

# **MODEL INTEGRATION FOR OPERATIONAL WATER RESOURCES PLANNING AND MANAGEMENT**

Report to the  
**WATER RESEARCH COMMISSION**

by

**DJ Clark, JC Smithers (Editors)**

Bioresources Engineering

School of Engineering

University of KwaZulu-Natal

**WRC Report No. 1951/1/12**

**ISBN 978-1-4312-0393-2**

**MARCH 2013**

**Obtainable from**

Water Research Commission  
Private Bag X03  
GEZINA, 0031

[orders@wrc.org.za](mailto:orders@wrc.org.za) or download from [www.wrc.org.za](http://www.wrc.org.za)

**DISCLAIMER**

This report has been reviewed by the Water Research Commission (WRC) and approved for publication. Approval does not signify that the contents necessarily reflect the views and policies of the WRC nor does mention of trade names or commercial products constitute endorsement or recommendation for use.

# EXECUTIVE SUMMARY

## RATIONALE

Water is a critical and a scarce resource in South Africa. Hence, it is essential that water resources are managed efficiently and equitably, as required by the National Water Act (NWA, 1998) of South Africa (Act 36 of 1998).

Computer models of water resource systems are a tool for understanding and managing water resources in South Africa. There are two main drivers for the need for better water resources modelling tools in South Africa: (i) in many catchments water demand exceeds available supply, and (ii) the requirements of the NWA. In DWAF (2004) it is reported that in the year 2000 already 10 of the 19 Water Management Areas (WMAs) in South Africa were water stressed, i.e. the demand for water exceeded supply. All, except one, of these WMAs are linked by inter-catchment transfers that assist in the transfer of water from areas with adequate supply and low demand to highly developed areas with high demand (DWAF, 2004). Water resource managers require improved water resources modelling tools for planning and operations to assist them in making management decisions leading to better water allocation and improved water use efficiency. There is a growing realisation of the need for integrated water resources management, including both water quantity and quality components and incorporating environmental, social, economic and political aspects of water, and this need is reflected in the NWA. Typically models are developed for specific domains within the water resource system and integrated water resources management will require integration of the models representing specific domains in order to provide a systems perspective for water management decisions.

This WRC project was preceded by a one year consultancy with the purpose of (i) evaluating user needs with regard to modelling for water resource planning and operations, (ii) reviewing existing modelling tools for water resource planning and operational management, and (iii) making recommendations for further research and the development of modelling tools (Pott *et al.*, 2008b). Based on the requirements identified in these consultations, a set of water management tasks and decisions and the modelling requirements to assist water managers with these was created. These tasks and decisions included: water quantity (yield) determination, assessing new licence applications, water quality, impact assessment, water use efficiency, data management and storage, monitoring, meeting licences/demands, auditing and compliance, flood management, and forecasting demands and supply. The

water resource modelling requirements identified to support these tasks can be summarised as: (i) the need to adequately represent real world complexity, and (ii) the need to integrate models representing different domains within the water resource system and other system requirements.

## **OBJECTIVES AND AIMS**

The broad objective of this project was to develop and integrate modelling tools to support water resource managers by meeting some of the modelling requirements identified for water resources planning and operations. The more specific objectives of this project were to:

- Review river network models which are suitable for water resource planning and operations and to select one to be integrated with the *ACRU* agrohydrological model.
- Investigate methods for linking different domain models such as a hydrological model and a river network model, then select and implement a suitable method for integrating the hydrological model and the selected river network model.
- Further develop the *ACRU* daily time step agrohydrological model in order to realistically represent the varying hydrological responses within the terrestrial hydrological system.
- Configure and apply the integrated hydrological model and river network model for selected catchments within the Inkomati WMA.

## **REVIEW AND EVALUATION OF RIVER NETWORK MODELS**

A review of river network models was conducted for the purpose of selecting a river network model to be used in this project. An initial review based on literature and available model documentation was conducted on the following models: AWRIS, BASINS, eWater CRC Source, HEC-ReSim, MIKE BASIN, MODSIM, REALM, RIBASIM, RiverWare, WEAP and WRAP. Each model was assessed against a set of required attributes. This initial review found that the MIKE BASIN, MODSIM, RiverWare and eWater CRCs Source models all had many of the attributes of a river network model required for this project. The review concluded by recommending that further evaluation of the MIKE BASIN, MODSIM and RiverWare models before a final decision could be made to select a river network model. The eWater CRC Source model was not considered further it was still under development.

Following the literature review of river network models, the functionality offered by the MIKE BASIN, MODSIM and RiverWare models was further evaluated by configuring a hypothetical test catchment in each model to verify information contained in the literature review, and to gain experience in setting up and running the models. Each model was evaluated against a range of criteria grouped in the following categories:

- User interface,
- GIS functionality,
- Flexible configuration,
- Water allocation,
- Scenarios,
- Accounting and auditing, and
- Operational use.

The MIKE BASIN model was found to be strong on the GIS requirements but weak in the accounting and auditing functionality. It has local, South African support and is relatively easy to use. MIKE BASIN was the easiest and quickest model to configure for the test catchment. RiverWare is strong on accounting and auditing, but is weaker on the GIS requirements. RiverWare is more flexible in the way that it can be configured, but requires greater expertise. Due to the complexity of setting up RiverWare, the user support provided by the developers proved invaluable. MODSIM was weak on the GIS requirements. Some aspects of the hypothetical test catchment could not be configured within the MODSIM model in this evaluation. The lack of user support for MODSIM was its main drawback. However, it is the only model evaluated that has no cost or licence required.

Each of the models was assigned a score for each of the evaluation criteria. The averaged scores shown in Table ES1 serves as a rough overall guide as to how the models scored relative to each other, but does not account for the relative importance of the different evaluation criteria. During the course of the evaluations it was noted that the level of user support and training available for the different models was an important factor for model selection.

Although MODSIM received the highest average score, the lack of adequate user support and difficulty in configuring aspects of the evaluation river network counted against it. Based on the evaluations described in this document, it was recommended that MIKE BASIN be selected for use in the project, largely due to its ease of use, strong GIS support through ArcGIS and availability of local user support and training.

Table ES1 Summary of evaluation ratings

Evaluation Criteria	MIKE BASIN (%)	RiverWare (%)	MODSIM (%)
User Interface	100	86	100
GIS Functionality	100	33	33
Flexible Configuration	95	91	100
Water Allocation	100	67	100
Scenarios	50	50	100
Accounting and Auditing	33	100	67
Operational Use	50	50	50
Average	76	68	79

## REVIEW AND EVALUATION OF MODEL LINKAGE MECHANISMS

Integrated Water Resources Management (IWRM) requires that social, environmental and economic aspects of water are included water management decisions. For IWRM it is necessary to integrate individual specialised models representing specific aspects of water resource systems. Integration of models requires some form of linkage mechanism to enable the exchange of data or functionality between models, preferably during runtime. Linking two models in series is usually a simple matter of running the first model, converting the output format of the first model to the input format of the second model, and then running the second model. One critical drawback of the series linking approach is that feedbacks between processes in the two separate models cannot be represented. To properly integrate the *ACRU* model and a river network model it would be necessary for data and information to be exchanged between the models on a time step by time step basis. Ideally the link between the *ACRU* model and the selected river network model should not be hard-coded but suitable for application in other scenarios and applications beyond the *ACRU* – MIKE BASIN link developed in this project. There are various means of integrating models, including linking models in parallel. The following model linkage mechanisms and component modelling systems were reviewed: OpenMI, OMS, JAMS, TIME, LIQUID, ESMF, MMS, HLA and CCA. This review concluded that the OpenMI interface specification standard was the most appropriate linking mechanism for use in the project. The advantages of OpenMI are that it is generally accepted as a *de facto* standard, is strongly supported by the OpenMI Association, has been widely adopted by key research and commercial model developers, thus providing a useful set of compliant models, and has been well documented. Though use of the OpenMI interface specification standard is not a requirement for the project, OpenMI's widespread adoption would mean that this project

would not be limited to a once off link between two specific models, but would be a useful test case leading to potential further linking of models to support IWRM.

The model linkage mechanisms supported by the MIKE BASIN , MODSIM and RiverWare, models were investigated and, in particular, evaluated to determine whether these models support the OpenMI model linking interface or could be adapted to support OpenMI. The evaluation of each model consisted of two parts, a review of the literature about the linking mechanisms and personal communications with the developers, and a technical evaluation in which a simple software implementation of the linking mechanism was created to better understand the mechanism and to assess its ability to meet the criteria set for use in the project. None of the MIKE BASIN, MODSIM and RiverWare models are OpenMI compliant and therefore their suitability for adaption through the creation of OpenMI compliant wrappers was investigated. MIKE BASIN does not support an alternative linking mechanism, but does provide access to its model engine and satisfies the criteria necessary for the creation of OpenMI compliant wrappers. MODSIM does not support an alternative linking mechanism, but does provide access to its model engine, though only satisfies some, but not all, of the criteria necessary for the creation of OpenMI compliant wrappers. RiverWare provides an alternative linking mechanism through its Data Management Interface (DMI) and batch mode capabilities, and through the implementation of a custom RiverWare solution it was demonstrated that some, but not all, of the criteria for OpenMI could be satisfied. Based on this evaluation of the linkage mechanisms by which these three models could be linked to the *ACRU* model, MIKE BASIN was found to be the only model of the three that could be linked to the *ACRU* model through the development of OpenMI wrappers. In addition, support for the model by DHI South Africa appeared to be good. This conclusion confirmed the recommendation that MIKE BASIN model should be selected for use in the project.

## **ACRU MODEL DEVELOPMENT**

As part of this project some changes were made to the *ACRU* model and the design of its associated model input files to ensure that the *ACRU* model is suited for use in both water resources planning and operations modelling, and is capable of representing real world complexity. In particular, the input files required further development with respect to their use for operational modelling. Several changes were made to the ModelData and ModelConfiguration XML schemas used for *ACRU* model input in order to refine the design and include new functionality such as scenario management, the storage of state data required to hot-start the model, a means of storing dynamic data, use of forecast data and

improved linkages to external data files. These changes to the schemas required corresponding changes to be made to the .Net and Java *XmlModelFile* libraries and the *ACRU* model itself.

A considerable amount of work was done in this project to revise the initial design of the *ModelData* and *ModelConfiguration* schemas and this has resulted in a design that is not only more robust but is expected to provide the *ACRU* model with model input functionality necessary for both planning and operations modelling. The design of these schemas is expected to be stable from this point on and no substantial changes to the design are expected. Following the revision of the design for the *ModelData* and *ModelConfiguration* schemas, various changes were made to the *ACRU* model itself to be compatible with and to make full use of model input and configuration files that use these schemas and to make the model more suitable for use for water operations modelling. Restructuring the data structure used within the *ACRU* model was a bigger undertaking than initially anticipated largely due to the complexity of dynamic type variables, however, this restructuring was critical to enable the *ACRU* model to handle time series more efficiently especially with regard to state and dynamic type data variables. The Component structure was also revised to simplify it and make it compatible with the new concepts introduced into the *ACRU* model configuration file such as Hydrological Response Units (HRUs). The introduction of the concept of resources using the *RResource* class was also a step forward from a conceptual and model extensibility point of view. Further development of the *ACRU* model has taken place to implement new functionality such as scenarios, hotstarting and the storage of state data, dynamic variables and flexible spatial component configurations. A Java version of the *ModelDataAccess* was also created.

## **IMPLEMENTATION OF OPENMI FOR MODEL LINKING**

The OpenMI interface specification standard was accepted as the most appropriate linking mechanism for use in the project. The OpenMI Association supports the standard and has released the OpenMI 1.4 version of the OpenMI Standard and associated OpenMI Software Development Kit (SDK) in 2005. OpenMI 1.4 has been adopted and implemented in a range of models, many belonging to well know international developers of water resources modelling software. The OpenMI Standard is duplicated in Java and .Net versions and each of these is supported by a corresponding OpenMI Software Development Kit (SDK) containing a default implementation of the OpenMI Standard interfaces and other helper classes. The OpenMI 2.0 version of the OpenMI Standard was recently released to provide additional user requirements not met by the OpenMI 1.4 version. It was decided that in this



project the *ACRU* and MIKE BASIN models would be made OpenMI 1.4 compliant for the following reasons: (i) the OpenMI 2.0 SDK for Java is still under development and has not been released, (ii) the problems experienced with the use of the OpenMI 2.0 SDK for .Net during an initial attempt to create an OpenMI 2.0 wrapper for MIKE BASIN indicated that the supporting tools for the OpenMI 2.0 version may not be mature enough to use reliably, and (iii) all the models registered on the OpenMI Association website as being OpenMI compliant are currently only OpenMI 1.4 compliant.

There are two main approaches to making a model OpenMI compliant, which is through the use of, either an OpenMI compliant wrapper or direct implementation in a model's source code. The wrapper approach was used in this project for the following reasons: (i) access to the source code of the models is not required and no changes are made to the model, (ii) the wrapper option is easier and takes advantage of functionality already coded into the classes provided in the SDK's, and (iii) wrapping would enable both OpenMI 1.4 and OpenMI 2.0 compliant versions of the wrappers to be provided at some point in the future, without the two versions potentially conflicting with each other.

To meet OpenMI compliance a model needs to implement the *ILinkableComponent* interface of the OpenMI standard. The model wrapping tools provided by the OpenMI Association achieve this through two classes, the first class is a wrapper class for the model engine which implements the *IEngine* interface, and the second class is the linkable component which implements the *ILinkableComponent* interface by extending the *LinkableEngine* class and accesses the model engine through the first wrapper class.

An OpenMI 1.4 compliant wrapper for MIKE BASIN was successfully developed. Tests were performed to validate the OpenMI 1.4 compliant wrapper worked as expected. Data operations applied to output from a *LinkableComponent* were also successfully implemented and tested.

Though the initial implementation of the OpenMI 2.0 compliant wrapper for MIKE BASIN was successful, problems with the OpenMI 2.0 Configuration Editor prevented further development and application of this wrapper. If the OpenMI Association releases a stable version of OpenMI 2.0 SDK, it would be recommended to use the OpenMI 2.0 compliant wrapper for MIKE BASIN, as the OpenMI 2.0 Standard is an improvement to the OpenMI 1.4 Standard.

An OpenMI 1.4 Java and OpenMI 1.4 .Net compliant wrappers were also successfully developed for the *ACRU* model. Tests indicated that the wrappers were working as expected and the *ACRU* and MIKE BASIN models were successfully linked using OpenMI for a simple test catchment.

The completed OpenMI 1.4 compliant wrappers for *ACRU* and MIKE BASIN were further tested by linking a simple test catchment configuration for *ACRU* to a simple hypothetical river network configuration for MIKE BASIN. The linked *ACRU* and MIKE BASIN models were successfully run using OpenMI and the data values transferred between the models were checked to ensure that the correct values were transferred for the correct model time steps. These tests highlighted the fact that even though two models may be OpenMI compliant, it is important that users have a sound understanding of both models to ensure that they are correctly linked.

## **USE CASES FOR THE LINKED MODELS**

There are two main reasons for integrating models: (i) to gain functionality not available in an individual domain model, and (ii) to model feedbacks between the system components represented by the individual models, to better represent the system being modelled. If there are no feedbacks between the systems being represented by each individual model, then the models can be integrated using a simple series link, otherwise a model linking mechanism, such as OpenMI selected for this project, can be used to link the models in parallel. Parallel linking involves running one model for a single time step, then using the output from this model as input to another model which is run for a single time step, and this process is repeated for individual time steps until the end of the simulation time period. The models may have different time steps and the links between them may be uni-directional or bi-directional.

Linking the *ACRU* and MIKE BASIN models, means that *ACRU* can provide input data such as streamflow required by MIKE BASIN, and in turn MIKE BASIN provides functionality such as flow routing, easier and more flexible water user configurations and water allocation methods which are not available in *ACRU*. Prior to this project the *ACRU* and MIKE BASIN models have been integrated by means of simple series links, where *ACRU* is used to generate a streamflow time series which is then translated and used as input to MIKE BASIN. This approach works well if there are no feedbacks between the terrestrial hydrological system being modelled by *ACRU* and the river network system being modelled by MIKE BASIN. However, a common example of a feedback between these two systems is

irrigation demand and supply, and this particular feedback problem resulted in the developers of MIKE BASIN including an irrigation module into the MIKE BASIN model, so that the feedback was dealt with within the model.

The advantages of using OpenMI to dynamically link models are that it:

- Saves manual translation between model output and input file formats.
- Saves developing custom code to link models, as any OpenMI compliant model can be linked to any other OpenMI compliant model.
- Enable chains of models, e.g. one-to-many and many-to-one, which would be difficult in custom code which is usually one-to-one.
- Enables linking models with different spatial and temporal resolutions.
- Enables feedbacks to be modelled.

Having successfully created OpenMI compliant wrappers for *ACRU* and MIKE BASIN the next step was to define a set of use cases to demonstrate how these two models linked using OpenMI could be used for a range of modelling scenarios. Use cases were developed for: simple uni-directional streamflow links, uni-directional streamflow and groundwater links, bi-directional streamflow, irrigation requirement and irrigation supply links, and water quality links. A simple use case giving an example of the use of the integrated models for short term operational links was also created. In each use case *ACRU* component-variable to MIKE BASIN component-variable pairs are specified to assist future users in setting up model links. In some use cases there may be more than one way to configure the model links and it is up to the user to decide which is best for their particular application depending on how the individual models have been set up. Though OpenMI makes linking models easier, users will still require a thorough conceptual understanding of both models in order to link them correctly.

## **CASE STUDY**

Both the *ACRU* model and the MIKE BASIN model were configured for the Kaap River Catchment to demonstrate the use of the integrated models in a real catchment. The Kaap River is a tributary of the Crocodile River located in the Inkomati Water Management Area (WMA), in Mpumalanga, South Africa. The 8 quaternary catchments of the Kaap River Catchment were subdivided into a total of 22 smaller subcatchments, which in turn were subdivided into hydrological response units (HRUs) based on land use.

Initial verifications prompted further investigation regarding the *ACRU* configuration. The observed streamflow at flow gauging weirs X2H010, X2H024 and X2H008 were used to verify the simulated streamflow. The best simulation was obtained at Weir X2H010 with poorer verifications at weirs X2H024 and X2H008. The poorer verifications at weirs X2H024 and X2H008 were attributed to poor observed streamflow data, but changes in land use, increased abstractions and the influence of farm dams could also be contributing factors. The MIKE BASIN configuration was further updated with assistance from DHI-SA, to include water abstractions and depth/area/volume relationships for dams in the configuration.

## **DISCUSSION AND CONCLUSIONS**

The project started by identifying the modelling requirements for water resources planning and operations to meet the requirements of the National Water Act of South Africa. It was recognised that it is unlikely that any one model would be able to meet all these requirements. To meet these requirements a collection of models covering all aspects (hydrology, environmental, economic and social) of water resource systems is required, and these models need to be integrated to model real world complexity and to ensure that any important feedbacks within the system are represented. The project thus aimed to demonstrate the integration of different domain models, with the linking of a hydrological model and a river network model, as a case study, in order to meet some of the modelling requirements identified for water resources planning and operations.

The initial review of river network models resulted in a recommendation that the MIKE BASIN, MODSIM and RiverWare models be evaluated in more detail. The result of the detailed evaluation was that the MIKE BASIN model was selected for use in the project largely due to its ease of use, strong GIS support through ArcGIS and availability of local user support and training.

Further development of the *ACRU* model and its associated model input files has resulted in the model being better suited for use in both water resources planning and operations modelling and is now capable of more realistically representing real world complexity.

The review of model linkage mechanisms resulted in OpenMI being selected for the following reasons: it is a generally accepted as a *de facto* standard, is strongly supported by the OpenMI Association, has been widely adopted by key research and commercial model developers, the provision of a useful set of compliant models, and is well documented. The OpenMI model linkage framework was successfully implemented to create an OpenMI 1.4

.Net wrapper for the MIKE BASIN model and both OpenMI 1.4 Java and .Net wrappers for the *ACRU* model. An important lesson learned while setting up and testing the integrated models was that though OpenMI may make it easy to link compliant models, a detailed understanding of the models being linked is required to ensure that valid links are created without compromising the integrity of either model. To aid in the application of the integrated models a number of use cases were described with details of which variables should be linked in each model as a guide to future users of the integrated models.

To demonstrate the application of the integrated models, the models were configured and run for the Kaap River Catchment in the Inkomati WMA. The poor verifications of simulated streamflow against observed streamflow highlighted the need for more accurate data and at a finer spatial and temporal resolution, including: rainfall, streamflow, land cover, land use practices, soils, water transfers and water abstractions.

The project was successful in demonstrating the implementation of OpenMI by successfully linking the *ACRU* and MIKE BASIN models which represent two often separately modelled domains within water resource systems. The use of these linked models is expected to be a useful tool for water resources modelling for planning and operations in South Africa. This project was a test case for model integration of legacy models using OpenMI and, given the successes achieved, there is no apparent technical reason why other models representing other domains cannot also be made OpenMI compliant. In addition to the fact that the *ACRU* model can now be easily linked with MIKE BASIN, OpenMI compliance means that these models can be linked to a range of other OpenMI compliant models, many from well-known developers of software for water resources modelling.

In this project, the advantages of linking models in parallel to provide a more holistic systems view of water resources and better representation of feedbacks between components in the different domains being modelled, were demonstrated. Some potential limitations of linking models include, the requirement for expert knowledge of all models to be linked, reduction in performance in running simulations, due to the linkage mechanism, and increased uncertainty in the simulation results introduced by linking the models.

## RECOMMENDATIONS FOR FUTURE RESEARCH

This project has demonstrated that integration of independent domain models using OpenMI is possible, and has explained and demonstrated the advantages of model integration in being able to better represent real world complexity and thus to provide a systems view of water resource systems. The application of the integrated *ACRU* and *MIKE BASIN* models by users outside of the project team would not be easy, as an understanding of the OpenMI model linkage mechanism and the individual models is required. An open modelling environment named Delta Shell is being developed by the Dutch research institute Deltares. This integrated modelling will include OpenMI tools to enable models to be linked but also facilitate communication between models and the modelling environment which will provide GIS, data management and analysis tools. Delta Shell should be investigated further once it is released, both for the modelling environment itself and the approach adopted to facilitate use of OpenMI.

The performance penalty and memory requirements when linking models using OpenMI needs to be further investigated. Once a stable version of OpenMI 2.0 SDK has been released by the OpenMI Association the development of OpenMI 2.0 compliant wrappers for the *ACRU* and *MIKE BASIN* models should be considered as it is expected to offer better performance and improved user interface tools for linking models.

The integration of additional models, such as groundwater, water quality and economics models, using OpenMI would enable the OpenMI model linkage mechanism to be tested further. The integration of additional models, representing other domains, would also enable the investigation of the advantages and potential problems associated with modelling feedbacks between the various domains.

Considerable expertise has been developed through this project in the use of OpenMI to dynamically link legacy models. While the linked models have been demonstrated to operate on a real catchment, it is recommend that the expertise developed in the project be used to install and operationalize the linked models such that they can be used by water resource managers (e.g. by a CMA). It is anticipated that this will lead to further developments and refinements in order to meet the requirements of the water resource managers. This will also utilise the expertise developed during the project which, with no follow up research or operationalization project, is in danger of dispersing and being lost to the water community in South Africa.

## ACKNOWLEDGEMENTS

The WRC for funding this research project.

The Reference Group of this WRC project for their contributions during the project:

Mr W Nomqophu (Chairman)	Water Research Commission
Mr T Badenhorst	Department of Water Affairs
Mr A Bailey	SSI
Ms KT Chetty	University of KwaZulu-Natal
Mr MJC Horan	University of KwaZulu-Natal
Mr B Jackson	Inkomati Catchment Management Agency
Mr A Jeleni	Muondli Consulting
Prof GPW Jewitt	University of KwaZulu-Natal
Mr H Keuris	Department of Water Affairs
Dr N Lecler	SASRI
Mr KB Meier	Umgeni Water
Dr B Mwaka	Department of Water Affairs
Mr SM Ngoepe	Department of Water Affairs
Ms C Nthuli	Department of Water Affairs
Mr AJ Pott	DHI
Prof RE Schulze	University of KwaZulu-Natal
Mr MJ Summerton	Umgeni Water
Mrs I Thompson	Department of Water Affairs
Mr CD Tylcoat	Department of Water Affairs
Mr P van Niekerk	Department of Water Affairs
Mr NA Ward	Department of Water Affairs
Ms T Zokufa	Department of Water Affairs

The University of KwaZulu-Natal (UKZN) for provision of office space, administrative support and computing equipment to project team members. The administrative staff in the School of Engineering for their general assistance.

DHI-SA for providing copies of the MIKE BASIN software with associated licenses and dongles to the project team, for use within the project, as part of an agreement with the University of KwaZulu-Natal, providing training in the use of MIKE BASIN, support with the

MIKE BASIN configuration for the Kaap River Catchment, and for their assistance in getting the configurations of MIKE BASIN used for the evaluation of linking mechanisms running.

To the Inkomati Catchment Management Agency (ICMA), for allowing DHI-SA to provide support to the project team in configuring MIKE BASIN for the Kaap River Catchment.

CADSWES for providing an evaluation licence for the RiverWare software and especially Edith Zagona and David Neumann for their valuable assistance via e-mail in getting the configurations of RiverWare used for the evaluation running and assistance with the evaluation of linking mechanisms.

John Labadie and Johan Van Zyl for their response to e-mailed queries regarding MODSIM.

Richard Kunz for extracting the daily temperature and Penman-Monteith time-series data use in the Kaap River Catchment configuration.

