is developed using The Invisible Modelling Environment (TIME, Rahman *et al.* 2003) and includes a graphical UI, a command line interface and a web service interface (Penton *et al.* 2011). Most uses of the system are catered for by the graphical UI, including model building, model calibration and running scenarios. The command line and web services interfaces support batch runs and other styles of automation on either a local machine or remotely over a network.

Source allows users to open, create, change and save project files which are made up of one or more scenarios. The scenarios contain the configurable catchment and river system models. There are two scenario types available in Source: the River Management Schematic Scenarios for river forecasting, operations and river planning purposes, and scenarios created with the Geographic Wizard for catchment modelling. These scenarios include various components including a range of UI based tools for data analysis and import/export, catchment runoff models, ground–surface water interaction models, demand models, river regulation and storages, water quality models and a variety of resource assessment models. Even with such a comprehensive list of available models, to ensure applicability to a wide variety of hydrological modellers that covers multiple disciplines and scales, additional flexibility and customisations are required, since it is unfeasible to build functionality to meet the requirements for every user of Source. Nor is it practical to constrain users to only use Source's hard-coded functionality.

Using an object-oriented paradigm in the software architecture provides a means to substitute alternative, but in some way equivalent components, based on common interfaces. For example, a common interface may be defined for data sources that allows multiple sources of data to be used and substituted, without affecting the existing system (Argent 2004; Papajorgji 2005). The Source architecture supports user-developed models or tools to be 'plugged in' to different parts of the modelling process. Plug-ins offer flexibility without the significant computational burden of interpreters (which executes the code without prior compiling) or difficulties in modifying and maintaining the code base (eWater CRC 2011a). Source provides services for the plug-ins to use, including the Plug-in Manager which is a way for the plug-ins to register themselves with the host application, and protocols for the exchange of data. As the plug-ins effectively become components of the software at runtime, they too must be written to run on the .NET platform, although they do not need to be written in the same language that is used for the core product. Plug-ins are compiled to form dynamic-link libraries (DLLs). In the plug-in code, the user defines the location within Source that the DLL will be executed (the 'plug-in point'). Source supports plug-in points for catchment, node or link models, data preprocessing tools, and extending or customising UI behaviour (Figure 1). Plug-ins can be categorised into two basic types: UI and Model Integration (MI) plug-ins.

UI plug-ins can be used to perform pre- and post-processing functions, for example, exporting model results to custom file formats. UI plug-ins can also perform and/or automate data processing tasks which would have otherwise been performed manually as a pre- or post-processing exercise, such as reformatting RR data from the Australian Bureau of Meteorology's SILO data drill (Jeffrey *et al.* 2001).

UI Plug-ins are conceptually different from MI plug-ins, although all MI plug-ins have some UI component. MI plug-ins are models of hydrological processes or management to be incorporated into Source. The model equations are solved for discrete time steps within the simulation period, for instance, for RR or groundwater flux models.

There are two basic ways of implementing a MI plug-in. The first way is to develop a class within a code that 'inherits' from an existing model template and implements or 'overrides' the required algorithms, such as the 'run time step' procedure required by most temporal models (it is called runTimeStep() in Source). The underlying mechanism to find the custom built methods is software reflection (Perraud *et al.* 2005). At the initialisation phase, Source will scan through all loaded DLLs to find the classes that inherit from the relevant model template and make them available for selection by the end user. For example, a plug-in could inherit from the RR model library of TIME (Rahman *et al.* 2003) and

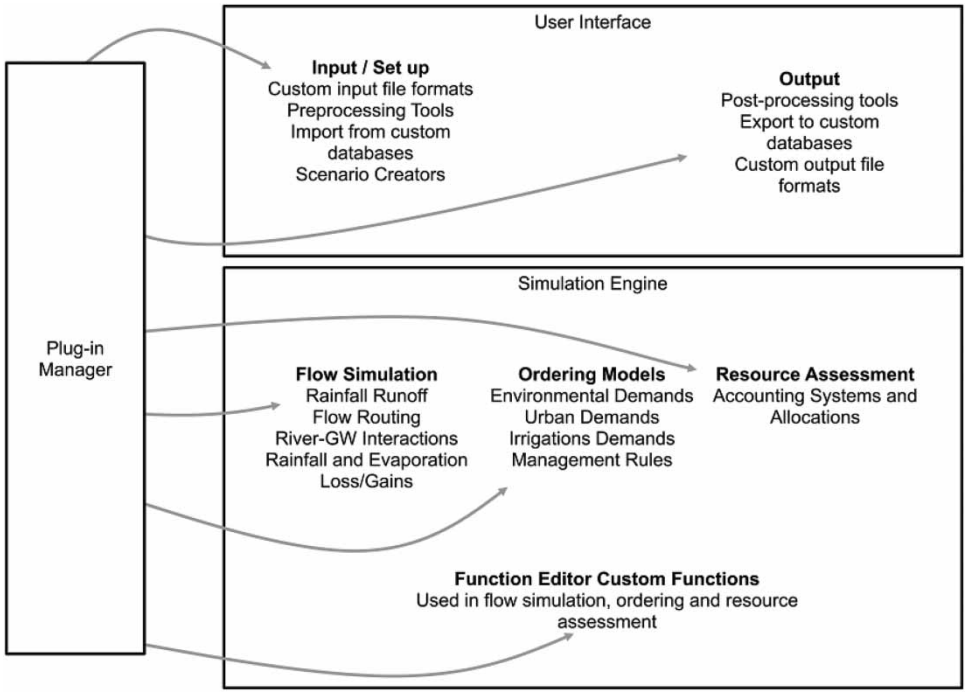

Figure 1 | Plug-in architecture of Source. The arrows show UI and MI plug-in points in Source.

then the custom RR model would appear, alongside existing models, in the list of available RR models. This method requires adding the persistence mechanism so that Source maintains the user specified parameters of the model each time the project file is created, run, saved or re-opened. Source uses an object relational broker called NHibernate (Perkins 2011) to store Source projects in a database.

The other way of implementing a MI plug-in is by creating a new function to be added to the language of the Source Function Editor. The Function Editor allows users to define a value using an expression via a range of arithmetic, comparison and logical operators, time variables and functions. This tool is embedded in many of the core model elements, typically where a management rule or some other human behaviour (such as demand) needs to be captured (Penton & Gilmore 2009). Model parameter and input values can be dynamically dependent on the values of other model parameters. Function Editor plug-ins are software components as in any other plug-in, except they are specifically accessed via the Function Editor.

Function Editor plug-ins have been termed Custom Functions and are identified by the use of the CustomFunctionContainer attribute on the containing class and also the CustomFunction attribute on the containing function (Figure 2). Users are able to use Custom Functions within the Function Editor to operate on specified data during run time (Kim *et al.* 2011, Figure 3). When there is a referral to a function, Source will search through the CustomFunction attributes that are being used to find available Custom Functions.

Dynamic linking of plug-ins to Source allows constant data exchange through intra-process communication, because the plug-in effectively becomes part of the host application. One-way exchange of data from the plug-in to the Source run time is the simplest form of MI. These types of algorithms can be run independently of Source and give the same results since no data are required from the simulation. In these cases, loose coupling may be preferred since there is little benefit to the predictive performance in fully integrating these types of models when the alternative will give the same results. On the other hand, loose coupling inhibits the ability to perform repeated sampling for things such as parameter optimisation. Two-way data exchange means that some feedback is required by the plug-in and the plug-in has a dynamic relationship with Source (Figure 4). To illustrate, a RR

```
using System;
using RiverSystem.TaskDefinitions;

namespace TestPlugin
{
    [CustomFunctionContainer]
    public class TestClass
    {
        [CustomFunction("testPlugin")]
        public static double TestFunction( double x1, double x2, double x3, double x4 )
        {
            // Start of user's code ////////////////
            ...
            ...
            ...
            // End of user's code //////////////////
            return x;
        }
    }
}
```

Figure 2 | Sample code required for writing a Custom Function plug-in. This displays the use of the CustomFunctionContainer attribute on the containing class and also the CustomFunction attribute on the containing function.

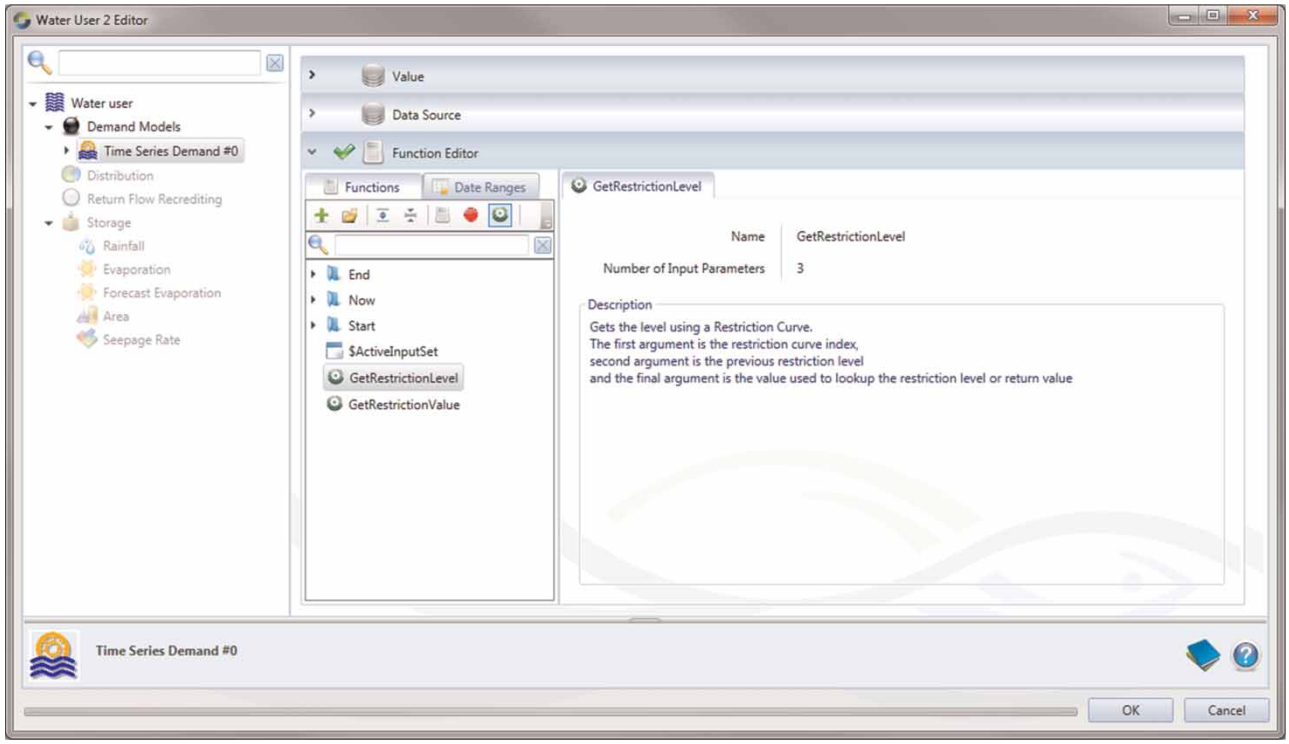

Figure 3 | Screenshot of the Function Editor showing the loaded Custom Function plug-ins 'GetRestrictionLevel' and 'GetRestrictionValue'.

model most typically has a one-way linkage to a river systems model. The RR model is respondent to climate, but not to any of the variables calculated within the river model. By comparison, a flow routing model operating within a single reach of the river model has, as one of its inputs, the incoming flow from upstream, thus it has a