The members of the Project Team for the work that they contributed to the project:

Prof JC Smithers (Project Leader)	University of KwaZulu-Natal
Mr DJ Clark (Responsible Researcher)	University of KwaZulu-Natal
Mr SLC Thornton-Dibb (Researcher)	University of KwaZulu-Natal
Mr A Lutchminarain (Researcher)	University of KwaZulu-Natal
Mr R Winckworth (MSc student)	University of KwaZulu-Natal



# TABLE OF CONTENTS

	Page
EXECUTIVE SUMMARY .....	iii
ACKNOWLEDGEMENTS .....	xv
TABLE OF CONTENTS .....	xvii
LIST OF FIGURES .....	xx
LIST OF TABLES .....	xxiv
LIST OF ABBREVIATIONS .....	xxvi
<b>1 INTRODUCTION AND OBJECTIVES .....</b>	<b>1</b>
<b>2 REVIEW AND EVALUATION OF RIVER NETWORK MODELS .....</b>	<b>12</b>
2.1 Review and Initial Evaluation of Available Models .....	12
2.1.1 User needs and requirements .....	15
2.1.2 River/reservoir network models .....	17
2.1.3 Discussion and recommendation .....	36
2.2 Detailed Evaluation of Selected Models .....	39
2.2.1 Evaluation criteria .....	42
2.2.2 MIKE BASIN .....	46
2.2.3 MODSIM .....	55
2.2.4 RiverWare .....	63
2.2.5 Results and recommendation .....	72
<b>3 REVIEW AND EVALUATION OF MODEL LINKAGE MECHANISMS .....</b>	<b>74</b>
3.1 Review of Available Mechanisms .....	75
3.1.1 Open Modelling Interface (OpenMI) .....	80
3.1.2 Object Modelling System (OMS) .....	96
3.1.3 Jena Adaptable Modelling System (JAMS) .....	102
3.1.4 The Invisible Modelling Environment (TIME) .....	105
3.1.5 LIQUID <sup>®</sup> Modelling Framework .....	115
3.1.6 Earth System Modelling Framework (ESMF) .....	119
3.1.7 Modular Modelling System (MMS) .....	120
3.1.8 High Level Architecture (HLA) .....	120
3.1.9 Common Component Architecture (CCA) .....	122
3.1.10 Discussion and recommendation .....	124
3.2 Evaluation of Linkage Mechanisms in Selected Models .....	126
3.2.1 Evaluation criteria .....	127
3.2.2 MIKE BASIN .....	130
3.2.3 MODSIM .....	133
3.2.4 RiverWare .....	136
3.2.5 Results and recommendation .....	144
<b>4 ACRU MODEL DEVELOPMENT .....</b>	<b>146</b>
4.1 Background .....	147
4.1.1 The ACRU 3.00 version .....	147
4.1.2 The ACRU2000 version .....	148

4.1.3	The <i>ACRUXml</i> version .....	149
4.2	Development of XML Input Files .....	151
4.2.1	ModelData schema .....	152
4.2.2	ModelConfiguration schema .....	160
4.2.3	XmlModelFiles libraries .....	168
4.3	Model Development .....	171
4.3.1	Component types and component configurations .....	173
4.3.2	Components .....	175
4.3.3	Internal data structure .....	182
4.3.4	Resources .....	189
4.3.5	Scenarios .....	191
4.3.6	State data and hotstarting .....	191
4.3.7	Dynamic data variables .....	192
4.3.8	Data readers and writers .....	192
4.4	Summary and Conclusions .....	193
5	IMPLEMENTATION OF OPENMI FOR MODEL LINKING .....	194
5.1	Development of OpenMI Wrapper for MIKE BASIN .....	199
5.1.1	OpenMI 1.4 wrapper .....	200
5.1.2	OpenMI 2.0 wrapper .....	204
5.2	Development of OpenMI Wrapper for <i>ACRU</i> .....	209
5.2.1	OpenMI 1.4 Java wrapper .....	210
5.2.2	OpenMI 1.4 .Net wrapper .....	212
5.3	Results and Recommendation .....	213
6	USE CASES FOR THE LINKED MODELS .....	214
6.1	Streamflow Links .....	218
6.1.1	Water users and return flows .....	219
6.1.2	Multiple water sources for a user .....	220
6.1.3	Water allocation options .....	220
6.1.4	Flow routing .....	220
6.1.5	Scenarios .....	221
6.2	Streamflow and Groundwater Links .....	221
6.3	Irrigation Requirement and Supply Links .....	224
6.4	Water Quality Links .....	225
6.5	Operations Modelling .....	226
7	CASE STUDY .....	228
7.1	MIKE BASIN Configuration .....	229
7.2	<i>ACRU</i> Configuration .....	232
7.2.1	Subcatchment and HRU configuration .....	232
7.2.2	Climate data .....	234
7.2.3	Soils and land-use .....	236
7.2.4	Dams and irrigated areas .....	237
7.3	Verification Studies on Quaternary Subcatchments X23A, X23C and X23E .....	240
7.4	Use Cases .....	251
7.4.1	Streamflow links .....	252

7.4.2	Streamflow links with flow routing .....	255
7.4.3	Streamflow and irrigation links .....	256
7.4.4	Results .....	258
8	DISCUSSION AND CONCLUSIONS .....	265
9	RECOMMENDATIONS .....	268
10	CAPACITY BUILDING.....	270
11	REFERENCES .....	271
	APPENDIX:.....	280

## LIST OF FIGURES

		Page
Figure 1.1	Mind map of the water system, management, modelling loop.....	7
Figure 2.1	Simple river diagram.....	17
Figure 2.2	Simple river network diagram .....	17
Figure 2.3	Schematic diagram of database and information processes in AWRIS (BoM, 2010b) .....	20
Figure 2.4	Schematic of hypothetical test catchment .....	39
Figure 2.5	FarmReservoir level-area-volume relationships .....	41
Figure 2.6	Reservoir 1 level-area-volume relationships.....	41
Figure 2.7	MIKE BASIN project within ESRI ArcMap (DHI, 2009).....	47
Figure 2.8	MIKE BASIN Catchment Properties window (DHI, 2009).....	48
Figure 2.9	MODSIM's user interface (Labadie, 2010a) .....	56
Figure 2.10	Catchment 1 configuration with Flowthru component (Labadie, 2010a).....	61
Figure 2.11	RiverWare's user interface (CADSWES, 2011).....	64
Figure 2.12	RiverWare's reservoir properties window (CADSWES, 2011).....	64
Figure 3.1	The OpenMI Version 1.4 standard interfaces (OpenMI, 2011a).....	85
Figure 3.2	Example of two model applications linked after implementation of the OpenMI interface standard (Moore and Tindall, 2005) .....	87
Figure 3.3	Example of the exchange of flow quantities between the two linked model applications shown in Figure 3.2 (Moore and Tindall, 2005) .....	87
Figure 3.4	Illustration of data exchange between OpenMI compliant models using the <i>GetValues</i> method (Moore and Tindall, 2005).....	89
Figure 3.5	OpenMI architecture namespaces (Moore and Tindall, 2005).....	91
Figure 3.6	Conceptual layout of OMS (Kralisch <i>et al.</i> , 2005).....	98
Figure 3.7	Application of compound components to control execution of model components in time and space (Kralisch <i>et al.</i> , 2005) .....	100
Figure 3.8	Representing spatial entities using OMSEntity objects (Kralisch <i>et al.</i> , 2005).....	100
Figure 3.9	Organisation of the JAMS framework Kralisch and Krause (2006).....	103
Figure 3.10	Architectural layers of the TIME modelling framework (Rahman <i>et al.</i> , 2003).....	107
Figure 3.11	Core classes of the Kernel layer (Rahman <i>et al.</i> , 2003) .....	107
Figure 3.12	Representation of units in TIME (Rahman <i>et al.</i> , 2003).....	109

Figure 3.13	Example of a simple model including TIME custom metadata tags (after Rahman <i>et al.</i> , 2003) .....	111
Figure 3.14	TIME Visualisation Framelet classes (Rahman <i>et al.</i> , 2003) .....	112
Figure 3.15	The main sections of LIQUID <sup>®</sup> (Branger <i>et al.</i> , 2010a) .....	116
Figure 3.16	Architecture of a LIQUID <sup>®</sup> module (Branger <i>et al.</i> , 2010a) .....	117
Figure 3.17	Illustration of river network model linkage mechanism requirements .....	128
Figure 3.18	Structure of the proposed custom RiverWare solution .....	139
Figure 4.1	Main components and data flows for <i>ACRUXml</i> and <i>SPATSIM_HDSF</i> .....	151
Figure 4.2	The initial design of the <i>ModelData</i> schema .....	153
Figure 4.3	The revised version of the <i>ModelData</i> schema .....	153
Figure 4.4	The new <i>ModelInfo</i> element containing a list of <i>Data</i> elements .....	154
Figure 4.5	The <i>Components</i> element in the initial <i>ModelData</i> schema .....	155
Figure 4.6	The <i>Components</i> element in the revised <i>ModelData</i> schema .....	156
Figure 4.7	The <i>Relationships</i> element in the revised <i>ModelData</i> schema .....	157
Figure 4.8	The <i>DataRef</i> element in the revised <i>ModelData</i> schema .....	157
Figure 4.9	The initial design of the <i>Data</i> element .....	158
Figure 4.10	The revised version of the <i>Data</i> element. ....	159
Figure 4.11	The initial design of the <i>ModelConfiguration</i> schema .....	160
Figure 4.12	The revised version of the <i>ModelConfiguration</i> schema .....	161
Figure 4.13	The new <i>ModelInfo</i> element in the <i>ModelConfiguration</i> schema .....	161
Figure 4.14	The <i>ComponentTypes</i> element in the initial <i>ModelConfiguration</i> schema...	162
Figure 4.15	The <i>ComponentTypes</i> element in the revised <i>ModelConfiguration</i> schema .....	162
Figure 4.16	The <i>Lookups</i> element .....	163
Figure 4.17	The <i>ComponentConfiguration</i> element in the initial <i>ModelConfiguration</i> schema. ....	165
Figure 4.18	The <i>ComponentConfiguration</i> element in the revised <i>ModelConfiguration</i> schema. ....	165
Figure 4.19	The <i>DataDef</i> element in the revised <i>ModelConfiguration</i> schema. ....	167
Figure 4.20	The <i>DataGroup</i> element in the revised <i>ModelConfiguration</i> schema .....	168
Figure 4.21	Simplified UML diagram of the <i>XmlModelFiles.ModelData</i> package .....	169
Figure 4.22	Simplified UML diagram of the <i>XmlModelFiles.ModelConfiguration</i> package .....	170
Figure 4.23	The main classes of the <i>ACRU</i> model .....	172
Figure 4.24	A UML Class Diagram showing the new main <i>Component</i> classes in the <i>ACRUXml</i> version of the model .....	177

Figure 4.25	A UML Class Diagram showing the main subcomponents of the main spatial Component classes in the <i>ACRUXml</i> version of the model .....	178
Figure 4.26	A UML Class Diagram showing the main Data classes used in the <i>ACRU2000</i> version of the model .....	184
Figure 4.27	A UML Class Diagram showing the new main Data classes created for the <i>ACRUXml</i> version of the model .....	185
Figure 4.28	A diagram of the extended ModelConfiguration schema showing the new Resource related elements .....	191
Figure 5.1	Illustration depicting a link between components in OpenMI 1.4 (OpenMI, 2012).....	194
Figure 5.2	An example of linking components in OpenMI 2.0 (OpenMI, 2012) .....	195
Figure 5.3	An example of an OpenMI compliant wrapper (Blind <i>et al.</i> , 2005).....	197
Figure 5.4	Example of an OMI file .....	199
Figure 5.5	Example showing <i>LinkableComponents</i> loaded and connected in the OpenMI 1.4 Configuration Editor .....	199
Figure 5.6	The OpenMI 1.4 SDK configuration editor.....	202
Figure 5.7	The <i>Run properties</i> dialog box of the configuration editor .....	203
Figure 5.8	<i>LinkableEngine</i> abstract class showing methods .....	205
Figure 6.1	<i>Status quo</i> simple series link by file translation .....	216
Figure 6.2	<i>Status quo</i> simple series link by custom code .....	216
Figure 6.3	Linking <i>ACRU</i> and MIKE BASIN using OpenMI .....	217
Figure 6.4	Key to the use case diagrams used in this chapter .....	218
Figure 6.5	Streamflow use case .....	219
Figure 6.6	Surface runoff and groundwater recharge use case using the first method.....	222
Figure 6.7	Surface runoff and groundwater recharge use case using the second method.....	223
Figure 6.8	Streamflow, irrigation requirement and supply use case .....	225
Figure 6.9	Water quality use case .....	226
Figure 6.10	Operations modelling use case .....	227
Figure 7.1	The Kaap River Catchment is represented by the tertiary catchment X23 which is located in the primary catchment X, namely the Inkomati.....	228
Figure 7.2	Altitude variation in the Kaap River Catchment .....	230
Figure 7.3	MIKE BASIN configuration for the Kaap River Catchment .....	231
Figure 7.4	Location of quaternary catchments, subcatchments, flow gauging weirs and selected rain gauges.....	232
Figure 7.5	Subcatchment flow network for the Kaap River Catchment .....	233

Figure 7.6	Distribution of MAP in the Kaap River Catchment (after Schulze <i>et al.</i> , 2008).....	235
Figure 7.7	Rain gauges in or near the Kaap River Catchment .....	236
Figure 7.8	Broad soil classification within the Kaap River Catchment (after ISCW, 2005).....	238
Figure 7.9	Land-use of the Kaap River Catchment (after NLC, 2005).....	239
Figure 7.10	Example of configuration of HRUs within subcatchments .....	240
Figure 7.11	Flow gauging weir X2H010 (DWA, 2012b) .....	241
Figure 7.12	Simulated vs observed monthly streamflow for Weir X2H010.....	242
Figure 7.13	Accumulative rainfall, observed and simulated streamflow for Weir X2H010.....	242
Figure 7.14	Plantation coverage (After Jackson, 2012).....	244
Figure 7.15	Daily mean observed discharge for X2H010 .....	244
Figure 7.16	7-Day Minimum flow for Weir X2H010 (after Jewitt <i>et al.</i> , 1999).....	245
Figure 7.17	Rating curves for Weir X2H010 .....	246
Figure 7.18	Double mass plot of the rainfall and observed streamflow .....	246
Figure 7.19	Flow gauging weir X2H024 (DWA, 2012) .....	247
Figure 7.20	Daily mean observed discharge for X2H024 .....	247
Figure 7.21	Simulated vs observed monthly streamflow for Weir X2H024.....	248
Figure 7.22	Accumulative rainfall, observed and simulated streamflow for Weir X2H024.....	248
Figure 7.23	Flow gauging weir X2H008 (DWA, 2012) .....	249
Figure 7.24	Daily mean observed discharge for X2H008 .....	249
Figure 7.25	Simulated vs observed monthly streamflow for Weir X2H008.....	250
Figure 7.26	Accumulative rainfall, observed and simulated streamflow for Weir X2H008.....	250
Figure 7.27	The MIKE BASIN configuration for the streamflow use case.....	254
Figure 7.28	The MIKE BASIN configuration for the streamflow and irrigation use case .....	257
Figure 7.29	Accumulated flow ( $10^6 \text{ m}^3$ ) at Weir X2H022 for the period 1990 to 1999 ....	259
Figure 7.30	Flow rates for Weir X2H022 for early 1996.....	260
Figure 7.31	Accumulated flow ( $10^6 \text{ m}^3$ ) at Weir X2H022 for the period 1996 to 1999 ....	261
Figure 7.32	Flow rates for Weir X2H022 for the 1997-1998 rainy season.....	262
Figure 7.33	Flow rates for Weir X2H022 for the 1998-1999 rainy season.....	262
Figure 7.34	Example of the effect of flow routing on flow rates for Weir X2H022.....	263

## LIST OF TABLES

	Page
Table 1.1	Water resource management tasks and decisions for planning ..... 3
Table 1.2	Water resource management tasks and decisions for operations ..... 4
Table 2.1	RiverWare objects (after Wurbs, 2005) ..... 33
Table 2.2	Summary of selected models ..... 37
Table 2.3	Catchment details ..... 40
Table 2.4	Sequence of average monthly flows in the <i>Catchrun</i> file ..... 40
Table 2.5	Reservoir level characteristics ..... 41
Table 2.6	Lookup table for diversions to FarmReservoir ..... 42
Table 2.7	Water user details for the priority allocation test case ..... 44
Table 2.8	Reservoir curtailment levels ..... 44
Table 2.9	Water user details for the FWACS allocation test case ..... 45
Table 2.10	MIKE BASIN components (after DHI, 2009) ..... 50
Table 2.11	MIKE BASIN evaluation scores ..... 55
Table 2.12	MODSIM components (after Labadie, 2010a) ..... 58
Table 2.13	MODSIM evaluation scores ..... 62
Table 2.14	RiverWare components (after CADSWES, 2011) ..... 66
Table 2.15	RiverWare evaluation scores ..... 72
Table 2.16	Summary of evaluation ratings ..... 73
Table 3.1	A list of models and other software that are OpenMI Version 1.4 compliant (after (OpenMI, 2011b) ..... 94
Table 3.2	TIME custom metadata tags defined in the Kernel layer (Rahman <i>et al.</i> , 2003; Murray <i>et al.</i> , 2007) ..... 109
Table 3.3	Some specialised TIME data types (Rahman <i>et al.</i> , 2003) ..... 110
Table 3.4	Modelling tools within the eWater Catchment Management Toolkit (after <a href="http://www.toolkit.net.au">http://www.toolkit.net.au</a> ) ..... 113
Table 3.5	Simple quantitative evaluation of the systems reviewed where 1 = strong, 0 = average, -1 = weak ..... 126
Table 3.6	Comparison of river network models based on requirements ..... 144
Table 3.7	Comparison of river network models based on OpenMI criteria ..... 145
Table 4.1	Attributes of the <i>DataDef</i> element ..... 167
Table 4.2	The instance variables belonging to the <i>DData</i> class and their descriptions ..... 187



Table 4.3	Time series types that apply to the <i>DTimeSeries</i> class and their descriptions.....	188
Table 5.1	The <i>IEngine</i> interface methods.....	198
Table 5.2	Implementation of the <i>IEngine</i> interface methods in the <i>MBEngineWrapper</i> class .....	200
Table 5.3	The <i>LinkableEngine</i> abstract class methods .....	205
Table 5.4	Implementation of the <i>LinkableEngine</i> abstract class methods.....	206
Table 5.5	Implementation of the <i>IEngine</i> interface methods in the <i>AcruEngineWrapper</i> class .....	210
Table 7.1	Quaternary catchment and subcatchment areas.....	233
Table 7.2	Characteristics of rain gauges in or near the Kaap River Catchment.....	236
Table 7.3	Summary of land use in SubCatchment_10 .....	243
Table 7.4	Summary of land use in SubCatchment_11 .....	243
Table 7.5	The linked model variables for the streamflow use case .....	253
Table 7.6	The additional linked model variables for the irrigation use case .....	258
Table 10.1	Students supported by the project. ....	270

## LIST OF ABBREVIATIONS

<b>ANN</b>	Artificial Neural Networks
<b>API</b>	Application Programming Interface
<b>AWRIS</b>	Australian Water Resources Information System
<b>BASINS</b>	The Better Assessment Science Integrating Point and Nonpoint Sources
<b>BEEH</b>	School of Bioresources Engineering and Environmental Hydrology (former)
<b>CADSWES</b>	Center for Advanced Decision Support for Water and Environmental Systems
<b>CAT</b>	Case Analysis Tool
<b>CCA</b>	Common Component Architecture
<b>CDSI</b>	Conservation Delivery Streamlining Initiative
<b>CMA</b>	Catchment Management Agency
<b>CMT</b>	Case Management Tool
<b>COM</b>	Component Object Model
<b>CRC</b>	Co-operative Research Centre
<b>CRCCH</b>	Cooperative Research Centre for Catchment Hydrology Systems
<b>CWMS</b>	Corps Water Management System
<b>CWRR</b>	Centre for Water Resources Research
<b>DEM</b>	Digital Elevation Model
<b>DHI</b>	Danish Hydrological Institute
<b>DHI-SA</b>	Danish Hydrological Institute in South Africa
<b>DMI</b>	Data Management Interface
<b>DMSO</b>	Defence Modelling and Simulation Office
<b>DSS</b>	Decision Support System
<b>DWA</b>	Department of Water Affairs
<b>DWAF</b>	Department of Water Affairs and Forestry (former)
<b>ESMF</b>	Earth System Modelling Framework
<b>FOM</b>	Federation Object Model
<b>FSU</b>	Friedrich Schiller University
<b>FU</b>	Functional Unit
<b>FWACS</b>	Fractional Water Allocation And Capacity Sharing
<b>GIS</b>	Geographic Information System
<b>GUI</b>	Graphical User Interface
<b>HDSF</b>	Hydrological Decision Support Framework
<b>HEC</b>	Hydrologic Engineering Centre

<b>HEC-DSS</b>	HEC Data Storage System
<b>HEC-ResSim</b>	HEC Reservoir System Simulation
<b>HLA</b>	High Level Architecture
<b>HPC</b>	High-performance Computing
<b>HRU</b>	Hydrological Response Unit
<b>ICMA</b>	Inkomati Catchment Management Agency
<b>IFR</b>	In-stream Flow Requirement
<b>IWR</b>	Institute for Water Resources
<b>IWRM</b>	Integrated Water Resources Management
<b>JAMS</b>	Jena Adaptable Modelling System
<b>JNI</b>	Java Native Interface
<b>LP</b>	Linear Programming
<b>MAC</b>	Media Access Control
<b>MAP</b>	Mean Annual Precipitation
<b>MDB</b>	Murray-Darling Basin
<b>MDI</b>	Multiple Document Interface
<b>MMS</b>	Modular Modelling System
<b>N/A</b>	Not Applicable
<b>NexGen</b>	HEC Next Generation
<b>NWA</b>	National Water Act of South Africa, Act 36 of 1998
<b>NWRS</b>	National Water Resources Strategy
<b>ODBC</b>	Open DataBase Connectivity
<b>OMS</b>	Object Modelling System
<b>OpenMI</b>	Open Modelling Interface
<b>PEST</b>	Parameter Estimation
<b>PRMS</b>	Precipitation-Runoff Modelling System
<b>PUMMA</b>	Peri-Urban Model for Landscape Management PUMMA
<b>RBIS</b>	River Basin Information System
<b>Rcl</b>	RiverWare Command Language
<b>REALM</b>	The REsource ALlocation Model
<b>RIBASIM</b>	River Basin Simulation Model
<b>RPL</b>	RiverWare Policy Language
<b>RRM</b>	Rainfall-Runoff Model
<b>RTI</b>	Runtime Infrastructure
<b>RZWQM</b>	Root Zone Water Quality Model
<b>SDK</b>	Software Development Kit
<b>SEI</b>	Stockholm Environment Institute

<b>SIDL</b>	Scientific Interface Definition Language
<b>SMW</b>	Simple Model Wrapper
<b>SOM</b>	Simulation Object Model
<b>SWAT</b>	Soil and Water Assessment Tool
<b>TIME</b>	The Invisible Modelling Environment
<b>UKZN</b>	University of KwaZulu-Natal
<b>UML</b>	Universal Modelling Language
<b>URL</b>	Uniform Resource Locator
<b>USGS</b>	US Geological Survey
<b>WASP5</b>	Water Quality Simulation Program (Version 5)
<b>WEAP</b>	Water Evaluation And Planning
<b>WMA</b>	Water Management Area
<b>WRAP</b>	Water Rights Analysis Package
<b>WRC</b>	Water Research Commission
<b>WRPM</b>	Water Resources Planning Model
<b>WRYM</b>	Water Resources Yield Model
<b>XML</b>	Extensible Markup Language

# 1 INTRODUCTION AND OBJECTIVES

DJ Clark and JC Smithers

Water is a critical and a scarce resource in South Africa. Hence, it is essential that water resources are managed efficiently and equitably, as required by the National Water Act (NWA, 1998) of South Africa (Act 36 of 1998) .

Computer models of water resource systems are a tool for understanding and managing water resources in South Africa. There are two main drivers for the need for better water resources modelling tools in South Africa: (i) in many catchments water demand exceeds available supply, and (ii) the requirements of the NWA. In DWAF (2004) it is reported that in the year 2000 already 10 of the 19 Water Management Areas (WMAs) in South Africa were water stressed, i.e. the demand for water exceeded supply. All, except one, of these WMAs are linked by inter-catchment transfers that assist in the transfer of water from areas with adequate supply and low demand to highly developed areas with high demand (DWAF, 2004). Water resource managers require improved water resources modelling tools for planning and operations to assist them in making management decisions leading to better water allocation and improved water use efficiency. There is a growing realisation of the need for integrated water resources management, including both water quantity and quality components and incorporating environmental, social, economic and political aspects of water, and this need is reflected in the NWA. Typically models are developed for specific domains within the water resource system and integrated water resources management will require integration of the models representing specific domains in order to provide a systems perspective for water management decisions.

This WRC project was preceded by a one year consultancy with the purpose of (i) evaluating user needs with regard to modelling for water resource planning and operations, (ii) reviewing existing modelling tools for water resource planning and operational management, and (iii) making recommendations for further research and the development of modelling tools (Pott *et al.*, 2008b). The following modelling requirements were identified by Pott *et al.* (2008b):

- The need to model water quality in addition to water quantity.
- The need to model at appropriate temporal and spatial scales.
- Models need to represent real life complexity to adequately mimic hydrological processes and realities on the ground.
- Modelling tools are required for both planning and operations.

- Integrated modelling in a Decision Support System (DSS) is required.
- Alternative methods and scenarios of water allocation/apportionment, including Fractional Water Allocation and Capacity Sharing (FWACS), need to be assessed in order to promote efficient water use.
- The conjunctive use of surface water and groundwater needs to be integrated in models.
- The modelling system should link irrigation with water supply limited by operating rules in order to simulate crop yields.
- To assess the impacts of transferring water use rights.
- To include modelling of economic and social impacts.
- To assess impacts of climate change on water resources and agricultural productivity.
- To perform real time modelling for operational management.
- To include a DSS for managing real time volumetric water abstractions.
- The operational modelling must account for real life operational situations.
- Feedback loops between water demand and supply to determine impact of different operating decisions must be included.
- Flow routing is necessary for operations modelling.
- The modelling system should include water operating rules and releases that can be applied on a day-to-day basis.
- Tools are required to operationalize the reserve.
- The modelling system must be able to use climate forecasts to aid operational decisions.
- Water accounting and auditing of water use combined with metering and monitoring is necessary.
- Modelling results must be verified against measured data.
- More user friendly model front and back ends are necessary to assist in setting up models and communicating results to stakeholders.

In a subsequent meeting with Mr Brian Jackson from the Inkomati CMA, the following modelling requirements were identified (Jackson, 2009):

- Water accounting and auditing.
- Operationalization of the Reserve at a daily time scale.
- Determine quantity of surplus water (i.e. in excess of allocated water).
- Easy means of running scenarios to assess impacts of restrictions, licences and trading on water users, especially downstream users.

- Physical modelling to evaluate land use scenarios (e.g. impact of change of land use on water resources).
- Model the impact of off-channel dams.
- Operate dams as part of a system not individually.
- Use of short and long term forecasts for planning.
- Need to be able to easily update models with rainfall and system state data (e.g. dam levels).

Based on these requirements a set of water management tasks and decisions and the modelling requirements to assist water managers with these was created. The main tasks and decisions were divided into two sets, planning and operations, and are described in Table 1.1 and Table 1.2 respectively.

Table 1.1 Water resource management tasks and decisions for planning

<b>Task/Decision</b>	<b>Description</b>
Water quantity (yield) determination	One of the primary water management tasks is to estimate the quantity of water available within a catchment and the level of assurance of this availability. These estimates need to account for the spatial and temporal variability of the climate variables driving the hydrology. In addition to climate variability the influence of climate change also need to be considered. The methods used to estimate water availability for planning purposes needs to be compatible with the methods used for water resource operations.
Assessing new licence applications	Water managers need to assess water use licence applications to determine: <ul style="list-style-type: none"> <li>• if there is sufficient quantity of water available,</li> <li>• of a suitable quality,</li> <li>• the impacts of any associated change in land use, and</li> <li>• the impact of quantity and quality of water discharged.</li> </ul>
Water quality	There is increasing awareness and concern regarding water quality in catchments. The NWA requires water managers to assess and manage the quality of the water resources under their control.
Impact assessment	Catchments are in a continual state of change as they develop. These changes include: urbanisation, industry, land use and management changes, irrigation, transfer of water use rights and water infrastructure such as dams. Water managers need to assess the impacts of these changes on water availability and quality.
Water use efficiency	Water managers should promote water use efficiency, especially in stressed catchments, to increase assurance of supply to existing users or make water available to new water users. It may be possible to increase water use efficiency through the adoption of alternative water allocation methods such as FWACS in place of a priority based system. Water use efficiency should be considered when allocating water use licences, including socio-political criteria in addition to economic benefits.

Table 1.2 Water resource management tasks and decisions for operations

<b>Task/Decision</b>	<b>Description</b>
Data management and storage	In order for water managers to make informed decisions they require data and information about the water resource they are managing. This data includes historical data, real time data and records of water trades. This data and information needs to be obtained, quality controlled, stored, accessed and analysed.
Monitoring	If the required time varying data and information are not available from state or commercial sources then a monitoring network will need to be established at a suitable scale to monitor streamflow, rainfall and climate variables used to estimate evaporation such as temperature, humidity and solar radiation.
Meeting licences/demands	The National Water Act makes provision for a Reserve to meet basic human needs and environmental requirements. A Reserve determination needs to be conducted, then a plan to fulfil the Reserve requirements and finally how to provide water for the Reserve operationally through releases from a dam or restrictions on water users. The water resources in a catchment need to be allocated to meet demands in priority order of the Reserve, international obligations and then demands from other sectors (e.g. industry, irrigation). In catchments with water infrastructure such as dams and diversions it is necessary to operate this infrastructure to provide water to licensed water users downstream. Water management includes conjunctive use of both surface water and groundwater.
Auditing and compliance	To give effect to water use licences users need to be informed of the water allocation quantities, surplus water and restrictions during droughts. Water use licences are only of use if all water users are honest in only using the water allocated to them and it may be necessary to monitor actual water use by means of weirs and flow meters so that water use can be audited. Monitoring of flows may also be necessary to ensure compliance with the Reserve.
Flood management	Flood management plans need to be put in place to enable control of floods, prevent development in high risk areas and to provide early warning systems.
Forecasting demands and supply	Recent advances in climate forecasting enable water managers to plan ahead in time to the next day, week, month or season and estimate future water demands and availability which can assist in the operational decisions they make in real time. Recent advances in remote sensing technologies provide water managers with valuable information about the current status of water resources within a catchment and potential crop water requirements.

The water resource modelling requirements identified to support these tasks include the need to adequately represent real work complexity, the need to integrate models representing different domains within the water resource system and other system requirements.

Water resource systems are complex even in their natural state and anthropogenic development within these systems adds to the complexity. All models are a simplification of the systems they represent, however, for a model to be useful it must be capable of



adequately representing real work complexity for a given application. In terms of water resource modelling representing this real world complexity includes being able to:

- Represent non-homogenous hydrological responses due to spatially variable catchment characteristics such as land use and soil type within a catchment, by modelling catchment subdivisions referred to as hydrological response units (HRUs). In a study by Chetty (2009) it was shown that area weighting soils in subcatchments and dividing a subcatchment up into HRUs based on land cover resulted in significantly better simulation results compared to using the dominant land use and soil for a subcatchment.
- Represent complex catchment configurations, including HRUs, irrigated areas, rivers, dams, intercatchment transfers and diverse water users.
- Model individual water users or groups of similar water users.
- Model water supply to a user from more than one water source.
- Model multiple water user demands from each water source.
- Model inter-catchment transfers on a time step by time step basis.
- Perform flow routing through rivers and dams to estimate flow lags and attenuations.
- Represent feedbacks between different hydrological processes or components of the water resource system on a time step by time step basis, for example irrigation demand, supply and return flows.
- Model dynamic changes in land use, management and abstractions (e.g. for irrigation) within catchments during a simulated time period.

There is a growing awareness of the need for a systems perspective of water resources instead of focusing on specific domains within the water resource system, such as surface water, groundwater or the environment. Ideally a water resources planning and operation toolkit should include models of terrestrial surface water, groundwater, the river and dam network, the water supply network, water quality, ecology and economic and social aspects of the water resource system. Many models have been developed for each of these domains, but these models need to be linked to give a systems perspective to enable water managers to make better decisions. Ideally these models need to be linked “in parallel”, i.e. on a time step by time step basis to model feedbacks between processes simulated in the different domains. Ideally these models should be run within a common modelling framework to minimise duplication of common tools for GIS and time series analysis.

Other modelling system requirements include:

- Being able to model hydrological processes in a physical conceptual manner.
- Being able to model at appropriate and variable spatial and temporal scales.
- Graphical user interfaces (GUIs) for models that are user friendly and enable easy and flexible configuration of model, including GIS and data analysis tools.
- Being able to easily configure and run scenarios.
- Provision and management of real time data for operations modelling.
- Adaptive modelling where simulated state parameters in models can be updated using observed real time data.
- Data management structures and tools and efficient model input and output.
- Evaluation of uncertainty in model domains and overall modelling uncertainty.
- Suitability for use for both planning and operations modelling.

Water resource managers, for example employed by Catchment Management Agencies (CMAs), will be required to perform many water management tasks for water resource planning and operations as shown in Figure 1.1. The water management tasks and decisions and the modelling requirements listed above are an indication of the need by water managers for a better understanding of the physical hydrological system they are managing and for better modelling tools to assist in managing the finite water resources under their control in an equitable and sustainable manner. To meet these requirements it will be necessary to integrate models developed for specific domains within the water resource system. The models selected need to represent the real world complexity of the physical hydrological system at appropriate spatial and temporal scales. As shown in Figure 1.1, the physical hydrological system, consisting of catchments and water users, requires management of the finite water resource, and to do this managers require integrated modelling tools to adequately represent the physical hydrological system being managed.

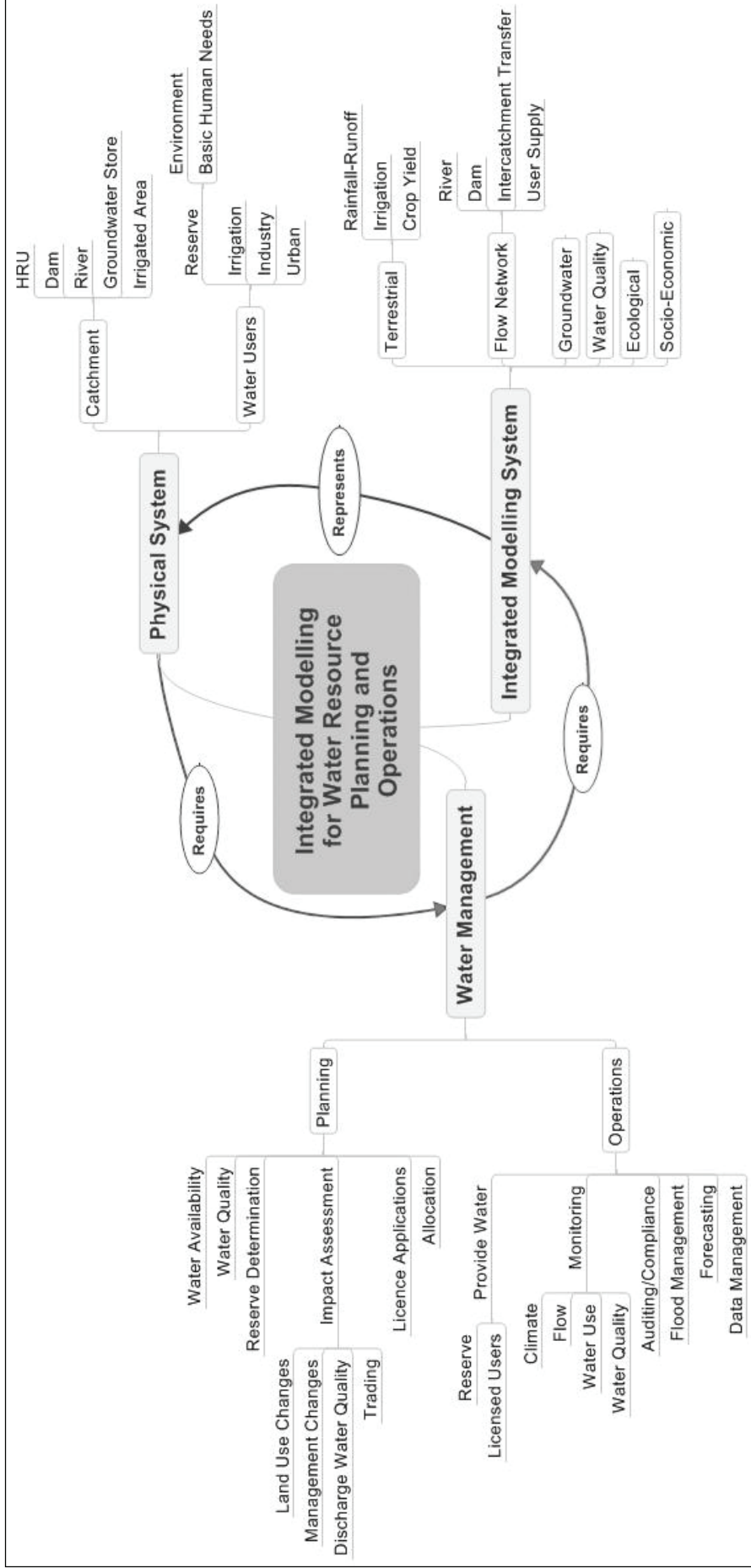


Figure 1.1 Mind map of the water system, management, modelling loop

Water resources modelling tools are required both for water resources planning and for water resources operations. Water resources modelling for planning is widely practiced in South Africa, though many of the water resources modelling tools currently in use were developed prior to the 1998 National Water Act (NWA, 1998) and may need to be updated, extended or replaced in order to meet the requirements of the National Water Act. The use of water resources modelling for operations appears to be less widely practiced in South Africa and is an area of water resources modelling that requires further development and implementation. Recent improvements in climate forecasting and remote sensing have increased the data and information available to water resource managers and expanded the scope for water resources modelling to aid operational decisions by water managers. It is important that the concepts and methodologies used in water resources modelling for planning and operations are compatible.

In South Africa the Pitman model used together with the Water Resources Yield Model (WRYM) and the Water Resources Planning Model (WRPM) are the modelling tools predominantly used for water resource planning. The WRYM and WRPM models operate at a monthly time step and are generally used at a relatively coarse spatial scale. The WRYM does not model water quality. The *ACRU* agrohydrological model has been used for more detailed planning studies, especially in studies of land use and climate change impacts but is not strong in the area of network modelling. More recently the *ACRU* model has been used together with the MIKE BASIN model, where *ACRU* is used to generate streamflow which is then used in MIKE BASIN for network modelling. However this is still a series link between the models and feedbacks such as irrigation return flows and water quality cannot be represented. These modelling tools in their current state are not suitable for assisting water managers in many of the tasks and decisions listed above. There is a growing realisation internationally of the need to integrate models representing different domains. The main shortcomings of the models mentioned above are limitations in being able to represent real world complexity, suitability for operational modelling and the ability to be easily linked with models for other domains. The restructuring of the *ACRU* model into an object oriented structure in the Java programming language (Lynch and Kiker, 2001) and the recent implementation of XML model input files (Clark *et al.*, 2009) were significant moves towards being able to model real world complexity. However, further development is still required to enable more flexible catchment configuration scenarios and modifications to make it suitable for modelling both the spatially non-uniform hydrological responses within a catchment and for operational modelling.

The broad objective of this project was to develop and integrate modelling tools to support water resource managers by meeting some of the modelling requirements identified for water resources planning and operations. The more specific objectives of this project were to:

- Review river network models which are suitable for water resource planning and operations and to select one to be integrated with the *ACRU* agrohydrological model.
- Investigate methods for linking different domain models such as a hydrological model and a river network model, then select and implement a suitable method for integrating the hydrological model and the selected river network model.
- Further develop the *ACRU* daily time step agrohydrological model in order to realistically represent the varying hydrological responses within the terrestrial hydrological system.
- Configure and apply the integrated hydrological model and river network model for selected catchments within the Inkomati WMA.

The first step towards an integrated modelling system is to model land based hydrological processes at suitable spatial and temporal scale to represent real world complexity. It is proposed that the *ACRU* agrohydrological model be used of this purpose for the following reasons:

- It has been developed and applied extensively in South Africa and is on the South African Department of Water Affairs (DWA) list of recommended models.
- The physical conceptual nature of the model makes it suitable for modelling a variety of land use scenarios.
- The object oriented model structure and object oriented XML input file structure is capable of representing real world complexity.
- It operates at a daily time step, which makes it suitable for operational modelling.
- The object oriented model structure enables parallel processing which enables feedbacks between catchments to be modelled.
- It includes water quality modules for sediment yield, salinity, and nitrogen and phosphorus modelling.
- It can be easily adapted to provide additional functionality required for operations modelling.
- It includes the concept of water ownership which is necessary for water accounting.

The *ACRU* model does not have sufficient river network modelling capabilities and specialised river network models typically rely on simple rainfall-runoff models or require streamflow as an input. Suitable river network models should:

- Model both water quality and quantity.
- Be suitable for planning and operations modelling.
- Support different water apportionment rules.
- Include functionality for water accounting and auditing.
- Include functionality to support water trading.
- Be able to solve complex water demand and supply networks.
- Be able to do flow routing to account for flow lags and attenuations.
- Be suited to linking with *ACRU* and other models.

Several suitable river network models are reviewed and evaluated in Chapter 2, resulting in a recommendation that the MIKE BASIN model be selected for use in conjunction with the *ACRU* model.

To integrate models it is necessary for data and information to be exchanged between the models. Linking two models in series is usually a simple matter of running the first model, converting the output format of the first model to the input format of the second model, and then running the second model. One critical drawback of the series linking approach is that feedbacks between processes in the two separate models cannot be represented. There are various means of integrating models, including linking models in parallel, these are reviewed and evaluated in Chapter 3. As a result of this review and evaluation the OpenMI Standard and model linking framework were recommended as being the most appropriate for use in this project. The implementation of OpenMI using wrappers for both the *ACRU* and MIKE BASIN models is explained in Chapter 5. The integration of the *ACRU* hydrological model and the MIKE BASIN river network model in this project was intended as a test case for the integration of additional models for other domains such as detailed groundwater models and socio-economic models. A set of use cases for the application of OpenMI to link the *ACRU* and MIKE BASIN model is presented in Chapter 6.

In the assessment of modelling requirements to support water resource planning and operations, a few areas of further development were identified for the *ACRU* model, including the following which are described in Chapter 4:

- Refinements to the design of the XML input files for the *ACRU* model to include: relationships between components of the hydrological system, catchment configuration options, a means of storing dynamic data and improved linkages to external data files.

- Modifications to the design of the XML input files for the *ACRU* model, to make it suitable for operational modelling, including: storage of state variables and scenario management.
- Modifications to the *ACRU* model, to make it suitable for operational modelling, including: storage of state variables, hotstarting, scenario management and saving output to other data formats.

It was proposed that the *ACRU* model and the MIKE BASIN model be configured for a selected catchment within the Inkomati WMA to demonstrate the use of the integrated models in a real catchment. The Inkomati WMA was proposed for the following reasons:

- It is a stressed catchment;
- It contains a variety of water users including irrigation, industry, urban, the Kruger National Park and international obligations to Mozambique and Swaziland;
- Monitoring of water use is already taking place;
- The Catchment Manager, Mr Brian Jackson, was supportive of this research;
- The project could build on and contribute to other research projects within the catchment.

The Kaap River Catchment was selected for the case study. The configuration of the *ACRU* and MIKE BASIN model for this catchment, and the application of the integrated models, is described in Chapter 7.

The outcomes of the project which include a review of river network models, a review of model linkage mechanisms, further development of the *ACRU* model, development of OpenMI compliant wrappers for the *ACRU* and MIKE BASIN models, and examples of how these integrated models can be applied is discussed in Chapter 8. Recommendations for future research are listed in Chapter 9.

## **2 REVIEW AND EVALUATION OF RIVER NETWORK MODELS**

**SLC Thornton-Dibb JC Smithers and DJ Clark**

There are a number of river network models and modelling tools available from software developers and research groups internationally. These models were reviewed and evaluated to gain a better understanding of how these models work and the functionality offered by each individual model to enable the most appropriate model to be selected for use in the project.

### **2.1 Review and Initial Evaluation of Available Models**

The planning and management of water resources are complex processes as many competing demands have to be simultaneously met. In South Africa these include demands related to meeting human needs for water, supplying water to sustain the environment through an environmental reserve, meeting growing demands from industry for water while also supporting food security by the allocation of water to meet irrigation demands. Fresh water is a critical and limited resource, particularly in a semi-arid country like South Africa and the increase in demand as well as the uncertainty and variability of supply, both spatially and in quantity and quality, adds to the complexity of managing this vital resource. Although there is a continued growth in the demand for water resources there are few, if any, dams being built in developing countries, thus emphasizing the need to maximise the potential of reservoirs and improving water use efficiency (Labadie, 2004).

The National Water Act (NWA, 1998) has transformed the way water is controlled, from a system of rights based on land ownership (the riparian system) to a system designed to allocate water equitably in the public interest according to the National Water Resources Strategy (NWRS), as detailed in DWAF (2004). This has been a significant change, which has influenced the user needs in South Africa for water resource planning and operational activities, bearing in mind that the currently applied water resources planning systems were largely conceptualized and developed in the era of the 1956 Water Act. As a consequence, the management of water resources in South Africa is currently faced with a number of inter-related challenges which include:

- To accurately determine, at detailed spatial and temporal scales and in an integrated manner, how much water can be allocated in various catchments (and subcatchments) in South Africa.



- To most appropriately allocate the limited water resources with stakeholder input and in accordance with the NWRS, and issue entitlements (licenses) amongst competing users. The allocation of water resources is critical given that more than 50% of catchments in South Africa are deemed to be over-allocated, with the demand for water exceeding the ability of the systems to supply the water within acceptable levels of assurance (DWAF, 2004).
- To ensure that license conditions are adhered to and to ensure that water resources are managed efficiently.

Planning models and methods are required for the first two challenges, while operational models and methods are required to address the third challenge. There should be a direct link between the planning and operational models in that both types of models have similar characteristics.

Planning models usually work on relatively coarse time steps, e.g. monthly or even annual time steps, and make use of historical data or synthetic streamflow sequences generated via stochastic algorithms from the historical data sets. The operational models on the other hand need to operate on a more current “now-time” basis, and may make use of forecast data (e.g. daily to seasonal forecasts), and often need to better reflect the complexities within catchments than the planning models, including the institutional arrangements (operating rules) prevalent in the catchment. Welsh (2011) indicated that in Australia the most common time step for planning models is now daily, with monthly models being phased out. Welsh (2011) also indicated and that operational models in Australia run at daily to hourly time steps and always include forecast data, and that both planning and operational models include the institutional arrangements (operating rules). It is important to note that finer time step and spatially detailed hydrological data can be accurately aggregated into larger time steps, but the converse is not true. Thus, in order to meet the requirements of the National Water Act (NWA, 1998) and NWRS (DWAF, 2004), it is anticipated that future planning and operational models will be based on spatially and temporally detailed configurations. The basis of these systems will be a rainfall-runoff model linked to a river network model. In addition, there has been a move towards a “bottom up” approach in water resources planning and management as stakeholders have become more actively involved in driving the requirements and needs in the planning process, as well as demanding more transparent systems to aid in policy and decision making (Assaf *et al.*, 2008).

In South Africa, the Pitman model used with the Water Resources Yield Model (WRYM) and the Water Resources Planning Model (WRPM) are the modelling tools predominantly used

for water resource planning (Basson *et al.*, 1994; Mckenzie and Van Rooyen, 2003). The WRYM and WRPM models operate at a monthly time step and are generally used at a relatively coarse spatial scale. The WRYM does not model water quality. Many of South Africa's dams lack a comprehensive system of operating rules, as shown in a Department of Water Affairs and Forestry business review (Manqoyi and Nyabeze, 2006). As a consequence, many of the dams in South Africa are managed using operator experience and rules of thumb and hence there is no clear link between planning and operations. The WRYM and WRPM models were not included in this review based on the conclusions made in Frezghi (2007) that the WRYM's monthly time step is too coarse to support all aspects of the implementation of the National Water Act (NWA, 1998) and the National Water Resources Strategy (NWRS), as detailed in DWAF (2004). For example, these models are not suited to modelling short term operational decisions requiring a daily time step, such as releases for environmental flows which mimic the natural hydrology, water quality processes where a monthly time step is too long, and the ability to route flows with lagging and attenuation of hydrographs through a river network. In addition, the WRYM is based on a priority allocation method and does not make provision for alternative allocation methods which may result in more effective allocation of water resources in some circumstances.

Generic simulation models are available for both supporting water resource planning and management in catchments and also for facilitating stakeholder involvement (Wurbs, 2005). Reservoir/river system network models track the movement of water through a system using volume mass balance accounting procedures (Assaf *et al.*, 2008). One of the challenges in developing these models is including different levels of complexity appropriate to the experience and needs of the user.

The objective of this section is to review river network models which are suitable for water resource planning and operations and to select a network model to be integrated with the *ACRU* daily time step hydrological model.

According to Wurbs (2005), a significant amount of work has been published, in addition to unpublished work, on developing and applying reservoir/river system models over the past 50 years and he provides an inventory of 15 generalised river network models used internationally and reviews 5 of these models in more detail (SUPER, HEC-ResSim, RiverWare, MODSIM and WRAP). More recently, Assaf *et al.* (2008) reviewed five river network simulation models (MODSIM, MIKE BASIN, RIBASIM, WBalMo, and WEAP) for river planning and operations.

Pott *et al.* (2008b) reviewed a number of models and systems used both locally and internationally for water resources planning and operations and also conducted a survey of stakeholders involved in water resources planning and management in South Africa. From the review of models and needs identified, Pott *et al.* (2008b) identified that the RiverWare model (Zagona *et al.*, 2001), the BASINS framework developed by the USA EPA (EPA, 2007), the MIKE BASIN model (DHI, 2010) and the Australian Water Resources Information System (AWRIS) currently under development (BoM, 2010b), as all having the potential to meet some or all of the needs identified. Comprehensive literature reviews of models used for water resources planning and operations have been conducted by Frezghi (2007) and Kime (2010) and are utilised in the model reviews in this section.

### **2.1.1 User needs and requirements**

From the expert panel and stakeholder workshops conducted by Pott *et al.* (2008b) the following issues related to water resource management in South Africa were raised:

- Water quality planning is lagging far behind planning for water quantity.
- Implementation of the National Water Act is not being operationalized.
- Tools and methods are needed which promote the efficient use of water by water users.
- Concerns were raised regarding the potential impact of climate change.
- Both short-term operational modelling and long term planning must take account of real life operational situations and monthly time step models are not capable of mimicking much of the complexity that exists at an operational level.
- New water apportionment methods need to be considered and supported by planning and operational models.
- There is a need for models to support the transfer of water use-rights.
- Both water accounting and auditing of water use is necessary.