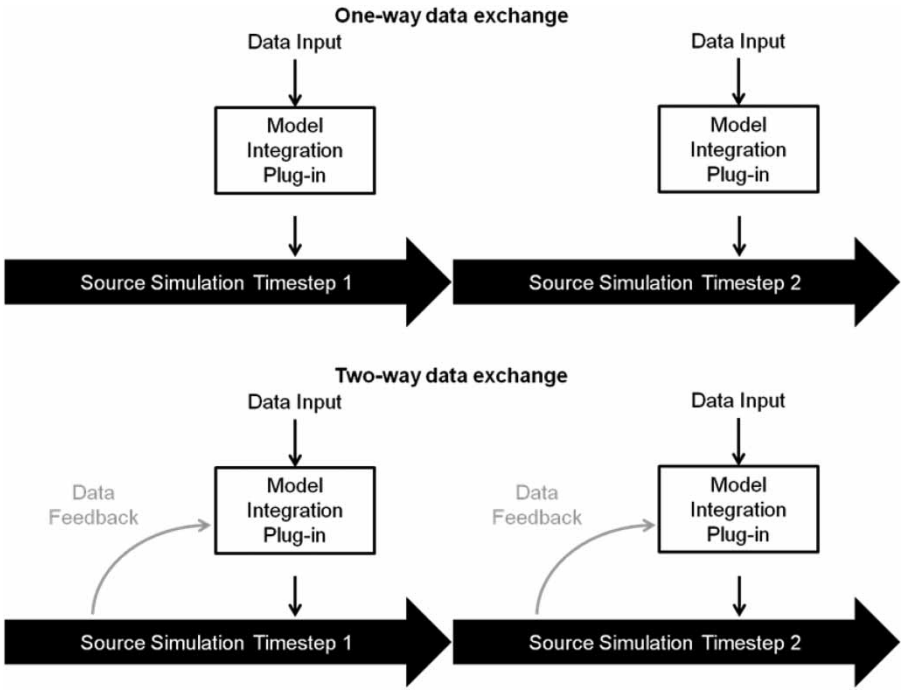

Figure 4 | Data flow during the time step run for one-way and two-way exchange of data between Source and the plug-in. A two-way exchange plug-in accesses Data Feedback that is calculated during the simulation but is only collected at the time of execution of the plug-in. Note that Source only computes results for discrete time steps.

two-way data exchange with the rest of the river systems model. Finally, a demand model might be implemented as either a one-way or a two-way data exchange. If the demand model is independent of the operations of the river, in that it is not subject to demand restrictions or delivery shortfalls, then the demand model will produce the same result regardless of what happens elsewhere in the river systems model and could be implemented with one-way data exchange, perhaps as a loosely coupled component. If, on the other hand, the predicted demand is responsive to water availability in the overall system, such as areas of crop planting being dependent upon water allocation decisions, then the demand model needs a two-way data exchange with the rest of the model.

EXAMPLE PLUG-INS

The following examples are given to demonstrate some potential uses of Source's plug-in functionality. The examples have been developed during trial demonstrations and testing of Source. The first example is for a data processing UI plug-in which is an important display of how plug-ins can be used to import data of a different scale, process it and run it in a Source simulation. It uses output from eWater Urban Developer (shortened to Urban Developer from here on) which is a modelling platform specifically designed to model the urban water cycle (eWater CRC 2011b). The second example is of a MI plug-in used as part of the resource assessment system for the Pioneer Valley, Queensland. This plug-in was developed as part of Queensland Department of Environment and Resource Management's Source trial demonstration to exhibit the extensibility of Source's resource assessment and Function Editor functionalities (eWater CRC 2011c). The final example was designed to exhibit the alternative to loosely coupling RR and river system models. This was achieved using the Function Editor and MI plug-ins using Custom Functions.

UI plug-in for upscaling Urban Developer to Source

In this practical use of UI plug-ins given by Dugge *et al.* (2012a), there was a requirement to use the daily model

results of an Urban Developer application (eWater CRC 2011b) to upscale the model to a larger time step in Source, generating surrogate models that are faster to run but that recreate important characteristics of the original model output. A sampling scheme method was employed to select appropriate aggregated monthly values for use in a regional model (as recommended by Coombes & Kuczera (2003) and Kuczera (2008)). The model extension was implemented in a plug-in. The plug-in takes the daily output of an allotment or cluster scale model and preprocesses the data for use in a water demand node in Source that implements the original model presented in Kuczera (2008).

The *k*-Nearest Neighbour (kNN) Upscaling Preprocessor plug-in uses Urban Developer results. The Source UI for the plug-in is shown in Figure 5. The plug-in allows users to load Rain and Tank Yield files that will be plotted against each other. The 'Clipping Threshold (c1)' and the 'Days at End of Month (c2)' parameter can be adjusted and the plug-in will calculate the corrected monthly rainfall time series for the given parameter set and plot the tank yield time series against this corrected rainfall time series. The correlation coefficient is shown in the lower right of the plot area. By clicking 'Auto-Calibrate', a steepest ascent hill climbing algorithm is started that searches for the set of parameters that maximise the correlation coefficient (Dugge *et al.* 2012b).

After a suitable parameter set has been found, the corrected rainfall and the tank yield time series can be exported into a daily time step or monthly time step to incorporate in large system modelling in Source. The correction with the same parameter set can be applied to one or more other rainfall time series.

UI plug-ins have the potential to save time and effort for tasks such as data assimilation and processing. These types of plug-ins would be most desirable for ongoing activities that involve large data sets. This loose coupling example is a case where this functionality could be performed external to Source. However, the workflow is improved by developing a plug-in. In this case, the more the plug-in is used, the more value that will be returned for the initial effort required to build the plug-in. As a one-off exercise, there would be hardly any benefit over performing this task in external software. This particular plug-in is, however, likely to be frequently used since it provides a user-friendly means of upscaling Urban Developer models. Integrating numerical models into the simulation run rather than importing the output of externally run output also provides some time-saving capacity.