Pott *et al.* (2008b) recognised the need to communicate the user rights to water users and to facilitate the trading of water. Based on inputs from the expert panel and stakeholder workshops, Pott *et al.* (2008b) envisage a water accounting system which will enable the following queries to be performed at a river node:

- the quantity and quality of water at the node (for a given point in time or time range),
- the water allocated to a user and hence the ownership of the water,
- the source/origin and destination of the water,
- the timing when the water will reach the abstraction point,

- the location where the water will be used,
- the water lost and changes in water quality as water flows downstream, and
- to be able to identify traded water.

In addition, records of water traded, trading opportunities and a comparison of actual vs. allocated water would need to be maintained by the water accounting system.

Wurbs (2005) suggested that the following should be taken into account when selecting a complex river water management decision-support model:

- The model should have a history of extensive development/application thus providing opportunities to correct deficiencies and add improvements.
- A sound technical and institutional foundation is necessary to support a framework that provides continued future modelling improvements.

Based on the above, the selected river network model should include all or most of the following attributes:

- Model both water quality and quantity.
- Be suitable for planning and operational modelling.
- Be able to simulate a range of spatial and temporal scales.
- Support different water apportionment rules, including priority based and fractional allocation of river flow and capacity sharing in reservoirs.
- Include functionality for water accounting and auditing.
- Include functionality to support water trading.
- Be able to solve complex water demand and supply networks.
- Be able to do flow routing to account for flow lags and attenuations and to determine releases necessary to obtain a specified peak discharge at a downstream point.
- Be suited to linking with the *ACRU* model and other models on a time-step-by-time step basis.
- Be user friendly, to facilitate flexible configurations, preferably with integration of GIS functionality, and the ability to run scenarios.

Other considerations include the purchase and maintenance cost of the model, the potential to develop a working relationship with the model developers, the level of available user support by the model developers to model users, and a history of model development, application and refinement with continued support for modelling developments in the future.

After their review of river network models, Pott *et al.* (2008b) recommended that the RiverWare model (Zagona *et al.*, 2001) and the AWRIS (BoM, 2010b) should be prioritized for further investigation to assess their potential to meet the needs identified. These and other river network model developments are reviewed in Section 2.1.2.

### 2.1.2 River/reservoir network models

A river network model can be viewed at a simple level as a schematic linkage of rivers and reservoirs. This is illustrated in Figure 2.1 and Figure 2.2 where River Reaches 1, 2, 3 and 4 feed into River Reach 5.

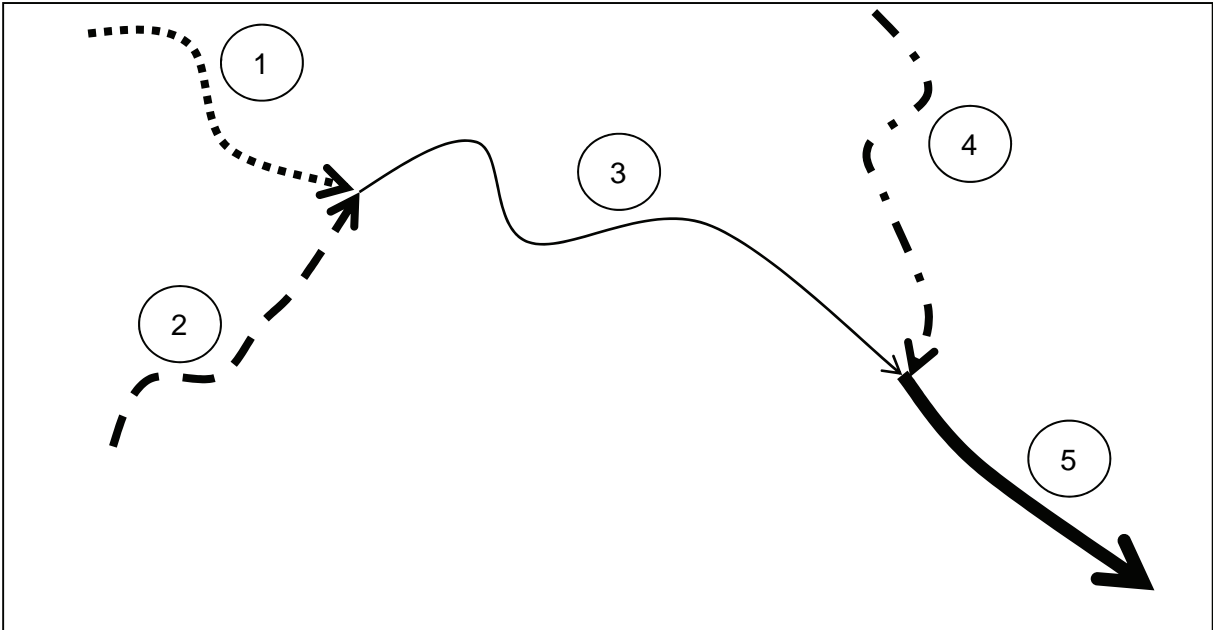


Figure 2.1 Simple river diagram

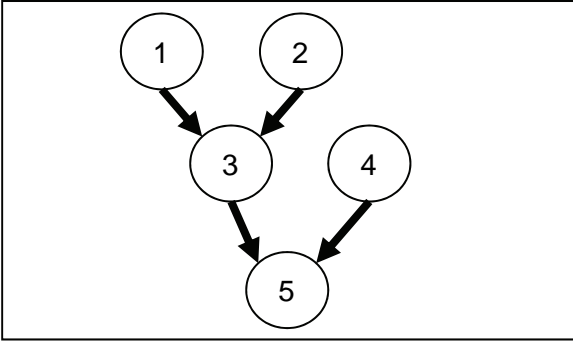


Figure 2.2 Simple river network diagram

In Figure 2.1 the assumption is made that there is one river reach per catchment thus the representation in Figure 2.2 could also be indicative of the catchment network.

River networks are in reality far more complex than the simple illustrations above. They are an integral part of catchment hydrology and include reservoirs, links to irrigation systems as well as other extractions, canals, wetlands, water user return flows and in some cases inputs from external systems such as inter-catchment water transfers. To manage the system effectively, a detailed understanding is required of the complex systems involved, including both the quantity and quality of water available at various locations within the system.

For a small catchment it may be assumed that all the stormflow generated from an event may reach the exit of the catchment on the same day. However, this may not be the case for large catchments (Smithers and Caldecott, 1995). Hence, the river network model is required to route the hydrographs through the system in order to simulate the lagging and attenuation of hydrographs as they move down the river network and through reservoirs.

In a river network model, streamflow from the subcatchments are input to the model. These may be observed, simulated or stochastically generated streamflow, which in the simulation case, requires that a rainfall-runoff model (RRM) simulates the streamflow which is linked to the river network model. Ideally the models need to be linked in such a way as to represent feedbacks between the two systems being modelled, for example, irrigation return flows.

The linking of a daily hydrological model with a river systems model can be accomplished by directly integrating the one model into the other or creating a new interface to link the two models. Frequently, the simulated output from the rainfall-runoff model is used as input to the river network model. The disadvantage of this serial-type link is that no feedback is possible between the two models (e.g. irrigation return flows) and ideally the two models should run in parallel on a time-step-by-time-step basis.

One approach to support integrated catchment management is the concept of using a Decision Support System (DSS) or a framework, which includes common general functionality (e.g. a common Graphical User Interface, common GIS tools, common analysis tools), but allows for multiple models to be incorporated within the framework. This approach requires the models that plug into the framework follow certain criteria and standards to facilitate the communication between the model and tools available within the framework. Hence, a requirement for the river network model selected for this project is that the model

must be able to link with other models. In the following sections a number of river network models are reviewed.

Wallbrink (2008) suggested the following reasons why models are useful for the management and/or operations of river systems:

- Resolution of competing demands for water resources and their associated trade-offs.
- Comparisons can be made between current water use and proposed future water use, including water allocation or sharing.
- Assist in policy formulation and land use permit decisions.
- The impact of climate change or variability and its possible effects on the water resources with a catchment.
- Better understanding of the river system.

Each of the following sections contains a brief overview of a model or system that was reviewed for this project.

### **2.1.2.1 AWRIS**

A literature review of the Australian Water Resource Information System (AWRIS) (BoM, 2010b) clearly indicates that an open supply of information for planning and management has been embraced. At the heart of AWRIS is an information system housed in a central database for the entire country that is managed by the Bureau of Meteorology (the Bureau). This information system is populated by water data collecting agencies that are, as a result of Water Regulations 2008, now required to supply specified data to the Bureau (BoM, 2010b). The standardization of the water information by the Bureau facilitates data accessibility by users, planners, water managers, stakeholders and policy-makers. The data is linked spatially via the Australian Hydrological Geospatial Fabric, referred to as Geofabric (BoM, 2010a), which can be queried and reports generated in various ways. Both observed data and modelled data are made available publicly which encourages both stake holder buy-in and transparency.

The Geofabric is a Hydrological Geodatabase developed as part of the AWRIS. It incorporates DEMs as well as hydrological networking information and facilitates setting up of models as well as access to data provided by the AWRIS system (BoM, 2010a).

The first phase of the AWRIS has been completed. One part of this is the water storage information for Australia that can now be accessed from:

<http://water.bom.gov.au/waterstorage/awris/index.html>. This web based tool gives an overview of the entire country’s water storage levels on a daily basis, or accumulate storages based on various spatial queries, for example by region or catchment. The user can drill down to a specific reservoir and view its current storage as well as changes in storage.

The AWRIS is being developed in phases with more being added over the next ten years (BoM, 2010b). This cutting edge development has vast resources being ploughed into it, both in expertise and in monetary terms, and is shown schematically in Figure 2.3.

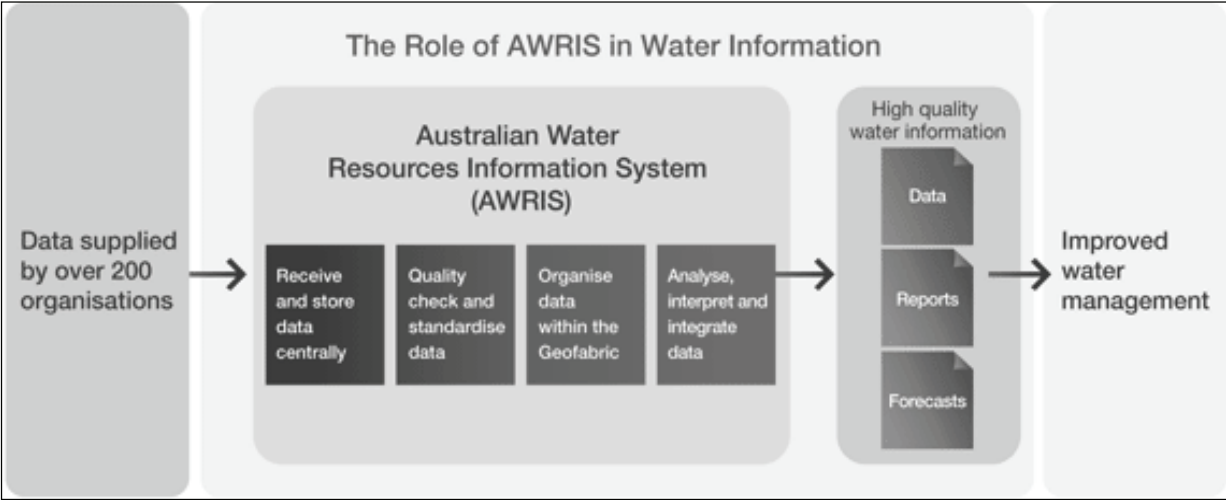


Figure 2.3 Schematic diagram of database and information processes in AWRIS (BoM, 2010b)

Thus AWRIS is more of an information system than a model and developments in eWater CRC Source, which is a part of the Australian system currently under development, are briefly reviewed in Section 2.1.2.3.

**2.1.2.2 BASINS**

The Better Assessment Science Integrating point and Nonpoint Sources (BASINS) (EPA, 2007) software system was first released in 1996 and has seen various improvements over the years with different version releases. The release of BASINS 4.0 has seen the most noteworthy changes with the move to an open source GIS system architecture providing a cost saving benefit (EPA, 2007). BASINS is used for analysing environmental systems to explore management alternatives at various spatial scales as it was developed to be used by both local and regional agencies. It enables both water quantity and quality catchment studies to be simulated and facilitates viewing data from a point or spatial scale (EPA, 2007). BASINS incorporates the WinHPSF and PLOAD models and also includes Parameter



Estimation (PEST) for WinHPSF and the Windows-based Climate Assessment Tool. PEST is a tool for automating the calibration process of WinHPSF and adds the ability to quantify uncertainty in the model predications (EPA, 2007). PLOAD is a lumped model for investigating water quality trends based on land-use (EPA, 2010).

A Data Download Tool and GIS enables access to observed data and other input information via links to the BASINS web site and other sources. BASINS also contains a GIS Project Builder, GIS Edit Tools, the ability to automatically or manually delineate catchment boundaries, a reporting facility to characterize catchments, a series of Surface Water Models, and customised databases. Software improvements include an automatic update feature to ensure that all the BASINS components are updated to the most current version (EPA, 2007).

Flow routing is performed within the models integrated into BASINS (EPA, 2010) and as such is not a feature made available from the BASINS framework itself. No facilities for water accounting or water ownership could be found as a result of a search through the BASINS electronic user manual (EPA, 2010).

### **2.1.2.3 eWater CRC Source**

The eWater Co-operative Research Centre (CRC) is developing a new National River Modelling Platform for Australia, along with partner organisations and additional funding from Australian Government agencies, to “manage and/or operate” river systems (Wallbrink, 2008). Wallbrink (2008) describes this platform as a flexible, extensible and component-based platform incorporating the “next generation of models” capable of taking into account the entire “water balance – from climate to runoff to river system to regulation and surface/groundwater interactions”. This flexibility enables the building of various systems based on the availability of data. The extensibility and component-based design facilitates the merging of models within the suite, which will range from the current uncertainty and risk models through to economic models in the future.

The eWater River Manager software, or ‘Source’ as it is referred to (Welsh, 2011), is currently under development and will replace the MSM-Bigmod, REALM and IQQM models for the Murray-Darling Basin (MDB) in Australia (Welsh and Podger, 2008; Welsh and Black, 2010). Welsh and Black (2010) refer to the REALM model as a monthly “generic optimisation model”, and the IQQM and Bigmod models as “generic daily simulation models”. Previously different models calibrated for specific subcatchments, and running at monthly or

daily time steps were used within large catchments in Australia, making it difficult to compare or accumulate outputs (Welsh and Podger, 2008). Source Catchments uses a common daily time step to overcome this problem in the MDB (Wallbrink, 2008). It is envisaged that the platform will provide a more consistent method for calibration of hydrological models and facilitate stakeholder involvement and is intended to aid in trade-off decisions, water trading, regulations and water accounting in a transparent process that can be repeated (Wallbrink, 2008).

Wallbrink (2008) states that the modelling platform puts Australia “at the forefront of model building internationally” and that once their vision is achieved their models may have great potential for application internationally where contention over water resources is high, for example, in Southern Africa. The Source Rivers component is currently under development. However, a beta version could possibly be obtained for evaluation on request (Miller, 2011).

Delgado *et al.* (2011), in the draft Source User Guide which is currently under development, describes Source as “an application for describing and modelling the behaviour of river systems. Source is designed to support the construction and operation of river models that mimic river behaviour over arbitrarily-long periods (days, weeks, months, years, centuries)”.

Source has the facilities to track water ownership through a system enabling queries to be made that determine how much water and where in the system it is located for a particular water owner. A river system can therefore be sub-divided based on the water owner which adds management functionality which could facilitate water trading, in both storages and links and restrictions for water owners. The development of separate resource allocation systems for water owners facilitates capacity sharing, continuous sharing and water accounting. Optimisation algorithms, scenarios and economic analysis are some of the features of Source (Delgado *et al.*, 2011). The base code is currently being made OpenMI compliant (Welsh, 2011). eWater CRC’s Source is made up of Source Catchments, Source Rivers and Source Urban (eWater, 2010a).

Source Catchments has a user friendly GUI and facilitates setting up various scenarios with the aid of a Scenario Wizard. The concept of a Functional Unit (FU) is used within Source Catchments, where FUs are defined as “areas of similar hydrological behaviour or response (e.g. land use)” (eWater, 2010b). A FU is equivalent to a Hydrological Response Unit (HRU). Each FU can have a different Rainfall-Runoff Models (RRM) assigned to it, or a particular RRM can be assigned to all the FUs in a particular subcatchment, or assigned based on FU

type. Currently the RRM's available for selection in Source Catchments include: AWBM, Sacramento, SIMHYD, and SURM (eWater, 2010b).

The Source Catchments component is currently free for the first 12 months with an annual renewal fee of (AUD) \$ 990 and is available from their web site (eWater, 2010a).

#### **2.1.2.4 HEC-ResSim**

The Hydrologic Engineering Centre (HEC) is a division of the Institute for Water Resources (IWR) of the U.S. Army Corps of Engineers. The Reservoir System Simulation (HEC-ResSim) model is a part of the HEC Next Generation (NexGen) Software Development Project and replaces HEC-5 (HEC-5, 1998). It is intended to simulate reservoir operation and has a Graphical User Interface (GUI), makes use of the HEC Data Storage System (HEC-DSS), and has data management capabilities and reporting features (Wurbs, 2005; HEC-ResSim, 2010).

HEC-ResSim is suited to planning studies and for modelling flood control operations by reservoir control personnel. Specifically coded algorithms are used to simulate multi-purpose, multi-reservoir systems with user selected time-steps, which may range from 15 minutes to one day. Various routing options are provided (Wurbs, 2005).

HEC-ResSim consists of three modules: catchment configuration, river network and simulation (HEC-ResSim, 2010). The catchment configuration module provides a common framework for catchment configuration and definition which can be used by different HEC-based modelling applications.

The river network module builds a schematic network describing the physical and operational elements of the system, including alternative scenarios, and can be used to create and edit elements on a river reach. The simulation module configures the simulation, performs the computations and enables viewing of the results. Seven different routing methods, which are mostly Muskingum-based methods, are available to route the flows in the river reaches. (HEC-ResSim, 2010).

#### **2.1.2.5 MIKE BASIN**

The MIKE BASIN model is river network simulation model where rivers are represented by a network of branches and nodes and a mass balance is performed for each accounting step

(DHI, 2010). According to Ershadi *et al.* (2005), the philosophy behind MIKE BASIN is to keep the modelling simple and intuitive while still providing comprehensive planning and management insight. The model is an extension to and runs from within the ESRI ArcView GIS environment (Wurbs, 2005) via a user-friendly graphical user interface, making the model a potentially powerful tool for enhancing communication between stakeholders involved in water management.

MIKE BASIN incorporates the physical layout of the river basin and the water resource system infrastructure, enables naturalised streamflow to be specified for incremental catchments, the specification of different water user schemes including their water demand, and water resource infrastructure and management operations.

Although a monthly time step is commonly used for MIKE BASIN applications, the time step can be specified by the user if a specific need is encountered (Wurbs, 2005). Reservoirs and abstraction points can be used to control the allocation of water through the specification of a set of rules which are capable of simulating riparian rights or prior rights systems (Wurbs, 2005). This is facilitated by various water sharing and allocation algorithms (DHI, 2010). When available water quantities do not meet the requirements from multiple stakeholders, the allocation is solved based on priority levels that can be set at a local (i.e. node) or global scale (Christensen, 2004).

Pott *et al.* (2008a) showed that the *ACRU* model has been successfully linked in series with MIKE BASIN via the AAMG development in the Oliphant's River and the Mhlathuze River catchments. Kime (2010) also used a combination of *ACRU* and MIKE BASIN to develop a water accounting and auditing system to track water ownership, but encountered limitations to do this in MIKE BASIN.

#### **2.1.2.6 MODSIM**

MODSIM is a comprehensive, generalized river basin management decision support system which has been under continuous development and enhancement since 1979 at Colorado State University (Labadie, 2005). The Bureau of Reclamation and various other entities have sponsored studies in which MODSIM has been applied. MODSIM is based on object-oriented programming and represents a number of water allocation models based on network flow programming, including the early Texas Water Development Board models among others (Wurbs, 2005).

Wurbs (2005) describes MODSIM as a “general-purpose reservoir/river system simulation model based on network flow programming designed for analysing physical, hydrologic, and institutional/administrative aspects of river basin management”. Long-term planning is supported with the use of monthly time steps, medium-term management by using weekly time steps and short-term operations by the use of daily time steps. User-specified relative priorities are used to allocate water to meet diversion, instream flow, hydroelectric power, and storage targets, as well as lower and upper bounds on flows and storages.

Water quality simulation and the joint use of ground and surface water are optional capabilities (Labadie, 2006a). Allocation decisions are not affected by either future inflows or future releases as a network flow programming solver is used for each individual time interval (Wurbs, 2005).

The river/reservoir system topology is built by clicking and dragging icons representing various hydrologic nodes via the graphical user interface (Wurbs, 2005; Labadie, 2006a). Raster image files may be imported as background layers over which the network nodes etc. can be spatially laid out (Labadie, 2006). A data management system controls data structures embodied in each model object and this facilitates data input and queries. Data files are prepared interactively and time series data can be either imported from Microsoft Access databases, Excel spreadsheets, comma separated value ASCII files, copied and pasted, or manually input. The network flow optimization can be automatically executed via the graphical user interface. The results from the simulations performed are output to graphs and there is also the facility to produce customized reports with the aid of the Custom Code Editor within MODSIM (Labadie, 2005; Wurbs, 2005; Triana and Labadie, 2007).

Earlier versions of MODSIM incorporated the PERL scripting language for customization (Wurbs, 2005). This has since been replaced by a Custom Code Editor that makes use of Microsoft .NET Framework compatibility and allows customised code to be written in Visual Basic.NET or C#.NET. The Custom Code Editor enables the customised code to be compiled into machine code which has improved the runtime performance (Labadie, 2006a).

Stream inflows and reservoir evaporation rates are required as input and monthly, weekly, or daily time steps may be used in the simulation. A lag methodology is for routing daily streamflow with built in calibration procedures provided for computing local streamflow gains and lag parameters (Wurbs, 2005). When simulating at a daily time-step, hydrologic streamflow routing can be achieved either via the Muskingum method or by user-specified time-lagging. Routing can be applied to any link between nodes (Labadie, 2006a).

There are three types of nodes which may be defined. These are non-storage nodes (e.g. river gauges, diversion dams, tributary confluences, and sites where return flows enter the river), demand nodes (e.g. consumptive diversions or instream flow requirements), and reservoir nodes. The types of links used to connect nodes include artificial, natural flow, general flow, storage ownership, and accrual and a number of constructs are available for modelling complex water allocation schemes (Wurbs, 2005).

The configuration of the river/reservoir system is stored in an ASCII data file which includes all the information about the time series data and the physical features of the river system. The data file is command-value oriented, with each line of the input starting with a command that the input parsing code associates with a model construct. Data values relevant to the modelled feature follow the command. The model user may also specify system constraints through the Custom Code Editor (Wurbs, 2005).

According to Labadie (2006a) the customisation features of MODSIM were effectively used in an economic study in the San Joaquin River catchment. Complex water pricing structures such as tiered water pricing, as well as increased water prices, were modelled along with changes in reservoir operations and environmental flows to improve water management (Labadie, 2006a).

A simplified groundwater component and limited water quality modelling options are available in MODSIM and the model has also been linked with the U.S. Geological Survey MODFLOW groundwater model (Wurbs, 2005).

Network flow programming is used to simulate complex water management and allocation systems. User input cost coefficients and constraints for each prioritised link are used in the objective function and an optimisation algorithm minimises the objective function while meeting all constraints for a single time step. For each time step, nonlinear aspects are dealt with in a sequential process where the computations are repeated in an iterative manner (Labadie, 2006a).

Water allocation is based on the priorities assigned by the user to storage and hydroelectric power targets, lower and upper bounds on both storages and flows, diversions and in-stream environmental flows (Labadie, 2006a). Other features, according to Labadie (2006a), include:

- rent pools,

- water banking,
- flow augmentation plans,
- exchanges that allow flexible system operations (still maintaining water rights and contract legality), and
- flood control.

GEO-MODSIM is an ArcView extension that enables MODSIM to be incorporated within an ArcGIS environment which facilitates the use of GIS tools to configure the networks and run other spatial tools (Triana and Labadie, 2007). According to Labadie (2011), the GEO-MODSIM extension should be publicly available by August 2011. At this stage it has not been established if there is a cost for this extension, but MOSIM Version 8.1 is available at no cost from the following URL: <http://modsim.engr.colostate.edu/index.shtml>

A recent application that makes use of MODSIM and the GEO-MODSIM is River GeoDSS (Triana *et al.*, 2010). Triana *et al.* (2010) used Artificial Neural Networks (ANNs) to model complex stream-aquifer interactions in the Lower Arkansas River catchment. According to Triana *et al.* (2010) there are several benefits of this system over network models such as RiverWare, RIBASIM, MIKE BASIN, IQQM and WEAP. These include:

- the incorporation into a fully functional GIS,
- a data-centred structure where the database management system is the fundamental component of the multi-application environment,
- extensive customisation tools allowing direct compilation of custom code into machine language, rather than the use of inefficient scripting languages,
- a highly efficient network flow optimization solver allowing applications for large-scale systems,
- inclusion of time-lagged routing of surface flows and stream-aquifer interactions,
- applicability to problems ranging from real-time operations to strategic planning,
- ideally suited for evaluating complex administrative rules and legal issues related to water and storage rights for priority-based water allocation, and the
- option to seamlessly link “more realistic”, well-calibrated models, rather than being confined only to internal modules.

#### **2.1.2.7 REALM**

The REsource ALlocation Model (REALM), developed in Australia, is described by Perera *et al.* (2005) as a generalised simulation model that simulates the accumulation and allocation of water resources within a water supply system. According to Schreidera *et al.* (2003),

REALM is used widely as a water allocation management tool in Australia. Perera *et al.* (2005) describe REALM as a modelling tool that can be applied to create specific water allocation models. This is achieved through a host of supply systems and operating options.