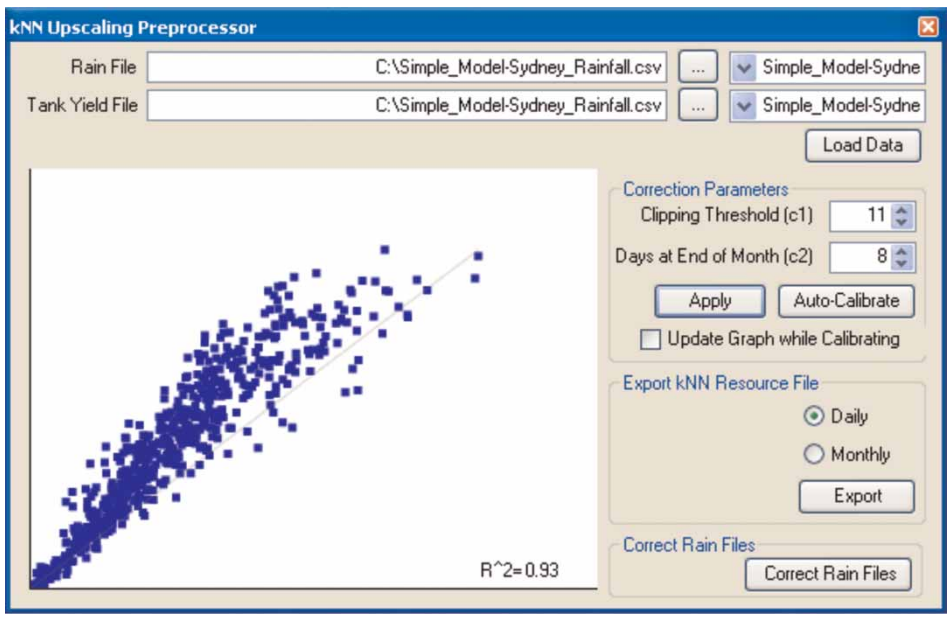

Figure 5 | Upscaling Preprocessor Source plug-in UI.

MI plug-in for water management rules for the Pioneer valley

Over the history of water management in Australia, the rules to define allocations to various water users have become complicated and vary greatly between jurisdictions. An appropriate use of the plug-in functionality is in defining water resource management rules that are complex and must be solved with an iterative procedure. Function Editor Custom Functions have been developed to calculate the announced allocations for two different water products in the Pioneer water supply systems in accordance with the sharing rules specified in the Pioneer Valley Resource Operation Plan (DNRW 2007). Allocations for water are determined based on the available surface water resource (in reservoirs) for different priority water users in the river system. High security class A water users are generally urban utilities where high security B water users are predominantly irrigators. Both the announced allocation percentage for high security class A and B are calculated at each time step. The function for the announced allocation percentages are given by

$$L_{m,a}, L_{m,b} = f(S_m, T_m, R_m, D_{m,a,sp}, D_{m,b,sp}, D_{m,a,nsp}, D_{m,b,nsp}, H_{m,a}, H_{m,b}) \quad (1)$$

where L is the announced allocation, S is the active storage, T is the transmission operational allowance, R is the minimum flow rate, D is the diverted volume, H is the high class priority water allocations. m denotes the index month, a and b denote the high security classes A and B, respectively, and sp and nsp denote the stream flow and non-stream flow period, respectively. To work out what the allocations should be for high security class A and B, there must be knowledge of the transmission losses that are expected to occur while running the system in order to adjust the allocations accordingly. The transmission operational allowance is calculated based on a predetermined piecewise linear relationship between the transmission allowance and allocation for high security class B. This is given by

$$P_m = (L_0, T_{0,m}), (L_1, T_{1,m}), \ldots, (L_n, T_{n,m}) \quad (2)$$

where P is the set of data points for the transmission operational allowance (given as Table A1 in the Appendix, available online at http://www.iwaponline.com/jh/017/125.pdf), and L_n is the index percentage announced allocation for High Class B Pioneer. This is linearly interpolated to work out the transmission operational allowance, given by

$$T_m = \text{INTERPOLATE}(P_m, L_{m,b}) \quad (3)$$

One can see that there is a circular reference problem. The assumed loss allowance, which must be known in order to calculate L_a and L_b, depends on L_b itself. Solutions for L_a and L_b use a relatively simple iterative algorithm. The result for L_b is fed back into the calculation over-and-over until the solution converges. The solution converges quickly because the result varies little with the assumed L_b. The implementation in Source is shown in Figure 6.

In this example, there is a two-way data exchange. The river system requires this integration to be able to behave like the actual system. In addition, these resource management rules cannot be modelled externally since there is an interdependence of data between the plug-in and Source at each time step. That is, data for the current and previous time steps need to be fed into the resource calculation, the announced allocation is determined, and then data is fed back into the Source simulation. This example also uses the plug-in functionality to solve a complex problem in the announced allocation calculation. There is non-linearity caused by the interdependence of the transmission and operational allowance and the announced allocation calculations. This is solved by a simple root finding solver that continuously performs the announced allocation and transmission and operational allowance calculations until the solution converges. This method would have been difficult to implement using conventional hydrological modelling tools and is a unique selling point of this approach.

MI plug-in for a RR model (GR4J)

The daily rainfall–runoff model, Génie Rural à 4 paramètres Journalier (GR4J) (Perrin *et al.* 2003), is imported as a series of Custom Functions in a plug-in into a Source Rivers scenario (Kim *et al.* 2011). The Function Editor performs the series of

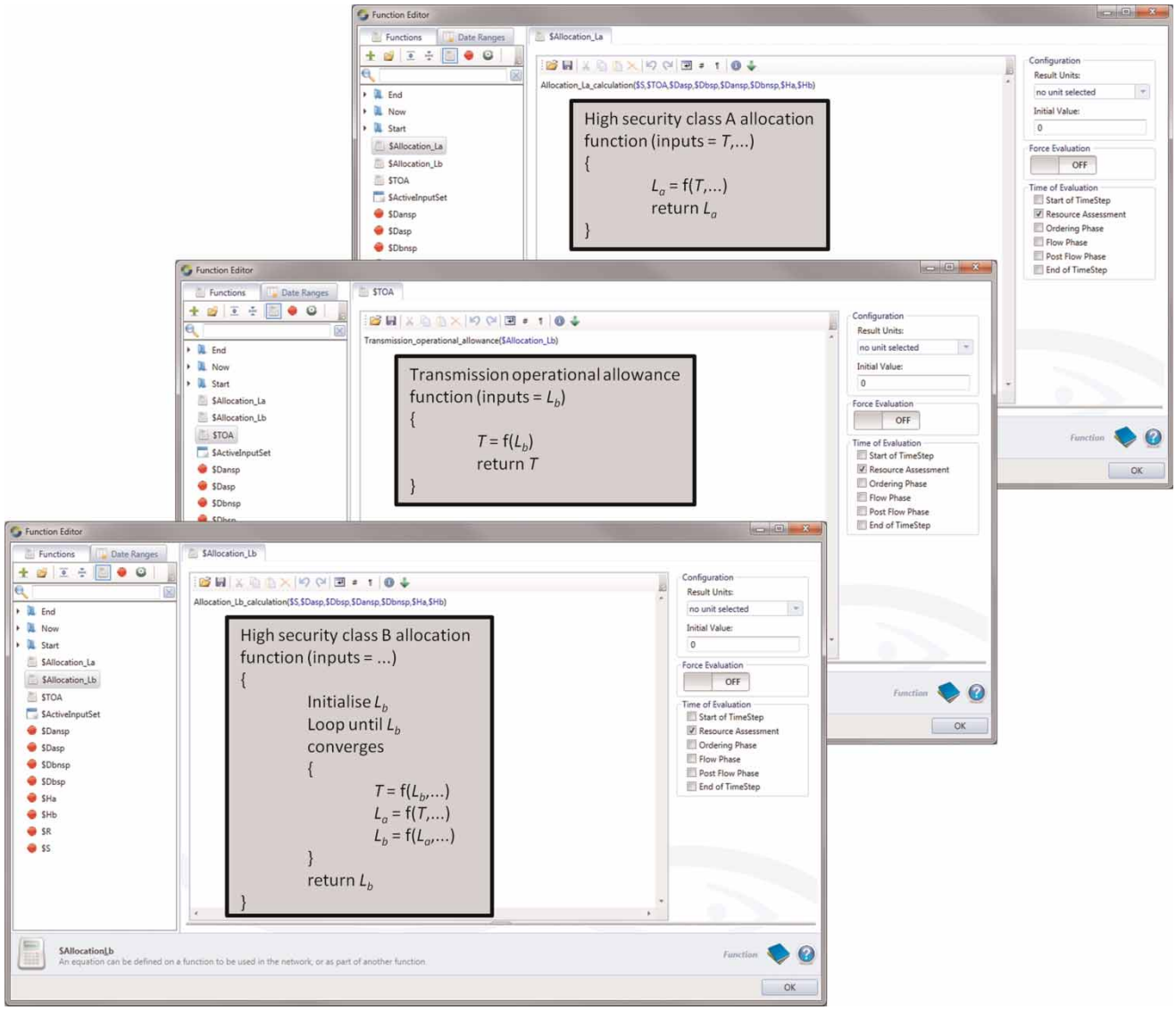

Figure 6 | Use of the Pioneer Valley Water Management plug-ins in the Function Editor. These are computed each time step.

calculations at each time step. The final expression requires 17 input variables to calculate runoff at each time step.

Four component types of the Function Editor were used in the RR model: Global Expressions, Variables, Time Series and Custom Functions. The calculations of GR4J are contained within the Custom Functions. Four input parameters, maximum capacity of the production store (x_1), groundwater exchange coefficient (x_2), maximum capacity of the routing store (x_3) and base lag of the unit hydrograph (x_4), were used for each Custom Function, which were calibrated externally. It is however, possible to calibrate the parameters using the automatic calibration wizard in Source (eWater CRC 2012d).

Eight of GR4J's variables are updated using Custom Functions at each time step. Figure 7's agreement between observed and modelled flow at a selected headwater catchment demonstrates a successful implementation of the GR4J algorithm using Function Editor and Custom Functions.

Using plug-ins to integrate models that are one-way data exchanges with Source can be an alternative to running in external software and manually importing results into

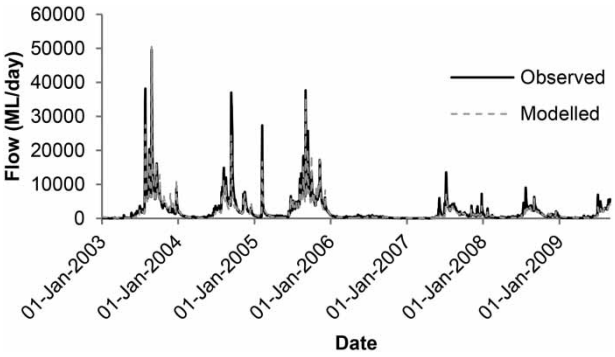

Figure 7 | Time series of GR4J RR model results compared to observed flow (Kim *et al.* 2011).

Source. Both methods would yield the same results. Other than the benefits to workflow for the user, there is seemingly little additional advantage in running a single simulation rather than two. However, the simulations have become much more reproducible, which is important for design studies. Furthermore, having a fully integrated model allows repeated sampling methods such as optimisation. The integration of the GR4J RR model into Source's Function Editor through the use of Custom Functions is a one-way data exchange example. There are many other possible ways of incorporating the GR4J results into a Source Rivers scenario, including using the output of a Source Catchments scenario. However, in this example of a one-way data exchange plug-in, automatic adjustment of GR4J parameters could be performed to observe the effect on the Source Rivers scenario. Although in this example there is the alternative of using external software to produce the same results, there are times where the integration of models are necessary to model the dynamic relationships between different processes.