Optimisation of water allocation in REALM is achieved by means of a network linear programming algorithm which accounts for penalties defined by the user at each time step. Mass-balance accounting at nodes are used and constraints on carriers, such as rivers channels or pipe carriers, control water movement. Transmission losses can also be accounted for by the carriers (Perera *et al.*, 2005).

According to Perera *et al.* (2005) REALM attempts to satisfy:

- evaporation losses in reservoirs,
- transmission losses in carriers,
- all demands (which may be restricted), to maximize reliability,
- minimise spill from the system in order to maximise yield, and
- minimum flow requirements and ensure that reservoir storage targets are met at the end of the season.

REALM in the process of being superseded by eWater's Source (Welsh, 2011).

#### **2.1.2.8 RIBASIM**

Wurbs (2005) and Assaf *et al.* (2008) report that the River Basin Simulation Model (RIBASIM) model links water inputs from various locations in a catchment with specific water users. A water balance is computed for the reservoir/river/use system to determine water availability and to evaluate a variety of measures related to infrastructure and operational and demand management. The simulated series may be linked to water quality and sedimentation analyses for river reaches and reservoirs and the composition of the flow can be interrogated. The user-friendly graphical interface of the RIBASIM model provides guidance on the design, simulation and the analysis phase, and supports the user's choice of GIS environment to configure the model, to enter object attribute information and to evaluate the simulation results (Wurbs, 2005; Deltares, 2010).

RIBASIM is described as "a comprehensive and flexible tool which links the hydrological water inputs at various locations with the specific water-users in the basin" and is used for river and catchment planning and management (Deltares, 2010). RIBASIM encapsulates a structured approach to catchment planning and management and enables evaluation of



distribution patterns of both water quantity and water quality measures related to infrastructure, operational and demand management in river reaches and reservoirs. It enables source analysis and water auditing to be performed at a point in a catchment. RIBASIM can be used for:

- long-term basin planning with time horizons ranging from 10 to 25 years,
- seasonal operational planning and short term (< 1 year) water allocation scheduling, and
- within season operational scheduling using both actual (observed) and expected (forecast) rainfall and allocation schedules (Deltares, 2010).

The model has been applied for more than 20 years internationally on both large, complex catchments which include independently simulated subcatchments which are combined into a single larger catchment simulation for catchment wide planning and management (Deltares, 2010). For example, Schellekens *et al.* (2003) assessed the feasibility of linking the RIBASIM network based water allocation model of a river with a raster based crop water balance model using the PC-Raster software package, which is a raster based dynamic modelling language, and concluded that the software package was not ready to create the on-line linkage. Nadomba *et al.* (2005) used RIBASIM to for water distribution in the development of GIS-based modelling tools and methods for sustainable and integrated management of water resources in the Nile Catchment. Tollenaar (2009) integrated a rainfall-runoff model using a series link with RIBASIM configured for the NILE catchment (referred to as RIBASIM-NILE) in order to investigate current and future discharges. According to van der Krogt (2011), the ability to link RIBASIM to other models on a time step (parallel) basis would require modification of the code by developers.

RIBASIM can simulate concentrations of substances in river reaches and reservoirs and can be linked to the HYMOS hydrological database and modelling system and to the DELWAQ water quality model to simulate detailed water quality processes. However, a user defined number of substances can be specified, e.g. salt, Biological Oxygen Demand, Nitrogen, Phosphorus, Bacteria and toxic substances (Deltares, 2010).

According to Deltares (2010), RIBASIM simulates the water balance of a catchment and computes the composition of flow at every location and at any time in the catchment. Drainage from agriculture, discharge from industry and re-use of water downstream can all be accommodated in the model. The flow routing options in RIBASIM (e.g. Manning formula, flow-level relation, 2-layered multi segmented Muskingum formula, Puls method and Laurenson non-linear “lag and route” method), which are executed on a daily basis started

on a user selected day and for any forecasted period, have been used within an early flood warning system.

RIBASIM includes a range of water management and water allocation features which include the following (Deltares, 2010):

- Both water allocation and source allocation priority per individual user.
- Operation rules for individual reservoirs and groups of reservoirs, groundwater management rules.
- Water allocation based the simulated target demands and target releases.
- Proportional allocation of water.

The current version of RIBASIM allocates water only on a priority basis and cannot perform capacity sharing of reservoirs or fractional allocation of streamflow between users. Additional coding will be necessary to implement this functionality (van der Krogt, 2011).

Various tools from the Delft Tools library may be used in conjunction with RIBASIM. These include (Deltares, 2010):

- Case Management Tool (CMT) enables a visualisation of the workflow diagram.
- Delft-GIS (Netter) software provides catchment analysis not available in conventional GIS.
- Ods\_View is used to export and present time series data.
- The Case Analysis Tool (CAT) is used to compare and evaluate simulation results.

Simulation of reservoir operation is strong feature of RIBASIM and includes (Deltares, 2010):

- The simulation of the water balance of reservoir in series or parallel including rainfall input, observed or expected inflow, evaporation, seepage losses and releases.
- The simulation of the hydraulic characteristics of the reservoir gates, sluices and turbines.
- Rule curves for flood control, maximum energy production, firm storage, zoning of the reservoir storage and hedging (water rationing) of target releases.
- Demand driven (target release) or supply orientated (storage controlled) operations and specific operation based on level control.
- The simulation of hydro-power station characteristics and power generation potential.
- The simulation of firm energy demand per time step taking catchment level water allocation into account.

RIBASIM includes various methods for simulating demand for water from agriculture, including input of the gross demand as well as the use of the DelftAGRI to simulate agricultural water demand, water allocation, crop yield and production costs (Deltares, 2010). An interactive graphical tool in RIBASIM enables the simulation of combinations of cultivars for specified areas planted and planting dates (Deltares, 2010).

The water balance of a groundwater aquifer, and thus the groundwater management options and the conjunctive use of surface (river and reservoir) and groundwater, can be simulated using RIBASIM (Deltares, 2010).

Assaf *et al.* (2008) conclude that RIBASIM is relatively easy to use but requires significant data for detailed analysis. A limited version of RIBASIM is freely available and the full model and documentation can be obtained at a “relatively low cost” (Assaf *et al.*, 2008). Currently for Version 7.00 of RIBASIM, the cost is 10 000 Euro with a 50 % discount for educational institutions and annual maintenance costs are 25 % of the purchase cost (van der Krogt, 2011). The developers of RIBASIM have improved their model over time, but recognise that the model still requires continuous modification and/or further extension and hence is always in a state of development (Assaf *et al.*, 2008). The source code for RIBASIM is currently not distributed to users (van der Krogt, 2011).

#### **2.1.2.9 RiverWare**

RiverWare has been developed at the University of Colorado Centre for Advanced Decision Support for Water and Environmental Systems (CADSWES) along with collaborative research and development from the Tennessee Valley Authority, the U.S. Bureau of Reclamation and the U.S. Army Corps of Engineers (Zagona *et al.*, 2001). RiverWare is a general river and reservoir modelling tool for planning, forecasting, operational scheduling, policy evaluation and water accounting (Zagona *et al.*, 2001). Perera *et al.* (2005) categorised RiverWare as a heuristic procedure guided by objectives defined by the user.

RiverWare focuses on river/reservoir systems and includes allocation of water for the environment, recreation, agriculture, hydropower, navigation, flood control and flood prevention, and also simulates water quality constituents. The model has an object oriented modelling approach. Features, represented as a type of object, contain data (slots) as well as the algorithms (methods) specific to that object type. The slots are made up of various data structures capable of storing, for example, data tables and time-series. The water ownership information is also contained within the feature objects (Zagona *et al.*, 2001).

RiverWare can accommodate various time-steps from hourly through to annual to facilitate multiple purposes, for example monthly to annual time-steps may be used for planning, or hourly to daily may be used for scheduling (Zagona *et al.*, 2001; Wurbs, 2005). Input data can either be forecast, or historical hydrology, or stochastically generated sequences. The model can then interact with multi-objective operating policies to produce output to aid in predictions, operating decisions and trade-offs in environmental and economic analysis.

Water Accounting is handled by four account types (Zagona *et al.*, 2010):

- storage,
- diversion,
- instream flow, and
- pass-through.

A number of different river reach routing methods are available in RiverWare (Zagona *et al.*, 2010). These include Time Lag, Impulse Response, Muskingum, Muskingum-Cunge, Kinematic Wave, and Storage Routing methods.

Streamflow is input at river nodes and RiverWare models the volume balances at reservoirs, hydrologic routing in river reaches, evaporation and other losses, diversions, and return flows. Groundwater interactions, water quality, and electric power economics may also be computed. Any number of reservoirs and stream reaches can be modelled (Wurbs, 2005).

Software tools are provided for constructing a model for a particular reservoir/river system and then running the model. These include a library of modelling algorithms, several solvers, and a language for coding operating policies. The tools are applied within a point-and-click graphical user interface (Wurbs, 2005).

A palette of object icons representing features of a river basin is provided, as listed in Table 2.1. Objects have slots, which contain variables and parameters associated with the physical process models and each object models one or more basic physical processes. The objects are linked to form the river network. The user selects objects by dragging icons from the palette to the workspace and customises each object by naming it, selecting computational options, and adding data (Wurbs, 2005).

Table 2.1 RiverWare objects (after Wurbs, 2005)

Object Type	Processes Modelled
Storage reservoir	Mass balance, evaporation, bank storage, spill, water quality
Level power reservoir	Storage reservoir plus hydropower, energy, tailwater, operating head
Sloped power reservoir	Level power reservoir plus wedge storage for long reservoirs
Pumped storage reservoir	Level power reservoir plus pumped inflow from another reservoir
Reach	Routing in a river reach, diversion and return flows
Aggregate reach	Many reach objects aggregated to save space on the workspace
Confluence	Brings together two inflows to a single outflow as in a river confluence
Canal	Bi-directional flow in a canal between two reservoirs
Diversion	Diversion structure with gravity or pumped diversion
Water user	Depletion and return flow from a user of water
Aggregate water user	Multiple water users supplied by a diversion from a reach or reservoir
Groundwater storage	Stores water from return flows
River gage	Specified flows imposed at a river node
Thermal object	Economics of thermal power system and value of hydropower
Data object	User-specified data for policy statements and post-processing
Bifurcation	Flow junction with single inflow and two outflows
Inline power	Run-of-river power production
Control point	Object used to regulate upstream reservoirs based on channel capacity

Data may input to RiverWare manually through the graphical user interface, by loading data files, or entered through the data management interface, which facilitates retrieving large datasets through an external program. Tabular and graphical displays of the model results are output. Time series associated with the various objects, such as reservoir and reach outflows, water quality and reservoir storages, elevations, and other water accounting data can be output (Wurbs, 2005).

Water ownership accounting and water quality computations are possible with the simulation and rule-based approaches while operational rules are used to solve for the rule-based simulation and optimisation approaches (Wurbs, 2005). The model has multiple solution methodologies (Zagona *et al.*, 2001). These include the following approaches:

- standard simulation mode,
- rule-based simulation, with operating policies or rules, and
- optimization, implemented through a linear, pre-emptive goal programming method.

Pure simulation can be performed when each object has the information required to “dispatch” the method/algorithm. Data are input to “slots” on the objects, either directly by

users or by propagation from other objects. The methods associated with an object may also set required values (Wurbs, 2005).

Based upon the data provided, the appropriate dispatch method is executed and the simulation is performed. When required, solution results propagate to other objects as appropriate and multiple links between objects may necessitate iterative solutions. When conflicting information results in an error state, the simulation is terminated. Parts of the model are not solved if not enough information is provided (Wurbs, 2005).

A rule language provides flexibility in expressing reservoir/river system operating rules and, in rule-based simulation, enables a solution when there is not enough information associated with the objects to obtain a solution. The user specified prioritised policy statements (rules) are interpreted by the rule processor to provide the additional information required for the solution to proceed. Slot values for the objects are set based on these rules and the state of the system. The rules are if-then constructs that examine the state of the system as functions of values of slots on the objects in the if-clause. Values are then set depending on that state. The rules are formulated by the model-user in the RiverWare rule language and entered through a graphical editor (Wurbs, 2005).

RiverWare combines a linear programming (LP) solver with pre-emptive goal programming in the optimization solution approach. The use of an optimization constraint editor and expression language in RiverWare enable users who are not proficient in LP to provide the required input information. Objectives and constraints are expressed in terms of physical variables such as pool elevation, flows, or spills or in terms of economic variables such as net replacement cost, future value of used energy, spill cost, and the cost of alternative power sources (Wurbs, 2005).

According to (Biddle, 2001), RiverWare has been effectively used for operational planning in the Tennessee Valley Authority System.

#### **2.1.2.10 WEAP**

Primary support for the development of the Water Evaluation And Planning (WEAP) model (SEI, 2011) came from the Stockholm Environment Institute (SEI). The Hydrologic Engineering Center of the US Army Corps of Engineers has been responsible for significant enhancements and numerous agencies have provided project support. WEAP has been applied in a number of countries for water assessments (Wurbs, 2005; SEI, 2011).

Wurbs (2005) describes WEAP as a reservoir/river/use system water balance accounting model where demands for water are met from surface and groundwater sources. WEAP maintains water balance databases, generates water management scenarios, and performs policy analyses.

Examples of the application of WEAP in South Africa can be found in Lévite *et al.* (2002) and McCartney and Arranz (2009). McCartney and Arranz (2009) used WEAP at a monthly time step to investigate various water demand scenarios in the Olifants River Catchment. As part of these scenarios they investigated possible future water demands and various strategies that could be implemented and the economic implications thereof. Lévite *et al.* (2002) describe the application of WEAP in the Steelpoort catchment, which is a subcatchment of the Olifants River Catchment. Various scenarios of water allocation were considered with an emphasis on management of water demand. The usefulness of WEAP as a tool to stimulate discussion and the interaction of stakeholders was noted by Lévite *et al.* (2002).

#### **2.1.2.11 WRAP**

The Water Rights Analysis Package (WRAP), developed as a suite of Fortran programs, simulates the management of the water resources of a river basin or multiple-basin region under a priority-based water allocation system. Long-term monthly time step modelling is performed to assess hydrologic and institutional water availability and reliability for water supply diversions, environmental instream flow requirements, hydroelectric energy generation, and reservoir storage, with an updated version being able to simulate at daily or other sub-monthly time steps. An interface (WinWRAP) for executing the programs within Microsoft Windows has been developed (Wurbs, 2005).

Inputs to WRAP are monthly naturalized streamflows and reservoir net evaporation less rainfall depths. Routines are provided to facilitate the estimation of naturalized streamflows and reservoir net evaporation. Options are available to distribute flows from gauged to ungauged sites. An adaptation of the Muskingum method is used for routing streamflow, and routines for calibrating routing parameters are provided (Wurbs, 2005).

Simulation modes include a single long-term simulation, automatic repetition of the simulation with adjustments to specified targets to develop a yield-reliability table that ends with the firm yield, and conditional reliability modelling based on many short-term simulations starting with the same initial storage condition (Wurbs, 2005).

### 2.1.3 Discussion and recommendation

The information in Table 2.2 is compiled from the above overviews of the characteristics of the various river network models. Included in Table 2.2 is summary of the model requirements discussed in Section 2.1.1 and, where sufficient information has been sourced, the characteristics of the selected models are assessed against these requirements.

From the summary provided in Table 2.2, the least suitable river network models to meet the listed attributes are HEC-ResSim, WRAP and RIBASIM. The main limitation of RIBASIM is the limitation to perform priority-based allocation only. The developments within the eWater CRC Source framework appear to hold significant potential, but the code is still under development and currently only some beta versions of the software are available on request. Not all components are functional as yet and this could affect the suitability of this suite of models for use in this project.

The RiverWare, MIKE BASIN, eWater CRC Source and MODSIM models were all found to have many of the required attributes. One known limitation of the current version of MIKE BASIN is that it cannot adequately perform all the functions for water auditing and accounting. An advantage of MIKE BASIN is the availability of support for the model in South Africa. Given the information gained from the literature, it thus appears that the RiverWare and MODSIM models have the largest potential to meet all the required attributes and functionality. Added advantages of MODSIM are that the model is available at no cost and that the MODSIM compiled libraries are provided for access by third party software. Although the River component of Source by eWater CRC appeared to meet the requirements for the project it was still under development and only a beta version was available at the time of the review (Miller, 2011), and the help documentation for the river component was also still being developed (Delgado *et al.*, 2011). Based on this review it was recommended that further evaluation of the MIKE BASIN, MODSIM and RiverWare models needed to be performed to facilitate selecting one model for use in the project.



Table 2.2 Summary of selected models

Model		Framework									
Attribute	RiverWare	MIKE BASIN	HEC-ResSim	MODSIM	WRAP	RIBASIM	eWater Source				
Descriptive Name	River and Reservoir Operations	GIS-Based Support for Planning & Management	Reservoir System Simulation	Generalized River Basin Network Flow Model	Water Rights Analysis Package	River Basin Simulation	Source (Beta version available in 2010)				
Model Development Organization	Center for Advanced Decision Support for Water and Environmental Systems (CADSWES), University of Colorado http://cadswes.colorado.edu/riverwar	Danish Hydraulic Institute http://www.dhisoftware.com/	USACE Hydrologic Engineering Center http://www.hec.usace.army.mil/	Colorado State University http://modsim.engr.colostate.edu/	Texas Commission on Environmental Quality, USA	Delft Hydraulics, http://www.wfdelft.nl/soft/ribasim/	eWater Cooperative Research Centre http://www.ewater.com.au/products/ewater-source/				
Planning	Yes	Yes	Yes	Yes	Yes	Yes	Yes				
Operational	Yes	Yes	Yes	Yes	Yes	Yes	Yes				
Model time step	1h to -1 year	User specified	15 min to 1 day	Month, week, 10, 5 & 1day	Daily to monthly	Daily	Daily to monthly				
Range of spatial scales	Yes	Yes	Yes	Yes	Yes?	Yes	Yes				
Flexible configuration	Yes	Yes	Yes	Yes	Yes?	Yes	Yes?				
GIS functionality	No	Yes	Yes	Yes, with GeoMODSIM	?	Yes	Yes				
User friendly	Yes	Yes	?	Yes	Yes?	Yes	Yes				
GUI	Yes	Yes	Yes	Yes	Yes	Yes	Yes				
Water quantity	Yes	Yes	Yes	Yes, with link to MODFLOW for groundwater simulation	Yes	Yes	Yes, contains other models (e.g. AWBM, SIMHYDC.)				
Flow routing	Multiple methods	Yes	?	Muskingum or user specified lagging	Muskingum-based	Multiple methods	Multiple methods				
Water quality	Salinity, dissolved solids, temperature, and dissolved oxygen	?	?	Salinity (QUAL2E)	Salinity	Yes, by link to DELWAQ	Yes				
Simulate different water allocation methods	Yes	Yes	?	Yes?	Priority based allocation only	Yes	Yes?				
Water accounting & auditing	Yes	Not able to track water	?	Yes	?	?	?				

Table 2.2 (continued) Summary of selected models

Attribute	Model				Framework			
	RiverWare	MIKE BASIN	HEC-ResSim	MODSIM	WRAP	RIBASIM	eWater Source	
Forecasting	Yes	Yes	?	Yes	?	Yes	Yes?	
Water trading	?	?	?	?	?	?	?	
Complex water supply and demand simulations and optimisation	Yes, includes customised rules for objects with pseudo code editor	Yes	Yes?	Yes, includes Custom Code Editor	?	Yes?	Yes?	
Scenarios	Yes (Under development)	?	?	Yes	?	No	Yes	
Model link protocol	Propriety	OpenMI?	?	?	?	?	Open MI?	
Potential to link to ACRU on daily time step basis	Yes	Yes (AAMG), but linked in series	?	Yes	?	No, would require code modification	Yes	
Potential for collaboration with model developers	Yes	Yes	?	Yes, requires contractual arrangement	?	?	?	
Organizing Computational Structure	Object-oriented, options for pure and rule-based simulation and optimization	Object-orientated	Object-oriented simulation progressing from upstream to downstream	Object-oriented network flow planning	Ad hoc simulation progressing in order of user-defined priorities	?	?	
Model user support	Yes (US)	Yes (SA)	?	Yes	?	?	Yes (Australia)	
User base	Yes	Yes	Yes	Yes	?	Yes	Code under development	
Access to source code	No	No	No	Access to compiled libraries	No	No	No	
Cost (2010)	Single node license: \$ 300 Annual renewal fee: Research \$ 900 Government/Commercial \$ 3000	Basic MB: € 6000 MB WQ: € 3000 Extended: € 8500	Free	Free	Free	Purchase: € 10000 (50% discount for educational institutions) Annual maintenance: 25% of the purchase cost	Free for first 12 months Annual renewal fee (Au) \$ 990	

## 2.2 Detailed Evaluation of Selected Models

The aim of the following evaluation of the MIKE BASIN, MODSIM and RiverWare models was to configure these models and evaluate them based on a set of criteria. This provided a better understanding of how these models work and enabled the capabilities of these models, as stated in the literature, to be confirmed. This evaluation also intends to test two allocation methods *vis.* priority allocation and Fractional Water Allocation And Capacity Sharing (FWACS), as described by (Lecler, 2004).

The methodology used for the testing of river network models was to create a hypothetical catchment with various water users. The hypothetical test catchment is made up of four subcatchments and various water users as shown in Figure 2.4. As far as possible the same hypothetical catchment was configured for each of the network models and then evaluated against a list of criteria (*cf.* Sections 2.2.1.1 to 2.2.1.7). For each of the network models the hypothetical catchment was configured to allocate water using two methods, (i) water user priority, and (ii) FWACS. No evaporation, transfer, or seepage losses were modelled as part of this evaluation.

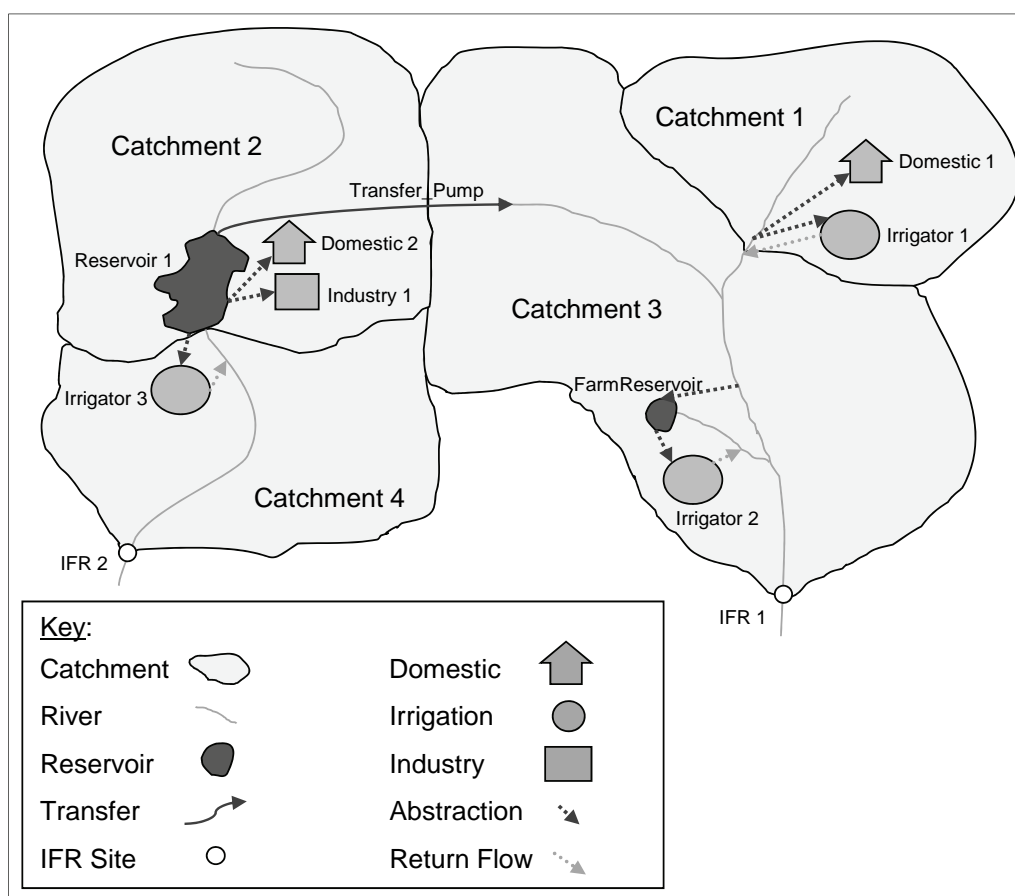


Figure 2.4 Schematic of hypothetical test catchment

Catchments 1 and 2 are runoff generating catchments, but for the purpose of simplifying the tracking of runoff from the upstream catchments for this evaluation, Catchments 3 and 4 are assumed not to generate any runoff. The details of the runoff generating catchments are contained in Table 2.3. The average monthly flow sequence specified in the *Catchrun* file is shown in Table 2.4 and was used to generate the runoff from Catchment 1 and Catchment 2 based on their catchment area and applied as daily inflow into the river in Catchment 1 and into the reservoir in Catchment 2.

Table 2.3 Catchment details

Catchment	Area (km <sup>2</sup> )	Flow sequence (time series file)
Catchment 1	100	<i>Catchrun</i>
Catchment 2	200	<i>Catchrun</i>

Table 2.4 Sequence of average monthly flows in the *Catchrun* file

Time	Specific runoff (l/s/km <sup>2</sup> )
1981/01/01	100
1981/02/01	100
1981/03/01	70
1981/04/01	60
1981/05/01	50
1981/06/01	40
1981/07/01	20
1981/08/01	10
1981/09/01	50
1981/10/01	70
1981/11/01	90
1981/12/01	100

The hypothetical test catchment includes two reservoirs; Reservoir 1 situated at the outlet of Catchment 2, and FarmReservoir which is a small off-channel farm reservoir within Catchment 3. The reservoir level characteristics assumed for these reservoirs are summarised in Table 2.5. The level-area-volume relationships are shown in Figure 2.5 for FarmReservoir and in Figure 2.6 for Reservoir 1.