Performance assessment

The time performance of simulations is a critical element for modellers. There is usually a very small performance penalty during application startup as the application has to scan for available plug-ins and load the associated DLLs into memory. The performance of running the plug-in is comparable to what it would have been if the plug-in was hard-coded into the container software, the exception being the case where the plug-in manager is doing some non-negligible operation while running the plug-ins. To investigate this, we compared the time performance between two methods for integrating a simple model: (1) using a custom function plug-in and (2) hard-coding. Results showed that there is no significant performance penalty for running the plug-ins compared to hard-coding in the application. A simulation that had 183 calls to plug-in code took a similar time to run the same simulation with the hard-coded method (Table 1). The reason for the similar timing is that once a plug-in is loaded into memory, it becomes part of the container software's process and the communication between plug-in and container becomes intra-process communication.

DISCUSSION

Software frameworks will have components to provide some flexibility to the user given that the user is using the software framework directly or the application used allows access to the components. In more specialised applications conceptual structures are more evolved and harder to change, and so it is more difficult to provide flexibility in these more developed applications. However, in these applications there are still options in providing components that can be accessed from the software framework or elsewhere. Plug-ins are able to enhance the flexibility of the applications that may have deep-seated conceptual structures.

From a flexibility perspective, it can be seen why many modellers prefer using software frameworks directly, such as TIME, or using generic modelling applications, such as MATLAB or R. Modellers are able to develop their own conceptual models, using a flexible scripting language. Modellers also appreciate the availability of flexible scripting languages in more specific applications, such as GIS.

Table 1 | Performance results for custom function plug-ins and hard-coding on an Intel Xeon CPU X5650 (2.67 GHz) with 24 GB RAM

Implementation	Number of calls	Total time (min:sec:millisec)
Custom Function plug-in	183	01:02:22
Function hard-coded	183	01:02:15

Source has been developed with flexibility in mind with plug-in infrastructure to support extensibility in the UI and available models. Some benefit can clearly be seen through the use of integrating plug-ins into hydrological modelling tools such as Source, since it gives the freedom of scripting languages, while still making use of the existing structure and UI. However, does the improvement in modelling tools mean that there will be an improvement to the results of the models themselves? The discipline of hydrology is broad and hydrologists have widely varied backgrounds. Increased flexibility in modelling software is generally desired since it increases usability but also widens the spectrum of potential users for a single piece of software, without users needing the software development background to build functionality themselves. Allowing users to draw from their own expertise would theoretically reduce the probability of human error since users could use modelling methods that have already been validated. On the other hand, it gives the opportunity for users to introduce untested methods and to potentially produce inappropriate results, ranging from subtly wrong to nonsensical. Therefore, this customisation capability gives greater responsibility to the users of the software to adhere to best modelling practices. It is obvious that the usefulness of incorporating plug-ins in hydrological modelling tools will depend on whether they will save the user time and energy, and whether they will improve the predictive performance of the model. In this way, the plug-ins may contribute to increase in demand and enhance the life-span of the tools.

The performance assessment showed that the penalty of using plug-ins is insignificant. There should not be any reduction of simulation performance apart for the extra time required to run the plug-in code itself, since cross-DLL communications should not be significant at the simulation stage of the application's execution.

CONCLUSION

The paper has introduced the plug-ins functionality of eWater Source, a newly developed integrated hydrological modelling software, and demonstrated through several examples the capability of this functionality in enhancing the performance of the software. There are three main ways the plug-in functionality of Source could improve on current hydrological modelling: workflow efficiency can be improved during data processing such as in the preprocessing plug-in for upscaling Urban Developer to Source; the predictive performance of the models can be increased by integrating significant processes in an oversimplified base model such as in the plug-in for Water Management Rules for the Pioneer Valley; and it allows users to impart their own knowledge base which is easier than adopting new concepts, but also could potentially allow integration over multiple disciplines resulting in connecting the wider modelling and research community. Some workflow streamlining can be achieved in integrating models using the plug-in functionality but it is only necessary when the introduced models require feedback. Users should only use this functionality if there are not any hard-coded or manual alternatives that could be achieved in less time. Creating more complex models than what is justified by the available data should be avoided in accordance with best practice modelling guidelines. Basic performance benchmarking showed that using plug-ins had insignificant performance penalties over hard-coding.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the members of the Source team, specifically David Penton of CSIRO for his contributions to the Function Editor functionality. We thank Matthew Stenson and Jie Chen of CSIRO for their constructive reviews of the manuscript.

REFERENCES

Argent, R. M. 2004 An overview of model integration for environmental applications – components, frameworks and semantics. *Environ. Modell. Softw.* **19**, 219–234.

Argent, R. M. 2007 E2 – past, present and future. In: *MODSIM 2007 International Congress on Modelling and Simulation* (L. Oxley & D. Kulasiri, eds). Modelling and Simulation Society of Australia and New Zealand, Christchurch, New Zealand, pp. 860–866.

Argent, R. M. & Mitchell, V. G. 1999 Putting GIS to work in catchment management: a case study from Port Phillip Bay.

Proceedings of Water 99 Joint Congress, Brisbane, The Institution of Engineers, Australia, pp. 1041–1046.

Argent, R. M., Grayson, R. B., Podger, G. M., Rahman, J. M., Seaton, S. P. & Perraud, J.-M. 2005 E2 – A flexible framework for catchment modelling. In: *MODSIM 2005 International Congress on Modelling and Simulation* (A. Zerger & R. M. Argent, eds). Modelling and Simulation Society of Australia and New Zealand, Melbourne, Victoria, Australia, pp. 594–600.

Argent, R. M., Perraud, J.-M., Rahman, J. M., Grayson, R. B. & Podger, G. M. 2009 A new approach to water quality modelling and environmental decision support systems. *Environ. Modell. Softw.* **24**, 809–818.

Black, D., Wallbrink, P., Jordan, P., Water, D., Carrol, C. & Blackmore, J. 2011 *Guidelines for Water Management Modelling: Towards Best-Practice Model Application*. eWater Cooperative Research Centre, Canberra, Australia.

Chatley, R., Eisenbach, S. & Magee, J. 2003 *Painless Plugins*. Department of Computing, Imperial College London. http://chatley.com/articles/pp.pdf (Technical Report retrieved May 2013).

Close, A. F. 1996 The BIGMOD Model of Flow in the River Murray. MDBC Technical Report 96/12. Murray-Darling Basin Commission, Canberra, Australia.

Coombes, P. J. & Kuczera, G. 2003 Analysis of the Performance of Rainwater Tanks in Australian Capital Cities. In 28th International Hydrology and Water Resources Symposium, Wollongong, Australia.

Cuddy, S. M. & Fitch, P. 2010 Hydrologists workbench – a hydrological domain workflow toolkit. In: *Modelling for Environment's Sake* (D. A. Swayne, W. Yang, A. A. Voinov, A. Rizzoli & T. Filatova, eds). *Proceedings of the 5th Biennial Conference of the International Modelling and Software Society, iEMSs*, July 5–8, 2010 Ottawa, Ontario, Canada, pp. 1562–1569.

Department of Natural Resources and Water 2007 *Pioneer Valley Resource Operations Plan*. State of Queensland.

DeVantier, B. A. & Feldman, A. D. 1993 Review of GIS applications in hydrologic modeling. *J. Water Resour. Plann. Manage.* **119**, 246–261.

Diment, G. A. 1991 Wide use of a generalised headworks and resources model: REALM. *Proceedings International Hydrology and Water Resources Symposium*. Institution of Engineers, Australia, National Conference Publication, Perth, Australia **91** (22), 579–583.

Dugge, J., Pedruco, P. & Hardy, M. J. 2012a analysis of the performance of a nearest-neighbour Surrogate model for the simulation of rainwater tanks. In: *Proceedings of the Seventh International Conference on Water Sensitive Urban Design*, February 2012, Melbourne, Australia.

Dugge, J., Pedruco, P. & Hardy, M. J. 2012b A nearest-neighbour Surrogate model for the simulation of rainwater tanks. In: *Managing Resources of a Limited Planet* (R. Seppelt, A. A. Voinov, S. Lange & D. Bankamp, eds). Proceedings of the 6th Biennial Conference of the International Environmental Modelling and Software Society, iEMSs, July 1–5, 2012, Leipzig, Germany.

Dutta, D., Welsh, W. D., Vaze, J., Kim, S. S. H. & Nicholls, D. 2012 A comparative evaluation of short-term streamflow forecasting using time series analysis and rainfall-runoff models in eWater Source. *Water Resour. Manage.* **26** (15), 4397–4415.

Dutta, D., Wilson, K., Welsh, W., Nicholls, D., Kim, S. & Cetin, L. 2013 A new modelling system for river operations with a case study from the Goulburn River, Australia. *J. Environ. Manage.* **121**, 13–28.

Dutta, D., Chen, J., Penton, D., Kim, S., Sheedy, T., Korn, A. & Welsh, W. 2014 Analysing efficiency of optimised and heuristic ordering methods in river system modelling for water allocations. *Water Resour. Manage.* **28**, 1713–1732.

Engel, B. A., Srinivasan, R., Arnold, J., Rewerts, C. & Brown, S. J. 1993 Nonpoint source (NPS) pollution modeling using models integrated with geographic information systems (GIS). *Water Sci. Technol.* **28**, 685–690.

eWater Cooperative Research Centre 2011a *How to Write a Source Plugin*. Canberra, Australia.

eWater Cooperative Research Centre 2011b *Urban Developer*. Canberra, Australia.

eWater Cooperative Research Centre 2011c The Pioneer Source Rivers Trial Application – Technical Report, Canberra, Australia.

eWater Cooperative Research Centre 2011d *Source User Guide*. Canberra, Australia.

Gijsbers, P. 2010 Opportunities and limitations of DelftFEWS as a scientific workflow tool for environmental modelling. In: *Modelling for Environment's Sake* (D. A. Swayne, W. Yang, A. A. Voinov, A. Rizzoli & T. Filatova, eds). Proceedings of the 5th Biennial Conference of the International Environmental Modelling and Software Society, iEMSs, July 5–8, 2010, Ottawa, Ontario, Canada, pp. 1587–1594.

Hameed, T. & O'Neill, R. 2005 River Management Decision Modelling in IQQM. In: *MODSIM 2005 International Congress on Modelling and Simulation* (A. Zerger & R. M. Argent, eds). Modelling and Simulation Society of Australia and New Zealand, Melbourne, Victoria, Australia, pp. 1957–1962.

Jeffrey, S. J., Carter, J. O., Moodie, K. B. & Beswick, A. R. 2001 Using spatial interpolation to construct a comprehensive archive of Australian climate data. *Environ. Modell. Softw.* **16**, 309–330.

Kim, S. S. H., Dutta, D., Singh, R., Chen, J. & Welsh, W. D. 2011 Providing flexibility in GUI-based river modelling software – Using plug-ins to create Custom Functions in Source IMS. In: *Proceedings of the MODSIM 2011 International Congress on Modelling and Simulation*, December 12–16, 2011, Perth, Australia, pp. 2345–2351.

Kuczera, G. 1997 *WATHNET-Generalized Water Supply Headworks Simulation Using Network Linear Programming, Version 3*. School of Engineering, University of Newcastle, Callaghan, NSW, Australia.

Kuczera, G. 2008 Urban water supply drought security: a comparative analysis of complimentary centralised and

decentralised storage systems. In: *Proceedings of Water Down Under 2008. Engineers Australia/Causal Productions*, Adelaide, Australia, pp. 1532–1543.