Table 2.5 Reservoir level characteristics

Reservoir	Bottom level (m)	Top of dead storage (m)	Reservoir crest level (m)	Flood control level (m)	Initial level (m)
Reservoir 1	534	536	545	543	540
FarmReservoir	534	535	545	545	540

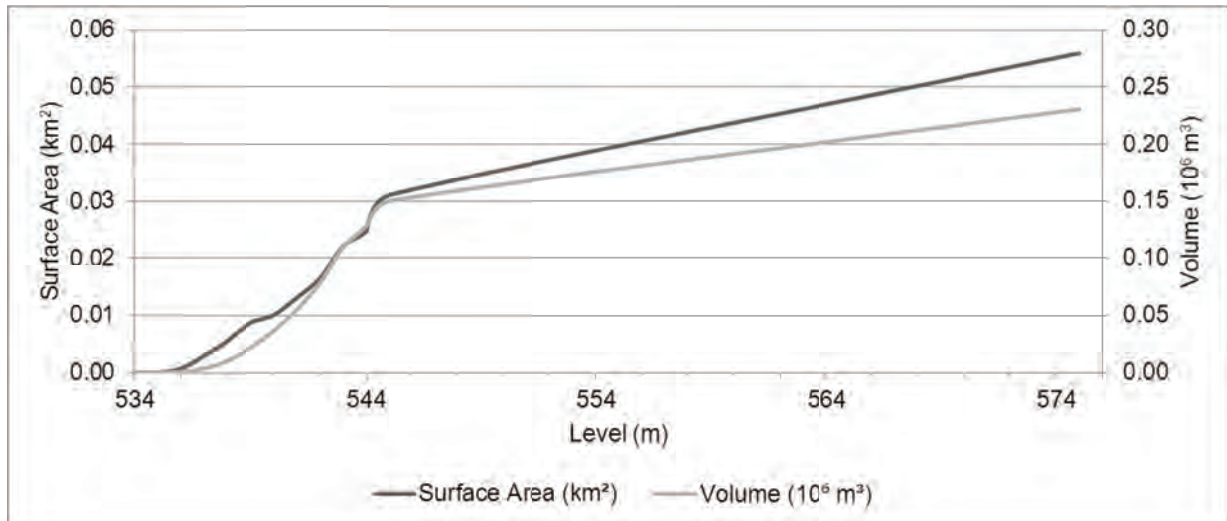


Figure 2.5 FarmReservoir level-area-volume relationships

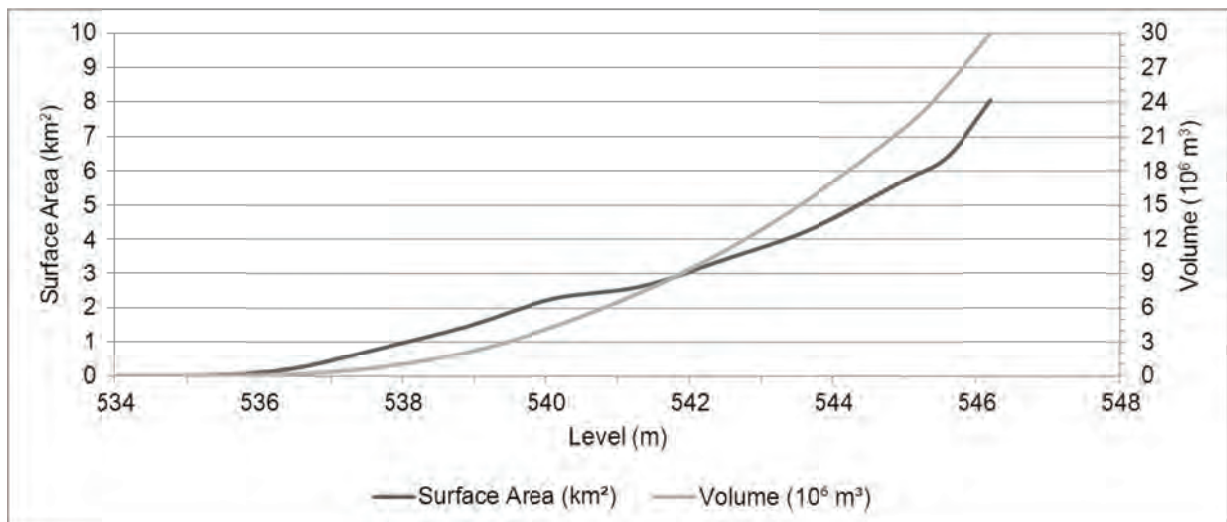


Figure 2.6 Reservoir 1 level-area-volume relationships

The lookup table shown in Table 2.8 was used to simulate a diversion pump that extracted water from the river in Catchment 3 to an off-channel farm reservoir (FarmReservoir) utilised for irrigation. It enabled the magnitude of the diversion to vary based on the flow rate in the main channel.

Table 2.6      Lookup table for diversions to FarmReservoir

Flow rate in river (m <sup>3</sup> /s)	Flow rate of diversion (m <sup>3</sup> /s)
< 2.0	0.0
2.0-3.0	0.5
3.0-9.0	1.0
> 9.0	0.0

## 2.2.1 Evaluation criteria

Each of the river network models were evaluated against a range of criteria to enable the models to be compared. The criteria used in the evaluation are listed in Sections 2.2.1.1 to 2.2.1.7 below.

### 2.2.1.1 User interface

To assess the user interface tools available to assist users in setting up a model the following user interface functionality was evaluated:

- (i) *Is there a Graphical User Interface (GUI)?*
- (ii) *Is the GUI well laid out and logical or intuitive?*
- (iii) *Is there a network visualizer?*
- (iv) *Can model input and output data be interrogated via the network?*
- (v) *Are wizards or expert systems provided?*
- (vi) *Can model input and output data be tabled and graphed?*
- (vii) *Can animations of model results be created?*

### 2.2.1.2 GIS functionality

If the model contains GIS functionality, the following criteria were assessed:

- (i) *Does the model have full GIS integration or is there only GIS visualisation?*
- (ii) *Does the model use input generated via the GIS?*
- (iii) *Can the output be accessed via GIS?*

### 2.2.1.3 Flexible configuration

The flexibility of configuration was assessed by the following questions:

- (i) *What components can be added to the network and what are their basic functions?*
- (ii) *Is there a limit to the number of network components or connections?*
- (iii) *Are multiple demands at extraction points permitted?*
- (iv) *Are water users able to extract water from more than one source?*

- (v) *Can curtailments be applied to water use requests?*
- (vi) *What constraints can be applied: for example, minimum or maximum flows?*
- (vii) *Can hydrological or hydraulic routing along channels be performed?*
- (viii) *Can routing through reservoirs be performed?*
- (ix) *Can in-stream flow requirements (IFRs) be simulated?*
- (x) *Can inter-catchment transfers be modelled?*
- (xi) *Can more complex operating rules be set up where their operation is dependent on the state of other features? For example, a transfer from a node upstream of a reservoir that is dependent on the water level of the reservoir.*

#### **2.2.1.4 Water allocation**

The water allocation method used has a critical impact on the distribution of the water resource to meet demands. In order to determine what water allocation methods can be implemented by the model and their functionality, the following criteria were assessed:

- (i) *What allocation methods are available within the model?*
- (ii) *Are the rules locally or globally based?*
- (iii) *What operating rules are available for reservoirs?*

For each of the network models both the water user priority and FWACS allocation methods were tested. The water user details used for these test cases are specified below.

##### Priority allocation

The user priority allocation system is an allocation method where water demands are allocated based on each user's priority. The available water is allocated to the highest priority users until all demands are met or there is no more water available for allocation. The demands and priorities applied for the priority allocation test case are summarised in Table 2.7, where Priority=1 is the highest priority. In order to cater for changes in the hydrological state of the network, such as periods of drought, curtailments or rule curve restrictions can be used to manage the water resource. The reservoir curtailment levels for Reservoir 1 and FarmReservoir are shown in Table 2.8, where the curtailment level indicates the reservoir storage level below which the user is curtailed to a specified percentage of the demand.

Table 2.7 Water user details for the priority allocation test case

Water users	Water source	Return flow	Demand (m <sup>3</sup> /s)	Priority
Domestic 1	Catchment 1	No	2.50	1
Irrigator 1	Catchment 1	Yes (10%)	2.50	2
Irrigator 2	FarmReservoir	Yes (10%)	0.01	1
Domestic 2	Reservoir 1	No	5.00	1
Transfer_Pump	Reservoir 1	Yes (100%) to tributary of river from Catchment 1	0.50	2
Industry 1	Reservoir 1	No	3.00	3
Irrigator 3	Reservoir 1	Yes (10%)	2.00	4
IFR 1	Catchment 1 and Reservoir 1	N/A	0.50	1
IFR 2	Reservoir 1	N/A	1.00	1

Table 2.8 Reservoir curtailment levels

Reservoir	Water users	Curtailment level (m)	Curtailment (%)
Reservoir 1	Domestic 2	541	80
Reservoir 1	Transfer_Pump	541	80
Reservoir 1	Industry 1	541	80
Reservoir 1	Irrigator 3	541	80
FarmReservoir	Irrigator 2	537	80

### Fractional allocation and capacity sharing

The FWACS allocation system is an allocation method where water users are allocated a fraction of the water flowing in a specified river reach and where water users may be allocated a share in the capacity of a reservoir which they may use to store water (Lecler, 2004). Thus FWACS, as described by Lecler (2004), has two parts, viz.:

- River flow is divided based on fractional allocation. Thus a water user has access to their fraction multiplied by the available flow in the river.
- Storage capacity is divided into storage accounts that water users can either own or rent. These predefined water accounts are operated as water accounts in a “Water Banking” system. The inflows into the storage reservoir are apportioned to the various water accounts; however, these inflow portions may differ from the storage portions. Water losses such as evaporation and seepage are shared across the water accounts. The water users manage their own water accounts separately which can promote more efficient water use and also permits water trade.

Allocation of water via a fractional allocation system, where water demands are met from a user’s predefined fraction of the resource, are detailed in Table 2.9. The two parts of



FWACS as described above are represented by the Domestic 1 and Irrigator 1 water users abstracting from the river in Catchment 1, and several other water users abstracting from Reservoir 1. Water users' storage fractions and inflow fractions for Reservoir 1 are assumed to be the same. Two scenarios were simulated, with the second scenario having a larger fraction apportioned in order to meet the requirements of IFR 2.

Table 2.9 Water user details for the FWACS allocation test case

Water users	Water source	Return flow	Demand (m <sup>3</sup> /s)	Fraction (Scenario 1)	Fraction (Scenario 2)
Domestic 1	Catchment 1	No	2.50	0.60	0.60
Irrigator 1	Catchment 1	Yes (10%)	2.50	0.40	0.40
Irrigator 2	FarmReservoir	Yes (10%)	0.01	1.00	1.00
Domestic 2	Reservoir 1	No	5.00	0.40	0.40
Transfer_Pump	Reservoir 1	Yes (100%) to tributary of river from Catchment 1	0.50	0.15	0.15
Industry 1	Reservoir 1	No	3.00	0.15	0.10
Irrigator 3	Reservoir 1	Yes (10%)	2.00	0.15	0.15
IFR 1	Catchment 1 and Reservoir 1	N/A	0.50	N/A	N/A
IFR 2	Reservoir 1	N/A	1.00	0.15	0.20

### 2.2.1.5 Scenarios

Scenario handling can be powerful tool in the hands of the modeller for exploring alternate solutions to water resource problems.

- (i) *Does the model provide scenario handling functionality?*

### 2.2.1.6 Accounting and auditing

In order to determine what accounting and auditing functionality exists in the model the following criteria were assessed:

- (i) *What queries can be made at points within the network?*
- (ii) *Can source and destination of water be determined?*
- (iii) *Can ownership of water be determined?*

### 2.2.1.7 Operational use

Modelling can be a useful tool in the operational management of water resources, but models developed primarily for water resources planning may not necessarily be suitable for

operational use. For efficient operational use the current state of the river/reservoir network needs to be known, and both short term and long term forecasts need to be included in the simulation. To facilitate this it is important to be able to simulate to a given point in time (e.g. current date) and then retrieve the model state variables in order to continue the simulation when updated observed or forecast data become available. Reservoir level is an example of a state value which is initialised at the start of the simulation and then varies during the simulation as inflows and outflows occur. For operational modelling the state of the reservoir would need to be either maintained when the simulation is restarted from where it was stopped or corrected based on observed data. The following criterion was assessed:

- (i) *Does the model enable storage of state data so that a simulation can be started or restarted using a simulated or actual state for a specified point in time? For example, can simulations be re-run using observed data values to replace forecast data values as they become available?*

The evaluation criteria listed and described in this section are not exhaustive and were chosen for the purpose of selecting a network model suitable for meeting the objectives of the project. However, beyond selecting a suitable model for the project an important consideration is the suitability of the models for use in South Africa as a tool for “real world” water management. These criteria enable the selected network models to be evaluated individually and relative to each other.

### **2.2.2 MIKE BASIN**

For this evaluation the 2011 version of MIKE BASIN with Service Pack 7 was used within ArcGIS 10.0. As MIKE BASIN is an extension for ESRI ArcMap, it requires that the ArcGIS software be installed first. The ESRI ArcGIS software requires a personal or site license. MIKE BASIN requires a USB dongle and a license file provided by DHI for the software to function. An academic evaluation license and USB dongle were obtained from DHI-SA to carry out this evaluation.

MIKE BASIN is one software product in the Mike by DHI suite (DHI, 2011d). The dedicated customer care e-mail address is [mikebydhi@dhigroup.com](mailto:mikebydhi@dhigroup.com), but there are also support centres in numerous countries around the world that can be contacted directly via e-mail or telephone for local support (DHI, 2011d). The e-mail address for DHI-SA who represent DHI in southern Africa is [mikebydhi.za@dhigroup.com](mailto:mikebydhi.za@dhigroup.com) (DHI, 2011d). While setting up this evaluation a number of queries were e-mailed to DHI-SA to ensure correct understanding of the model and this user support proved valuable.

### 2.2.2.1 User Interface

MIKE BASIN is an extension for ESRI ArcMap and therefore is run from within the ArcMap GUI using additional toolbars. The MIKE BASIN GUI with the test project loaded is shown in Figure 2.7. The MIKE BASIN Toolbar and the MIKE BASIN Results Toolbar are shown in Figure 2.7. The MIKE BASIN Toolbar is the primary tool for MIKE BASIN project management, setting up model networks and running the model. The MIKE BASIN Results Toolbar contains tools that facilitate the management and display of output generated by the model. Once flow network components have been added to the network layer, their MIKE BASIN properties can be easily accessed, for example, the properties of a catchment, as displayed in Figure 2.8.

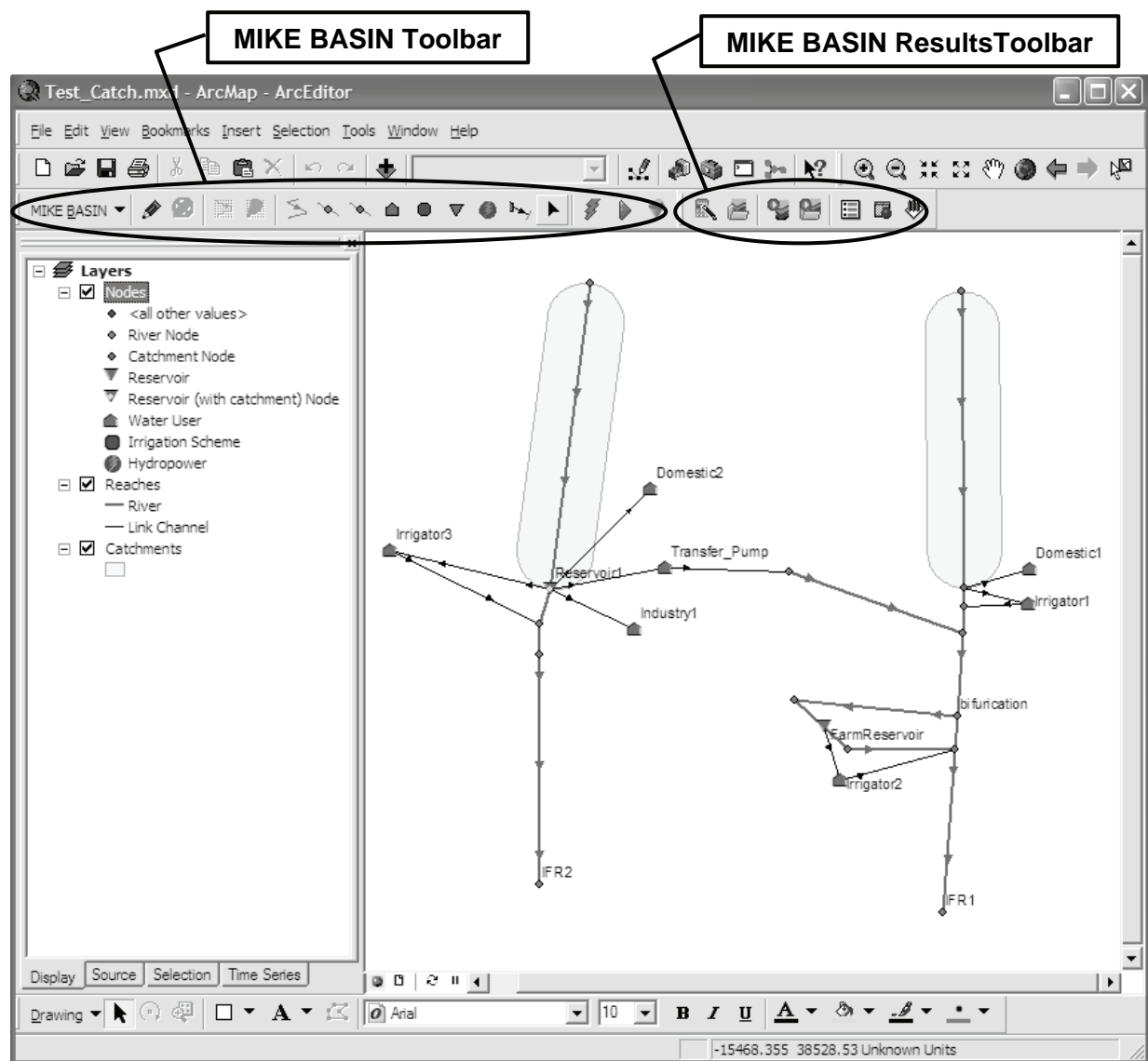


Figure 2.7 MIKE BASIN project within ESRI ArcMap (DHI, 2009)

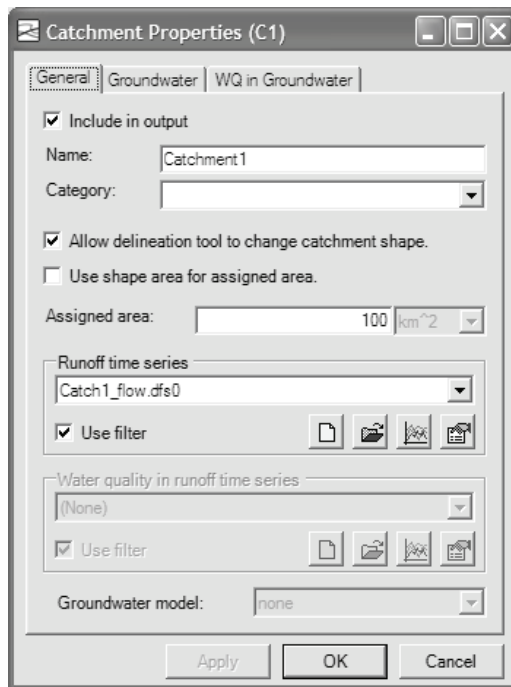


Figure 2.8 MIKE BASIN Catchment Properties window (DHI, 2009)

The output generated by the model can then be associated with ArcGIS map features which aids in the user friendliness of the interface. MIKE BASIN projects are saved as ArcMap Document files with a “mdx” extension. A geodatabase is also created with the same name as the MIKE BASIN project, but with an “mdb” extension, where this and any other files generated are, by default, located in the same folder as the MIKE BASIN project file.

The MIKE BASIN user interface was evaluated as follows:

(i) *Is there a GUI?*

Yes.

(ii) *Is the GUI well laid out and logical or intuitive?*

The interface, as indicated above, is largely based around the fact that MIKE BASIN is an extension for ArcMap and is thus easy to use if the user has previous working knowledge of ArcMap. For non-GIS users it is essential to first acquire GIS skills, which are not too demanding.

(iii) *Is there a network visualizer?*

Yes, networks are built within the map window of ArcMap.

(iv) *Can model input and output data be interrogated via the network?*

Yes, by selecting the *MIKE BASIN Feature Properties* icon from the MIKE BASIN Toolbar and then clicking on the feature of interest, within the network, in the map window.

(v) *Are wizards or expert systems provided?*

Yes, MIKE BASIN contains a result layer wizard that enables results to be processed and added as layers within the geodatabase. There are also tools to generate reports and graphs, and for management, diagnostics and optimization of systems.

(vi) *Can model input and output data be tabled and graphed?*

Yes, this includes output such as frequency analysis curves and time series analysis.

(vii) *Can animations of model results be created?*

Yes, result groups generated from the result layer wizard can be selected for animation.

### **2.2.2.2 GIS Functionality**

MIKE BASIN is an extension for ArcMap and is thus fully integrated with the GIS functionality of ArcMap. The flow network for MIKE BASIN is built up of Node, Reach and Catchment layers within ArcMap. These spatially explicit layers make it easier to configure the model as variables such as catchment area or reach length can be calculated using tools in the GIS. A Digital Elevation Model (DEM) can be added to the MIKE BASIN project and can be used to determine flow directions as well as aid in catchment delineation. In this example, no DEM was used, thus catchment areas were manually input using the Catchment Properties window shown in Figure 2.8. If the catchment had been generated from the DEM, or was associated with a pre-defined catchment polygon, then the area could be easily linked to this spatial feature.

The MIKE BASIN GIS functionality was evaluated as follows:

(i) *Does the model have full GIS integration or is there only GIS visualisation?*

The model has full GIS integration with ArcGIS.

(ii) *Does the model use input generated via the GIS?*

Yes, model inputs can be generated via GIS and fed directly into the model.

(iii) *Can the output be accessed via GIS?*

Yes, computed results can be displayed as GIS layers and generated output time series can be accessed via the GIS interface.











### 2.2.2.3 Flexible Configuration

The MIKE BASIN configuration flexibility was evaluated as follows:

(i) *What components can be added to the network and what are their basic functions?*

The network components available in MIKE BASIN are listed and described in Table 2.10.

Table 2.10 MIKE BASIN components (after DHI, 2009)

Icon	Component	Description / Functionality
	Catchment Node	Represents a River Node at the exit of a catchment.
	Catchment	Represent catchment runoff, ground water and ground water quality interaction with surface water.
	River Node	Acts as accumulation, bifurcation or extraction points, and enables minimum flow requirements to be specified.
	River Reach / Branch	Enables modelling of flow losses, flow capacities, hydraulics and water quality changes based on decay.
	Link Channel	A Channel is similar to a River Reach, but is primarily used to connect Water User, Hydropower or Irrigation Scheme components to their water sources and destinations.
	Water User	Represent water use or demand points that can also interact with ground water and water quality. A Water User can also be configured as a point source for water and water quality constituents.
	Irrigation Scheme	Enables calculation of irrigation demands based on crop growth and climatic conditions.
	Reservoir	A Reservoir can be simulated as a rule curve reservoir, allocation pool reservoir or lake. Various operating rules, spillway and water quality options are provided. An optional remote flow control node can be specified which is useful for setting up a downstream minimum flow requirement for a River Node for example.
	Reservoir with Catchment Node	Same as Reservoir put indicating that the reservoir is at the exit of a catchment with the catchment's flow entering the Reservoir.
	Hydropower	A Hydropower component enables representation of varying water needs based on varying power requirements and can only be linked to a Reservoir.

(ii) *Is there a limit to the number of network components or connections?*

The components and connections are stored as records in a MS Access geodatabase and thus the only limitation is the number of records permitted by MS Access.

(iii) *Are multiple demands at extraction points permitted?*

Yes.

(iv) *Are water users able to extract water from more than one source?*

Yes, users can extract from Reservoirs, River Nodes, Catchment Nodes, including groundwater. Water users are able to prioritise or set preferences for water sources if they are linked to multiple water sources.

(v) *Can curtailments be applied to water use requests?*

Yes.

(vi) *What constraints can be applied: for example, minimum or maximum flows?*

Minimum and maximum flows, and flow losses.

(vii) *Can hydrological or hydraulic routing along channels be performed?*

Yes, Linear Reservoir, Muskingum and Translation routing are available options. However, there are some limitations, as routing is not allowed for reaches immediately downstream of reservoirs or immediately downstream of nodes that have more than one upstream connection. However, the addition of more river nodes enables these limitations to be overcome.

(viii) *Can routing through reservoirs be performed?*

No, but the reservoir has flow release controls based on spillway capacity tables and flood control parameters.

(ix) *Can IFRs be simulated?*

Where a large reservoir controls the hydrology of a system, a remote flow control node may be specified for a Reservoir component to model IFRs. Minimum flow requirements may also be set for River Node components, but would need to be set at each River Node component upstream of the IFR site, as minimum flow requirements are determined at a local level.

(x) *Can inter-catchment transfers be modelled?*

Yes.

(xi) *Can more complex operating rules be set up where their operation is dependent on the state of other features?*

No, this is not possible from within the user interface, but may be possible through the use of macros to implement such rules, though this was not explored in this evaluation.

#### **2.2.2.4 Water Allocation**

The MIKE BASIN water allocation methods were evaluated as follows:

(i) *What allocation methods are available within the model?*

MIKE BASIN provides user priority and FWACS allocation methods. The allocations are determined at a local level, where water is distributed based on user demands at

a particular node. Only Reservoir components provide the facility to configure a remote node where, for example, a specified minimum flow level can be set.

*(ii) Are the rules locally or globally based?*

Generally rules are locally based and are performed in flow order, starting with the topmost river reach and progressing downstream.

*(iii) What operating rules are available for reservoirs?*

Reservoir components can be simulated as a rule curve reservoir, allocation pool reservoir or a lake, and thus priority water allocation or capacity sharing can be modelled.

### Priority allocation

The hypothetical catchment described in the introduction to Section 2.2 was set up in MIKE BASIN as displayed in Figure 2.7. Each water user at a node was assigned a local priority. Setting up the network and the priorities was relatively straight forward.

If a DEM or ESRI shape file had been used to configure the catchment, then the MIKE BASIN properties for the catchment could be easily configured to make use of that area information via a check box option as illustrated in Figure 2.8. However, for this evaluation the catchment areas were simply input into the appropriate field. The time series of runoff in l/s/km<sup>2</sup> was created by making a new time series file and copying in the catchment runoff data from Table 2.4.

Setting up the water users required adding Water User nodes, linking them to their sources and then specifying a demand time series. In this test only one water source was used but multiple sources could have been configured.

The assignment of the priorities for the two water users abstracting from the Catchment 1 node was specified using the node's properties window. The reservoir initial, bottom, dead storage and crest levels as well as their level-area-volume relationships were set for both reservoirs. Priorities were also specified for the water users using water from Reservoir 1 as well as IFR 2's minimum demand downstream of the reservoir via the remote flow control option. The curtailments to be implemented at specified reservoir levels were also set. In order to configure the transfer (Transfer\_Pump) which simulates an inter-catchment transfer that abstracts water from Reservoir 1 and feeds it into Catchment 3, a water user with a 100 % return flow into the node feeding Catchment 3 was used. A demand time series and return flows were set for Transfer\_Pump.



The results from running the MIKE BASIN model based on the user priority allocation rules were as expected. Irrigator 1 was curtailed at the beginning of June as the flow could no longer sustain both users' requirements. As Domestic 1 has a higher priority, its demands were met first. In July there is a further reduction in the streamflow at the abstraction point and Irrigator 1, who had the lower priority, received none of its demand and Domestic 1's abstractions were then curtailed to 2 m<sup>3</sup>/s, which was the remaining flow in the river channel. As specified by the curtailments, when the level in the Reservoir 1 reached 541 m the supply to all reservoir water users was curtailed to 80% of their demands. For the abstraction from the river in Catchment 3 to FarmReservoir, the results demonstrated that the lookup table used to divert water to FarmReservoir was successfully implemented in MIKE BASIN. IFR 1 (N28) and IRF 2 (N34) are control nodes at the exit of Catchment 3 and Catchment 4 respectively. IFR 2 was configured such that Reservoir 1 was used as a control to meet the required minimum flow of 1 m<sup>3</sup>/s. The results showed that this minimum flow was met.

#### Fractional allocation and capacity sharing

The components and network layout for FWACS was the same as that for the priority allocation shown in Figure 2.7, and only the properties were changed. The flow, at the river node in Catchment 1 from which users Domestic 1 and Irrigator 1 abstract water, is first divided into the fractional allocation portions of 60 % and 40 %. It is then determined how much of the demand by each user can be met from their allotted portions. In the results it was shown that when the flow drops to 6.0 m<sup>3</sup>/s it is sufficient to meet both demands of 2.5 m<sup>3</sup>/s, but this flow is first divided into portions based on each water user's fractional share. The portions are 3.6 m<sup>3</sup>/s (6.0 m<sup>3</sup>/s x 0.6) for Domestic 1, and 2.4 m<sup>3</sup>/s (6.0 m<sup>3</sup>/s x 0.4) for Irrigator 1. Thus the water demanded by Irrigator 1 could not be met even though Domestic 1 had surplus at that point. The reservoir capacity sharing functioned as expected with each user's capacity share being represented as a user pool. Using the FWACS Scenario 1 the minimum flow requirement of 1 m<sup>3</sup>/s for IFR 2 (N34) could not be met 100% of the time. In FWACS Scenario 2 the capacity allocation of the reservoir was changed by increasing the portion available to the downstream flow from 15% to 20% and reducing the capacity assigned to Industry 1 from 15% to 10%. These changes to the capacity allocation resulted in the minimum flow requirement at IFR 2 being met.

### **2.2.2.5 Scenarios**

The MIKE BASIN scenario handling functionality was evaluated as follows:

*(i) Does the model provide scenario handling functionality?*

Names can be assigned to model runs; this enables model output to be associated with a particular model run which aids in testing a number of modelling scenarios. However, no means is provided to store model inputs for different modelling scenarios other than saving each scenario in a different MIKE BASIN project.

### **2.2.2.6 Accounting and Auditing**

The MIKE BASIN accounting and auditing functionality was evaluated as follows:

*(i) What queries can be made at points within the network?*

Time series can be queried at each network component within MIKE BASIN. Both inflows and outflows can be queried.

*(ii) Can source and destination of water be determined?*

The sources and destinations are limited to directly linked nodes.

*(iii) Can ownership of water be determined?*

Water ownership is not accounted for.

### **2.2.2.7 Operational Use**

The suitability of MIKE BASIN for operational use was evaluated as follows:

*(i) Does the model enable storage of state data so that a simulation can be started or restarted using a simulated or actual state for a specified point in time?*

No, not directly. The model was run for the first half of the year and then the reservoir levels were manually copied across from the output files to the initial values in the property dialogs for the reservoirs and then run for the second half of the year. This resulted in the same reservoir levels obtained from the full year run.

### **2.2.2.8 Discussion**

A three day MIKE BASIN course was attended before this evaluation took place which aided in a relatively quick learning curve regarding the general methodology used to configure and run the MIKE BASIN model. The training manual from this MIKE BASIN course (DHI, 2011a) was also an invaluable resource while setting up and running this evaluation. DHI-SA has initiated a web based discussion forum (<http://www.dhi-students.co.za>) for students to facilitate interaction and learning (Pott, 2011). Online presentations are also available

from this discussion forum (Pott, 2011). The factors discussed above as well as the local user support meant that this was the quickest of the three models to configure for this evaluation.

An assessment of how well MIKE BASIN met the evaluation criteria is shown in Table 2.11, with the number of times the model met the criteria being totalled. MIKE BASIN scored high in the “User Interface”, “GIS Functionality” and “Water Allocation” evaluations by meeting all the requirements. The “Flexible Configuration” evaluation also scored highly. MIKE BASIN achieved lower scores in the “Scenarios”, “Accounting and Auditing” and “Operational Use” evaluations.

Table 2.11 MIKE BASIN evaluation scores

<b>Evaluation Criteria</b>	<b>Number of Criteria</b>	<b>Score</b>
User Interface	7	7
GIS Functionality	3	3
Flexible Configuration	11	10½
Water Allocation	3	3
Scenarios	1	½
Accounting and Auditing	3	1
Operational Use	2	1

### 2.2.3 MODSIM

MODSIM version 8.1 Beta was used for this test. MODSIM is freely available for download from the Colorado State University’s web site [<http://modsim.engr.colostate.edu/>] and does not require a dongle or licence. A version of MODSIM named Geo-MODSIM, which includes a GIS interface, is mentioned in the literature (Triana and Labadie, 2007), however this version is still under development and its launch has been delayed indefinitely (Labadie, 2011). It was not possible to attend a training course for MODSIM prior to this evaluation, so the use of the model was mainly learned through the user manual tutorials provided by the developers.

MODSIM is not a commercially sold model and dedicated user support was not available, though limited support was obtained by e-mail from the developer, Dr Labadie (Labadie@engr.colorstate.edu).

### 2.2.3.1 User Interface

MODSIM is a standalone application that operates on the Microsoft Windows operating platform. The GUI with the network representing the hypothetical test catchment loaded is displayed in Figure 2.9. MODSIM makes use of a Multiple Document Interface (MDI), thus unlike the other models tested, multiple model configurations can be loaded and viewed at one time, which is especially useful when setting up and running different scenarios. The network components are dragged from the Node Palette form, displayed in Figure 2.9, onto the network editor window to build up the network. The Network Settings window is used to set up general model parameters as well as run parameters.

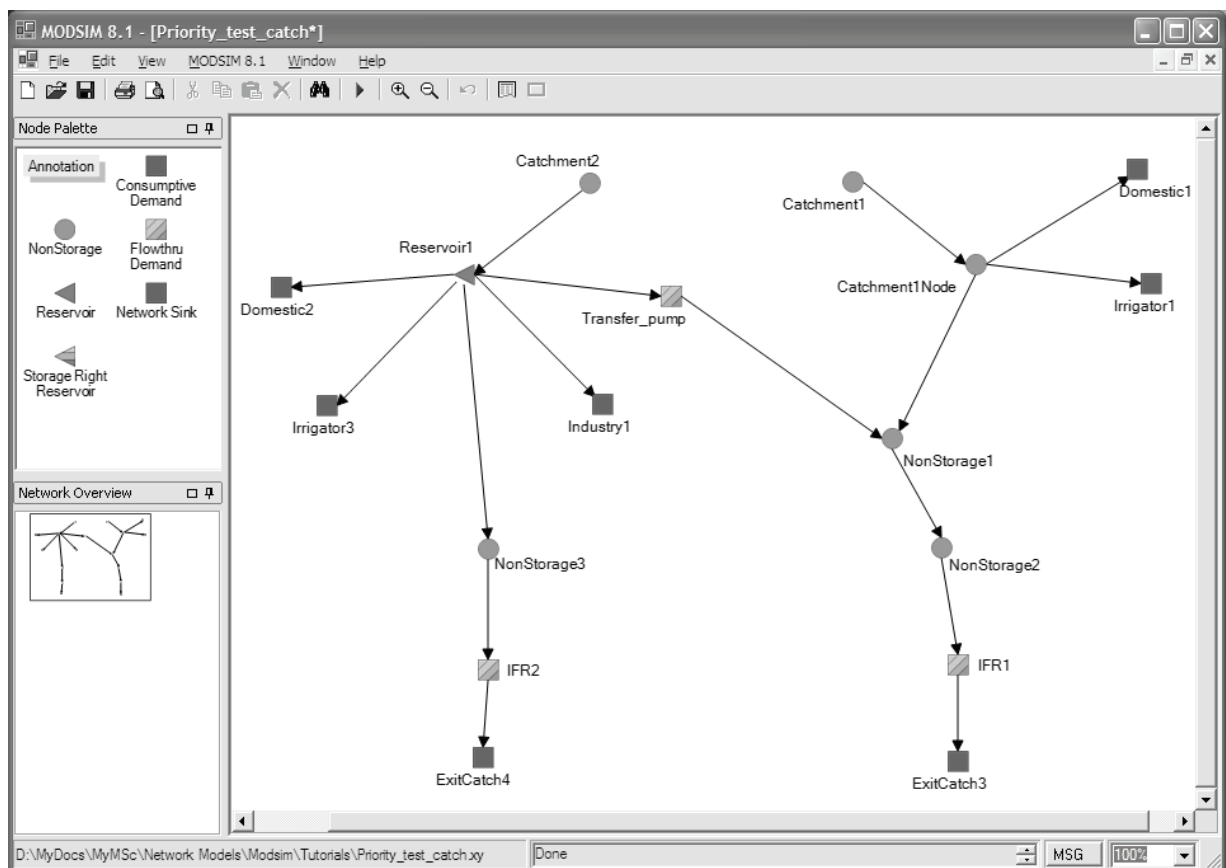


Figure 2.9 MODSIM's user interface (Labadie, 2010a)

The MODSIM user interface was evaluated as follows:

(i) *Is there a GUI?*

Yes, the GUI has been developed in Microsoft Visual C++ .NET and accesses the model libraries that can also be accessed via the Custom Code Editor or a .NET application (Labadie, 2010a).

(ii) *Is the GUI well laid out and logical or intuitive?*

As with RiverWare, MODSIM's GUI follows the general Microsoft Windows application layout with menus, a toolbar and a workspace where the networks are created but differs in that it makes use of a MDI. The Node Palette and Network Overview windows can be moved, docked and pinned, thus adding a degree of customisability to the interface.

(iii) *Is there a network visualizer?*

Yes, there is a network visualizer and a background image can be loaded into the network editor to facilitate visualising and laying out the network.

(iv) *Can model input and output data be interrogated via the network?*

Yes, by double clicking on a node or link the properties and input can be viewed and set. Right clicking on a link or node opens a localised menu where the graph option can be selected to open the graphing tool.

(v) *Are wizards or expert systems provided?*

MODSIM supports Monte Carlo Analysis as well as “a state-of-the-art” network flow optimization algorithm (Labadie, 2010a). MODSIM can also be customised through the use of Microsoft .NET.

(vi) *Can model input and output data be tabled and graphed?*

Yes, input and output data can be viewed in either a data table view or plotted as a graph.

(vii) *Can animations of model results be created?*

Yes, MODSIM has a tool to animate the output results for reservoirs, demand nodes and network links. The animation changes the colours of the links and nodes within defined colour ranges.

### **2.2.3.2 GIS Functionality**

The MODSIM GIS functionality was evaluated as follows:

(i) *Does the model have full GIS integration or is there only GIS visualisation?*

MODSIM version 8.1 Beta does not have any GIS functionality. Geo-MODSIM is a GIS based version MODSIM (Triana and Labadie, 2007), however it is not publicly available as indicated above.

(ii) *Does the model use input generated via the GIS?*

No, but GIS coverages can be exported as images and then imported as a background to assist in visualising networks.

(iii) *Can the output be accessed via GIS?*

No, the output can be accessed from the network visualizer but results cannot be displayed as a map.










### 2.2.3.3 Flexible Configuration

The MODSIM configuration flexibility was evaluated as follows:

(i) *What components can be added to the network and what are their basic functions?*

The components provided in MODSIM and their functions are summarised in Table 2.12.

Table 2.12 MODSIM components (after Labadie, 2010a)

Icon	Component	Description / Functionality
	Consumptive Demand	Nodes that represent water users or a demand. The demand can also represent an abstraction from ground water.
	NonStorage	Nodes that can represent catchment runoff, river nodes, confluences, diversions and abstraction points or input sources or water rights of a user.
	Flowthru Demand	A modified Consumptive Demand node that requires a destination node. Also referred to as a non-consumptive node i.e. used for controlling instream flow requirements. Can be used along with historical data for model calibration.
	Operations or Hydropower Reservoir	Operations of off-stream or main-stem reservoirs. Flood control, conservations pools and dead storage. Zones for storage balancing in multi-reservoir systems. Reservoir hydropower or run-of-river hydropower, pumped storage.
	Storage Right Reservoir	Storage ownership, pool rentals, water banking and service contracts and water rights storage accounts.
	Network Sink	Catchment outlet.
	Link	Can function as rivers, canals, pipelines and water rights. Channel losses and maximum and minimum flows.
	MultiLink	Multiple water sources and rights including nonlinear cost-discharge functions and non-linear discharge-channel loss functions.
	Routing Link	Streamflow and channel routing.
	Annotation	Can be used to add labels to the network.

(ii) *Is there a limit to the number of network components or connections?*

The number of components and connections appears to be limitless.

(iii) *Are multiple demands at extraction points permitted?*

Multiple water users may be connected to Reservoir and NonStorage nodes.

(iv) *Are water users able to extract water from more than one source?*

Yes, water users can extract from more than one source. Priorities and link costs then determine which source or combination of sources are used to supply the water user.

(v) *Can curtailments be applied to water use requests?*

Yes, Hydrologic State tables can be utilised to specify reservoir operating rules.

(vi) *What constraints can be applied: for example, minimum or maximum flows?*

Both minimum and maximum flow constraints can be applied.

(vii) *Can hydrological or hydraulic routing along channels be performed?*

Yes, Muskingum and user defined lag coefficients (Labadie, 2010a).

(viii) *Can routing through reservoirs be performed?*

Yes, restrictions to reservoir discharge can be made based on the hydraulic capacity and reservoir head. This is limited to one exit node.

(ix) *Can IFRs be simulated?*

Yes, Flowthru Demand components can be used to represent IFRs.

(x) *Can inter-catchment transfers be modelled?*

Yes.

(xi) *Can more complex operating rules be set up where their operation is dependent on the state of other features? For example a transfer from a node upstream of a reservoir that is dependent on the water level of the reservoir.*

Watch Links is a feature that facilitates setting up relationships whereby flow requirements at a point can be dependent on flow conditions at other locations within the network. Theoretically almost any conceivable operation can be configured through custom code.

#### **2.2.3.4 Water Allocation**

The MODSIM water allocation methods were evaluated as follows:

(i) *What allocation methods are available within the model?*

Priority, rent pool, user rights and ownership.

(ii) *Are the rules locally or globally based?*

The priorities are at a network level, i.e. globally based.

(iii) *What operating rules are available for reservoirs?*

Priority allocation with hydrologic state based reduction curves. Capacity sharing is also available.

Unlike MIKE BASIN and RiverWare, MODSIM's priority based water allocation system closely resembles the penalty structure of the WRYM developed and used in South Africa. The priorities are at global level and the network links can have an associated "link cost" that ensures allocation is made in accordance with specified water rights and priority rankings (Labadie, 2010a).

### Priority allocation

The hypothetical catchment described in the introduction to Section 2.2 was set up in MODSIM as displayed in Figure 2.9. Some aspects of the network for the hypothetical catchment could not be modelled, though it is possible that experienced users of the model may be able to work around these apparent limitations. The lookup table method used to divert water to the FarmReservoir in the other models could not be represented. The reduction fractions applied to the water users for the reservoir could also not be implemented. A Flowthru Demand node was used to model the inter-catchment water transfer and IFRs and Consumptive Demand nodes were used to represent the other water users.

The results from running the MODSIM model based on the user priority allocation rules showed that the priority allocation method worked in the same way for the Domestic 1 and Irrigator 1 water users as for the MIKE BASIN and RiverWare models.

### Fractional allocation and capacity sharing

From the literature it appears that it is possible to set up FWACS allocation in MODSIM (Labadie, 2010b). A Flowthru component in conjunction with NonStorage components was configured for the fractional allocation of the runoff from Catchment 1 to the Domestic 1 and Irrigator 1 components as shown in Figure 2.10. The capacity sharing configuration for Reservoir 1 was set up with the Storage Right Reservoir and NonStorage components. Thus the components in the network layout for FWACS had to be configured differently to that used for the priority allocation test.



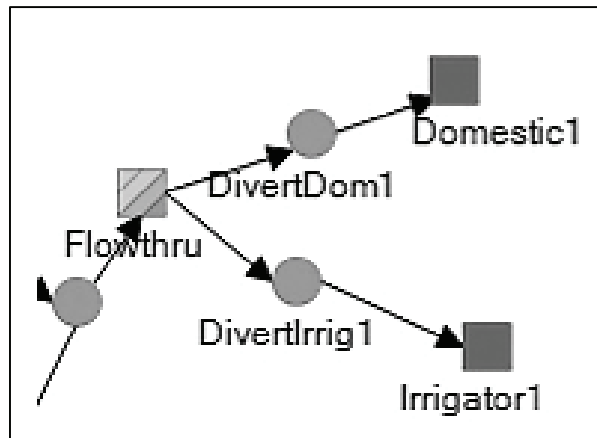


Figure 2.10 Catchment 1 configuration with Flowthru component (Labadie, 2010a)

### 2.2.3.5 Scenarios

The MODSIM scenario handling functionality was evaluated as follows:

- (i) *Does the model provide scenario handling functionality?*

Scenarios can be saved as separate MODSIM (.xy) files. If all the scenario files are open at the same time, in the MODSIM interface, the Scenario Analysis option enables the graphing tool to plot results from all the scenarios concurrently.

### 2.2.3.6 Accounting and Auditing

The MODSIM accounting and auditing functionality was evaluated as follows:

- (i) *What queries can be made at points within the network?*

Component properties, input time series and output can be queried at network links and components.

- (ii) *Can source and destination of water be determined?*

The sources and destinations are limited to directly linked nodes.

- (iii) *Can ownership of water be determined?*

MODSIM has a Water Rights Extension that facilitates keeping track of water ownership through the use of storage ownership accounts (Labadie, 2010a).

### 2.2.3.7 Operational Use

MODSIM can be used for real-time catchment management by linking it with data management systems what include forecast data (Labadie *et al.*, 2007; Labadie, 2010a).

The suitability of MODSIM for operational use was evaluated as follows:

- (i) *Does the model enable storage of state data so that a simulation can be started or restarted using a simulated or actual state for a specified point in time?*

No, not directly. As with the other models evaluated, MODSIM was first run for the full year of 1981, then for the first half and second of the same year. As with MIKE BASIN the reservoir levels were manually copied across from the end of the first half of the year to the initial value for the second half of the year. However, MODSIM required the reservoir storage volume as opposed to the reservoir level. The reservoir storages for the second half of the year thus continued from the same value where the first half of the year left off but did not follow the same curve that was obtained from the full year simulation. This could be as a result of other model state values that were not accounted for.

### 2.2.3.8 Discussion

Configuring the catchment for the FWACS allocation method proved to be challenging without appropriate user support and experience. Unfortunately, this evaluation was done without attending a course and as there is only one contact person for queries it proved difficult to resolve the details required to fully implement the water banking and ownership functionality described in the literature by Labadie (Labadie, 2010b).

An assessment of how well MODSIM met the evaluation criteria is shown in Table 2.13, with the number of times the model met the criteria being totalled. MODSIM scored high in the “User Interface”, “Flexible Configuration”, “Water Allocation” and “Scenarios” evaluations by meeting all the requirements. The MDI interface was a very useful and simple method of comparing scenarios as output from a point of interest could be easily compared across scenarios. The flexibility of changing rules and components between scenarios was also a strong point. MODSIM achieved lower scores in the “GIS Functionality”, “Accounting and Auditing” and “Operational Use” evaluations.

Table 2.13 MODSIM evaluation scores

<b>Evaluation Criteria</b>	<b>Number of Criteria</b>	<b>Score</b>
User Interface	7	7
GIS Functionality	3	1
Flexible Configuration	11	11
Water Allocation	3	3
Scenarios	1	1
Accounting and Auditing	3	2
Operational Use	2	1

## 2.2.4 RiverWare

For this test RiverWare version 6.0.3 – Mar 8 2011 10:53:10 was used. RiverWare requires a license file that is generated based on the Media Access Control (MAC) address of the computer on which it is installed. A six month academic evaluation license was obtained from the developers at the Centre for Advanced Decision Support for Water and Environmental Systems (CADSWES).

Modelling support is provided for RiverWare by CADSWES via e-mail ([riverware-support@colorado.edu](mailto:riverware-support@colorado.edu)). The responses to queries during the evaluation were prompt (generally next day), clear and included examples. The user support obtained during the evaluation was invaluable in providing a clear understanding of the modelling methodology used by RiverWare. The time zone differences between South Africa and the USA proved beneficial, as any queries were compiled at the end of the working day and e-mailed through to the support address, and generally the response e-mail was received by the next morning.

RiverWare refers to the network “objects”, but for the purpose of consistency this document will use the term network “components”.

### 2.2.4.1 User Interface

RiverWare is a standalone application. The RiverWare GUI with the test catchment loaded is displayed in Figure 2.11. The Run Control form enables model run settings to be easily changed and includes functionality to pause a run and step through a run at any point. It should be noted that when a model run starts all the output is cleared. Thus, if testing for a shorter period within the full simulation period, it is better to run from the start date and pause and then step through the period of interest. If this is not done, state or other required input would have to be specified just before the period of interest. The network components contain data containers called slots. Setting up the run parameters facilitates the subsequent addition of time series slots to the network components, as the time step specified on the Run Control form is used by the software for entering data into the time series slots. The network components are dragged from the Sim Object Pallet form onto the workspace in order to build up the network. The properties of network components added to the workspace can be set using their respective properties windows, as shown in Figure 2.12.