Mailhot, A., Rousseau, A. N., Massicotte, S., Dupont, J., Duchemin, M. & Villeneuve, J.-P. 1997 A watershed-based system for the integrated management of surface water quality: the GIBSI system. *Water Sci. Technol.* **36**, 381–387.

MathWorks, Inc. 2013 MATLAB Primer (R2013a). http://www.mathworks.com.au/help/pdf_doc/matlab/getstart.pdf (retrieved May 2013).

McDonald, M. G. & Harbaugh, A. W. 1988 *A Modular Three-Dimensional Finite-Difference Ground-Water Flow Model. Book 6, Chapter A1*. US Government Printing Office, Washington, DC, USA.

Murray-Darling Basin Commission 2002 Setting up of MSM_BIGMOD Modelling Suite for the River Murray System. MDBC Technical Report 2002/5.

Papajorgji, P. 2005 A plug and play approach for developing environmental models. *Environ. Modell. Softw.* **20**, 1353–1357.

Penton, D. J. & Gilmore, R. 2009 Comparing software for modelling the management rules that river operators implement. In: *Proceedings of MODSIM 2009 International Congress on Modelling and Simulation*, July 13–17, 2009, Cairns, Australia, pp. 3865–3871.

Penton, D. J., Leighton, B. P., Stenson, M., Rahman, J. & Bethune, M. 2011 Linking hydrological simulation models with workflow and optimisation software. In: *Proceedings of the MODSIM 2011 International Congress on Modelling and Simulation*, December 12–16, 2011, Perth, Australia, pp. 1251–1257.

Perera, B. J. C., James, B. & Kularathna, M. D. U. 2005 Computer software tool REALM for sustainable water allocation and management. *J. Environ. Manage.* **77**, 291–300.

Perkins, B. 2011 *Working with NHibernate 3.0*. John Wiley & Sons Inc., Indianapolis, IN, USA.

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

Perrin, C., Michel, C. & Andreassian, V. 2003 Improvement of a parsimonious model for streamflow simulations. *J. Hydrol.* **279**, 275–289.

Podger, G. 2004 *IQQM Reference Manual, Software Version 7.32*. New South Wales Department of Infrastructure Planning and Natural Resources, Sydney, NSW, Australia.

Podger, G. & Beecham, R. 2004 *IQQM User Manual*. New South Wales Department of Infrastructure Planning and Natural Resources, Sydney, NSW, Australia.

R Core Team 2013 *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. http://www.R-project.org/.

Rahman, J. M., Seaton, S. P., Perraud, J.-M., Hotham, H., Verrelli, D. I. & Coelman, J. R. 2003 It's TIME for a new environmental modelling framework. In: *Proceedings of MODSIM 2003 International Congress on Modelling and Simulation*, Vol. 4, pp. 1727–1732.

Rebolj, D. & Sturm, P. J. 1999 A GIS based component-oriented integrated system for estimation, visualization and analysis of road traffic air pollution. *Environ. Model. Softw.* **14**, 531–539.

Rousseau, A. N., Mailhot, A., Massicotte, S., Tremblay, J.-F., Bolduc, P., Duchemin, M., Dupont, J., Turcotte, R. & Villeneuve, J.-P. 1997 GIBSI – a watershed-based software system for water resources management. CSAE/SCGR Conference. CSAE/SCGR, 28–30 May 1997, Sherbrooke, Que.

Silberstein, R. P. 2006 Hydrological models are so good, do we still need data? *Environ. Modell. Softw.* **21**, 1340–1352.

Simons, M., Podger, G. & Cooke, R. 1996 IQQM – a hydrologic modelling tool for water resource and salinity management. *Environ. Software* **11** (1–3), 185–192.

Stenson, M. P., Littleboy, M. & Gilfedder, M. 2011 Estimation of water and salt generation from unregulated upland catchments. *Environ. Modell. Softw.* **26**, 1268–1278.

Victoria University and Department of Sustainability and Environment, Victoria 2005a *REALM User Manual*. http://www.water.vic.gov.au (accessed 31 March 2009).

Victoria University and Department of Sustainability and Environment, Victoria 2005b *REALM Worked Examples*. http://www.water.vic.gov.au (accessed 31 March 2009), ppp. 18–20, 205–239.

Wagener, T., Sivapalan, M., Troch, P. A., McGlynn, B. L., Harman, C. J., Gupta, H. V., Kumar, P., Rao, P. S. C., Basu, N. B. & Wilson, J. S. 2010 The future of hydrology: an evolving science for a changing world. *Water Resour. Res.* **46**, pW05301.

Welsh, W., Vaze, J., Dutta, D., Rassam, D., Rahman, J. M., Jolly, I. D., Wallbrink, P., Podger, G. M., Bethune, M., Hardy, M. J., Teng, J. & Lerat, J. 2013 An integrated modelling framework for regulated river systems. *Environ. Modell. Softw.* **39**, 81–102.

Wermelinger, M. & Yu, Y. 2008 Analyzing the evolution of eclipse plugins. In: *Proceedings of the 2008 International Working Conference on Mining Software Repositories, ACM*, May 10–11 2008 Leipzig, Germany, pp. 133–136.

Wurbs, R. A. 1994 Computer Models for Water Resources Planning and Management. IWR Report 94-NDS-7, US Army Corps of Engineers, Institute for Water Resources, Alexandria, VA, USA.

Yang, A. & Podger, G. 2010 An agent based hydrological simulation system for Murray-Darling Basin, Australia. In: *Ninth International Conference on Hydroinformatics HIC 2010*, September 7–11, 2010, Tianjin, China, pp. 1478–1485.

Zagona, E. A., Fulp, T. J., Shane, R., Magee, T. & Goranflo, H. M. 2001 RiverWare: a generalized tool for complex reservoir system modeling. *J. Am. Water Resour. Ass. AWRA.* **37** (4), 913–929.

First received 11 November 2013; accepted in revised form 16 June 2014. Available online 11 July 2014