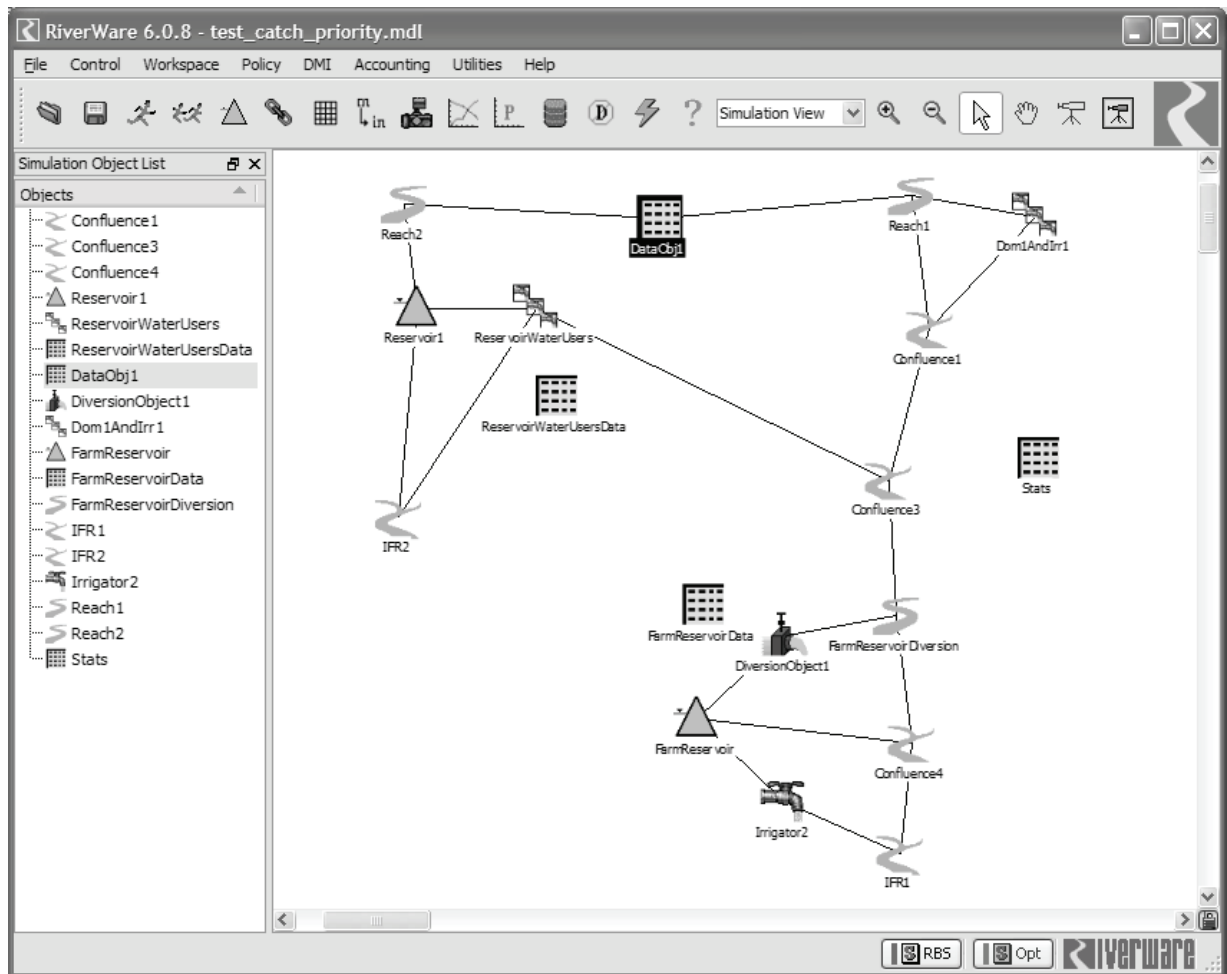


Figure 2.11 RiverWare's user interface (CADSWES, 2011)

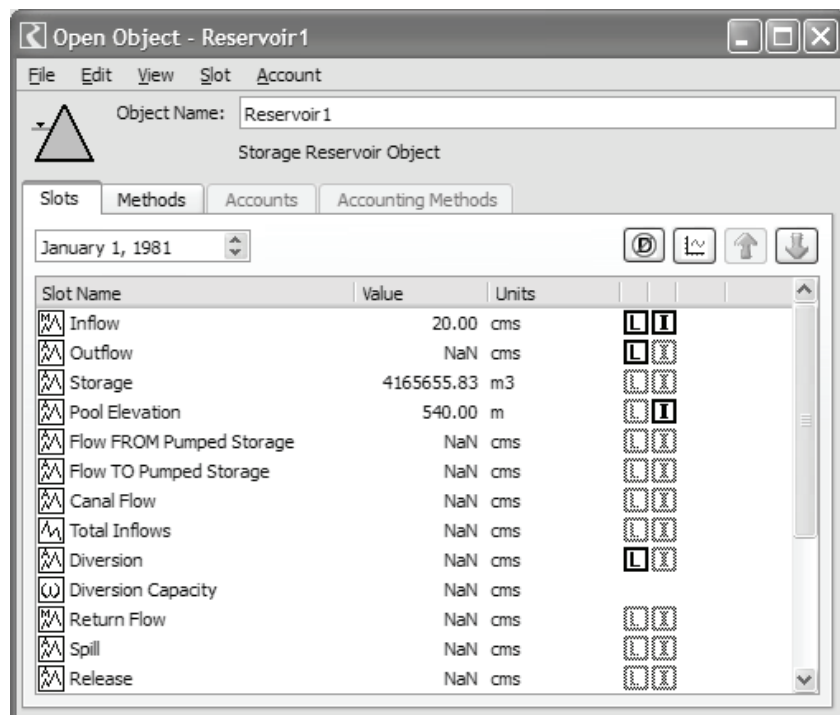


Figure 2.12 RiverWare's reservoir properties window (CADSWES, 2011)

The RiverWare user interface was evaluated as follows:

(i) *Is there a GUI?*

Yes.

(ii) *Is the GUI well laid out and logical or intuitive?*

The GUI follows the general Microsoft Windows application layout with menus, a toolbar and a workspace where the networks are created. The component selector window can be moved and docked, thus adding a degree of customisability to the interface.

(iii) *Is there a network visualizer?*

Yes, networks can be displayed in the workspace as either a simulation view (default) or a geospatial view. The simulation view enables components to be placed in a manner that facilitates visualizing the network and the geospatial view enables components to be placed relative to one another based on a x and y location. The network visualizer has a useful lock feature that prevents components from being accidentally moved around.

(iv) *Can model input and output data be interrogated via the network?*

Yes, input and output slots are contained in the network components and are thus easily accessible via the network. Expression slots are a useful means of calculating statistics and other user defined calculations. These are located in Data components and are thus accessible from the network but need to be clearly named to avoid confusion.

(v) *Are wizards or expert systems provided?*

RiverWare makes use of a RiverWare Policy Language (RPL) for both expression slots and rule-based computations. The RPL Palette form facilitates setting up complex functions, rules and expressions. Due to the complex nature of changing prerequisite inputs and outputs, and the possible effect on rules, a wizard is provided to analyse a model run at a component level, which is invaluable for determining why a run may have not been completed. Missing inputs required to run the methods for a particular component can be tracked with the aid of this tool.

(vi) *Can model input and output data be tabled and graphed?*

Yes, input and output data are stored in slots for which data can be represented in a tabular format and as graphs, including statistical analyses such as frequency analysis curves. RiverWare slots have the ability to specify input or output status at a record or time step level. Thus one data slot could contain both input and output values within the same time series. This is a unique feature of RiverWare compared to the other the models tested.

(vii) *Can animations of model results be created?*

No, animation of results is not a feature of RiverWare.

#### 2.2.4.2 GIS Functionality

There is no GIS functionality within RiverWare apart from the ability to visualise networks geospatially. Thus, the evaluation of GIS functionality is as follows:

(i) *Does the model have full GIS integration or is there only GIS visualisation?*

RiverWare only provides simple GIS visualisation.

(ii) *Does the model use input generated via the GIS?*

No.

(iii) *Can the output be accessed via GIS?*

No.

#### 2.2.4.3 Flexible Configuration

The RiverWare configuration flexibility was evaluated as follows:

(i) *What components can be added to the network and what are their basic functions?*

The RiverWare software includes a comprehensive list of network components which are listed and described in Table 2.14.

Table 2.14 RiverWare components (after CADSWES, 2011)























Icon	Component	Description / Functionality
	Storage Reservoir	A reservoir with Release and spillways and no hydropower facilities. Storage is a function of Pool Elevation as defined by an Elevation-Volume Table.
	Level Power Reservoir	A reservoir with a hydropower plant (Turbine Release) and spillways. Storage is a function of Pool Elevation as defined by an Elevation-Volume Table.
	Slope Power Reservoir	A reservoir with hydropower facilities and spillways. Storage is a combination of level storage and wedge storage. Wedge storage is defined by a table which relates headwater elevation and Inflow to a water surface profile.
	Pumped Storage	A reservoir which has reversible pump-turbines. The turbines may generate or pump at each time step. Storage is a function of Pool Elevation as defined by an Elevation-Volume Table.
	Reach	A river section which routes water using one of many possible routing algorithms. Reaches may lose water to a Diversion and gain water from Return Flow. Reaches can also have side inflows, gains, and/or losses.
	AggReach	An aggregate component which contains one or more Reach components.

Table 2.14 (continued) RiverWare components (after CADSWES, 2011)

Icon	Component	Description / Functionality
	Confluence	A flow junction with two Inflows and a single Outflow.
	Bifurcation	A flow junction with a single Inflow and two Outflows.
	Control Point	A component used to regulate upstream reservoirs so that channel Capacity at the control point is not violated.
	Inline Power	A component used to model power production on a stretch of reach with no storage (run of river power production).
	Canal	A bi-directional conveyance channel which delivers water by gravity between two Reservoirs.
	Pipeline	A component that models flow in a pipeline between two components.
	Pipe Junction	A component used with pressurized flow to split flows similar to a bifurcation or bring flows together similar to a confluence.
	Inline Pump	A component used to model a booster pump station. It controls solution direction, calculates added head and calculates the power consumed.
	Agg Distribution Canal	An aggregate component which serves to route Diversion Requests from a Water User upstream to a Diversion component. It also routes flow from the Diversion component down to the Water Users.
	AggDiversion Site	An aggregate component which contains zero or more Water Users. It diverts water from a Reach or a Reservoir. The Water User elements consume water and return excess flow to the system.
	Diversion Component	A component which diverts water from a Reservoir or Reach. The amount of water which may be diverted is based on water surface elevation, pumping parameters, or available water.
	Water User	A component that diverts water from a Reach or a Reservoir, consumes water, and then returns excess flow to the system.
	Ground Water Storage	An underground storage reservoir which receives Inflow from Water User Return Flow or Reach seepage and can return water to the system.
	Stream Gauge	A component used to represent stream gauge location. It shows the discharge data at a particular location in a model.
	Thermal Component	A component which models the economics of the thermal power system and the thermal replacement value of the hydropower.
	Data Component	A container for user-defined data to be imported to and/or exported from RiverWare. Data Components may also contain expression slots for performing user-defined calculations.

(ii) *Is there a limit to the number of network components or connections?*

There does not appear to be a limit on the number of components or connections.

(iii) *Are multiple demands at extraction points permitted?*

Yes, if more than one Water User diverts water from a Reservoir or Reach then an AggDiversion Site component must be used.

*(iv) Are water users able to extract water from more than one source?*

No, a Water User can extract water from individual Reservoir, Reach and Diversion components, but it does not appear to be possible to extract from multiple sources. This may be possible using more complex methods such as rules and expression slots, which were not explored in this evaluation.

*(v) Can curtailments be applied to water use requests?*

Yes.

*(vi) What constraints can be applied: for example, minimum or maximum flows?*

Yes, constraints can be applied, but these would have to be implemented by creating rules.

*(vii) Can hydrological or hydraulic routing along channels be performed?*

Yes, routing methods can be applied in Reach components and include Time Lag, Variable Time Lag, Impulse Response, Step Response, Variable Step Response, Muskingum, Kinematic, Muskingum Cunge, Muskingum Cunge Improved, MacCormack, Storage and Variable Storage routing.

*(viii) Can routing through reservoirs be performed?*

Yes, hydrologic inflow methods are available for reservoirs.

*(ix) Can IFRs be simulated?*

Yes, rules or Control Point components can be configured to model minimum flow requirements.

*(x) Can inter-catchment transfers be modelled?*

Yes.

*(xi) Can more complex operating rules be set up where their operation is dependent on the state of other features? For example a transfer from a node upstream of a reservoir that is dependent on the water level of the reservoir.*

Almost any conceivable operation can be configured using the RPL.

#### **2.2.4.4 Water Allocation**

The RiverWare water allocation methods were evaluated as follows:

*(i) What allocation methods are available within the model?*

The system is primarily geared towards a priority system, however in this evaluation examples of priority and fractional allocation were set up successfully. The water accounting ability could have also been used for the fractional allocation testing.

*(ii) Are the rules locally or globally based?*

Rules are run in order of their priority and not by component and can thus be applied at a global level.



*(iii) What operating rules are available for reservoirs?*

Reservoirs can have numerous methods applied to them, enabling user priority allocation and capacity sharing to be modelled.

Priority allocation

The hypothetical catchment described in the introduction to Section 2.2 was set up in RiverWare as displayed in Figure 2.11. As Aggregate Diversion Sites were used with each of their Water Users linked via a sequential structure, the Water User priority was based on the order in which they were listed in the Aggregate Diversion Site.

Setting up the network in RiverWare required dragging the appropriate components into the workspace and linking them appropriately. Initially some trial and error was required to select the most appropriate network components to represent the hypothetical catchment. As no catchment components are provided, expression slots were used to calculate the runoff from the catchments. Catchment runoff from Table 2.4 was converted to cubic meters per second ( $\text{m}^3/\text{s}$ ) and stored in a slot called CatchFlow in a Data Component called DataObj1. Two expression slots representing the two catchments were created *viz.* Catch1Flow and Catch2Flow. These were then set up to calculate runoff as the CatchFlow slot multiplied by the appropriate catchment area. The execution order of the expression slots was found to be important when running the model, and they were therefore set to execute at the beginning of each time step. These expression slots were then linked to the inflows for each catchment *viz.* Reach 1 and Reach 2. All the water users were set up in a priority allocation system based on their order in the AggDiversion Site components. The Dom1AndIrr1 AggDiversion Site component was configured with allocation to the Domestic 1 user first then to the Irrigator 1 user. Setting up the water users required first adding the AggDiversion Site components, then appending each user at a site in order of priority. The abstraction rates for these users were set in their corresponding Diversion Request slots.

The reservoir level-area-volume relationships and initial pool elevations were set in the appropriate slots for the Reservoir components. A Data Component, named ReservoirWaterUsersData was created and the water user demands and curtailments (reduction fractions) were added. Rules were configured with the aid of RiverWare's RPL Palette to specify the configuration of rules and how the slots interact with each other. The rules were grouped according to the reservoirs on which they operated and ordered so as to represent the hypothetical catchment described in the introduction to Section 2.2. Dead storage and crest levels were specified within the rules created to determine the water

available for each of the specific water users based on the calculated reservoir level. A release for minimum flow rule was used to represent the IFR 2 site downstream of Reservoir 1. For the inter-catchment transfer the Transfer\_Pump water user's Fractional Return Flow slot was set to 1 and its Return Flow slot was linked to the inflow in Catchment 3.

The results from running the RiverWare model based on user priority allocation rules were as expected. Irrigator 1 was curtailed at the beginning of June as the flow could not sustain both users' requirements. As the Domestic 1 user has a higher priority, its demands are met first. In July there was a further reduction in the in-stream flow at the abstraction point and the lower priority Irrigator 1 then receives none of its demand and the Domestic 1 user was then curtailed down to 2 m<sup>3</sup>/s which was the remaining flow in the river channel. The curtailments for the Reservoir 1 water users were implemented as expected. The outflow from the reservoir had some oscillation as a result of the simplified spillway rating table that had been applied. This spillway rating table was a requirement in RiverWare unlike MIKE BASIN where it was an optional setting. Abstractions from the river in Catchment 3 flow into FarmReservoir were as expected. The flows at IFR 1 and IFR 2, were as expected, except for the oscillations as mentioned above.

#### Fractional allocation and capacity sharing

The components and network layout for the FWACS allocation configuration were largely the same as the configuration for the priority allocation. There were, however, some extra slots and data components required, and rules were changed to meet the FWACS parameters and some changes were made to some slot linkages. The results obtained were as expected, and, as shown in the MIKE BASIN evaluation, there is a point in the simulation where the Domestic 1 user receives its full allocation while the Irrigation1 user is in deficit, even though there is sufficient water in the river. The minimum flow at IFR 2 was determined at each time step before the reservoir was fractionally allocated to the water users as the IFR had not been set up with a capacity share. This configuration was different to the MIKE BASIN test where the flows to IFR 2 downstream were allocated from an environmental fraction of the reservoir. Due to the way that the rules were configured, the minimum flow requirement had the highest priority and thus was always met.

#### **2.2.4.5 Scenarios**

The RiverWare scenario handling functionality was evaluated as follows:

- (i) *Does the model provide scenario handling functionality?*

RiverWare has a scenario manager utility in which a baseline is saved then various scenarios can be configured and compared. These scenarios are limited to changes in slot values as once the baseline has been saved the network components, links and rules cannot be changed (CADSWES, 2011). This functionality was not tested as part of this evaluation.

#### **2.2.4.6 Accounting and Auditing**

RiverWare has an extensive accounting system and water ownership can be accounted for within the water accounts manager. The accounting and auditing functionality was evaluated as follows:

- (i) *What queries can be made at points within the network?*

Input and output data slots can be queried from the network components. The slots within Data Components may not be site specific and thus clear naming facilitates analysis.

- (ii) *Can source and destination of water be determined?*

The RiverWare accounting system can keep track of the source and destination of water (CADSWES, 2011).

- (iii) *Can ownership of water be determined?*

Yes, the RiverWare accounting system can keep track of water ownership (CADSWES, 2011).

#### **2.2.4.7 Operational Use**

Numerous forecasting methods are available in RiverWare for Control Point, Reach and Reservoir components (CADSWES, 2011). The suitability of RiverWare for operational use was evaluated as follows:

- (i) *Does the model enable storage of state data so that a simulation can be started or restarted using a simulated or actual state for a specified point in time?*

No, not directly. This can be achieved manually in RiverWare by changing a slot records' status from output to input. The model was run from the 1<sup>st</sup> of January 1981 up to the 30<sup>th</sup> of June 1981. The records for both the reservoir levels on 30<sup>th</sup> of June were then changed to indicate that they were now input values and then the model was run for the period of 1<sup>st</sup> July to 31<sup>st</sup> December 1981.

### 2.2.4.8 Discussion

For this evaluation the methodology for setting up and running networks in RiverWare took longer to learn than MIKE BASIN for two main reasons: (i) it was not possible to attend a training course for RiverWare prior to this evaluation, so use of the model was mainly learned through the tutorials provided by the developers, and (ii) the model requires that the methods which control how water is allocated to be configured by the user. This manual configuration could be considerably quicker and offer more control in the hands of an experienced user.

An assessment of how well RiverWare met the evaluation criteria is shown in Table 2.15, with the number of times the model met the criteria being totalled. RiverWare scored high in the “User Interface”, “Flexible Configuration” and “Accounting and Auditing” evaluations. RiverWare achieved lower scores in the “GIS Functionality” , “Water Allocation”, “Scenarios” and “Operational Use” evaluations.

Table 2.15 RiverWare evaluation scores

<b>Evaluation Criteria</b>	<b>Number of Criteria</b>	<b>Score</b>
User Interface	7	6
GIS Functionality	3	1
Flexible Configuration	11	10
Water Allocation	3	2
Scenarios	1	½
Accounting and Auditing	3	3
Operational Use	2	1

### 2.2.5 Results and recommendation

The MIKE BASIN model was found to be strong on the GIS requirements but weak in the accounting and auditing functionality. It has local, South African support and is relatively easy to use. MIKE BASIN was the easiest and quickest model to configure for the test catchment. RiverWare is strong on accounting and auditing, but is weaker on the GIS requirements. RiverWare is more flexible in the way that it can be configured, but requires greater expertise. Due to the complexity of setting up RiverWare, the user support provided by the developers proved invaluable. MODSIM was weak on the GIS requirements. Some aspects of the hypothetical test catchment could not be configured within the MODSIM model in this evaluation. The lack of user support for MODSIM was its main drawback. However, it is the only model evaluated that has no cost or licence required.

The scores represented in the discussion chapters for each model were summarised by converting them to percentages as represented in Table 2.16. The average score for each model serves as a rough overall guide as to how the models scored relative to each other, but does not account for the relative importance of the different evaluation criteria. During the course of the evaluations it was noted that the level of user support and training available for the different models was an important factor for model selection.

Table 2.16 Summary of evaluation ratings

<b>Evaluation Criteria</b>	<b>MIKE BASIN (%)</b>	<b>RiverWare (%)</b>	<b>MODSIM (%)</b>
User Interface	100	86	100
GIS Functionality	100	33	33
Flexible Configuration	95	91	100
Water Allocation	100	67	100
Scenarios	50	50	100
Accounting and Auditing	33	100	67
Operational Use	50	50	50
<b>Average</b>	<b>76</b>	<b>68</b>	<b>79</b>

Although MODSIM received the highest average score, the lack of adequate user support and difficulty in configuring aspects of the evaluation river network counts against it. Based on the evaluations described in this document it was recommended that MIKE BASIN be selected for use in the project, largely due to its ease of use, strong GIS support through ArcGIS and availability of local user support and training.

### 3 REVIEW AND EVALUATION OF MODEL LINKAGE MECHANISMS

DJ Clark, A Lutchminarain and JC Smithers

Integrated water resources management (IWRM), which includes not only water quantity and quality aspects of water resources but also social and economic aspects, requires integrated water resource assessment. Assessment of complex hydrological systems often requires detailed process-oriented models (Barthel *et al.*, 2006). It is unlikely that a single model will be able to adequately represent every facet of a water resource system, which may include different scientific disciplines, different spatial and temporal scales, varying data availability and a variety of modelling objectives and stakeholders (Blind and Gregersen, 2005; Moore and Tindall, 2005; Gregersen *et al.*, 2007; Castronova and Goodall, 2010). Existing models are often developed for or have strengths in specific domains within the hydrological system and integration of models is a popular solution in attempting to model complexity (Moore and Tindall, 2005; Barthel *et al.*, 2006). Krause *et al.* (2005) state that combining and implementing approaches from natural and social sciences is a challenge to be faced in developing models and their application for IWRM. Different scientific disciplines approach system complexity and diverse scales in various ways, and use different modelling techniques and approaches to model design (Krause *et al.*, 2005). Integrated models must be compatible in terms of spatial and temporal scales and strategies for validation of both individual models and the integrated collection of models are necessary (Barthel *et al.*, 2006). Integration of models and modelling approaches for IWRM has led to research and development into integrated modelling environments, model interfacing specifications and modular modelling systems (Krause *et al.*, 2005). Integrated modelling environments typically include common data storage and formats, common data editing tools, and common spatial and temporal data visualisation and analysis tools. Integrated modelling environments are not included in this review as although they offer some degree of model integration by means of their common data formats, data repositories and analysis tools, they do not facilitate direct communication between models which is necessary for modelling feedbacks between system components. Gregersen *et al.* (2007) noted that some existing hydrological decision support systems are based on fixed combinations of specific hydrological and hydraulic models, but that the limited supported combinations sometimes resulted in compromises being made in representing the hydrological system being modelled. Krause *et al.* (2005) noted that there were two main research and development paths being followed with regard to model integration, direct integration of whole models through implementation of a model interface specification, and modular modelling systems

where modules representing individual processes are combined to create custom models. Both approaches have advantages and disadvantages.

To integrate two models it is necessary for data and information to be exchanged between the models. Linking two models in series is usually a simple matter of running the first model, converting the output format of the first model to the input format of the second model, and then running the second model. One critical limitation of the series linking approach is that feedbacks between processes in the two separate models cannot be represented. The first requirement for models to be linked in parallel is that each of the models to be linked must expose its engine such that: its variable can be read and written by third party software, and a simulation run can be initiated then executed on a time step by time step basis. If this first requirement can be met then it is possible to write custom code to link two specific models. However, a better approach would be to adopt and implement a more generic model linkage mechanism, such that any combination of models that implement this mechanism can be linked as required for a particular study. The aim of this chapter is to review and evaluate methods of linking models to gain a better understanding of model linkage mechanisms and to enable the most appropriate mechanism to be selected for use in this project. The integration of the *ACRU* model and a river network model in this project is expected to act as a test case for the integration of models representing other domains such as detailed groundwater models and socio-economic models. When evaluating the methods for linking models, the following requirements need to be considered:

- the method needs to be suitable for use with the *ACRU* model and a river network model,
- the method should require minimal changes to the code for the models,
- ideally the method should be based on a standard so that other compatible models can be linked in the future, and
- ideally the method should enable the models to be linked in parallel so that feedbacks between models can be adequately represented.

### **3.1 Review of Available Mechanisms**

As stated in the introduction, IWRM considers not only water quantity and quality, but also social and economic aspects of water resources. Often models are developed for, or have, strengths in specific domains within the hydrological system, for example, evapotranspiration, groundwater or hydrodynamic flow routing through river networks. For

IWRM it will be necessary to integrate one or more models to provide a holistic water resources modelling system for use in water resources planning and operations. An important consideration is that the models to be integrated may run at different spatial and temporal scales.

Over the years a number of strategies have been used to integrate models. As expressed by Krause *et al.* (2005), one of the simplest ways to combine models and modelling approaches from different domains or disciplines is the coupling of whole standalone models. There are many methods by which models can be coupled and these differ in their degree of complexity and the degree of interaction and feedback that can take place between the coupled models (Krause *et al.*, 2005). At the most basic level, model coupling involves using the output from one model as input for another model, Model-A could be run for say 10 years, the output files from Model-A are then reformatted to provide input to Model-B which is then run for the same 10 years. This approach is referred to as running models in series, that is, each model is run independently for the full time period one model after the other. The advantages of this approach are that it is simple and does not require any changes to the models used. There are two main problems with this approach, firstly, the effort required to reformat the output from one model to be suitable for use as input for the other model, and secondly, as stated by Krause *et al.* (2005) potential interactions and influences between the systems represented by the coupled models can only be realised in one direction, meaning that feedbacks cannot be modelled. The first problem can be overcome if both models use the same data input and output format.

A recent trend has been the development of integrated modelling environments or decision support systems such as DeltaShell (Donchyts and Jagers, 2010), LIANA (Hofman, 2005) and SPATSIM-HDSF (Clark *et al.*, 2009). These modelling frameworks have an important role to play in providing a modelling environment within which model users can operate without having to learn new user interface, data editing and analysis tools for each model, and in enabling model developers to concentrate on the science behind their models instead of having to re-invent the common functionality that is part of these modelling environments. For a particular model to be used within such a modelling environment, the model will need to be modified to read from and write to the environment's data format, but having done this would benefit from being able to use the common environment tools. Integrated modelling environments assist in standardising the way in which models are run and resolve the problem of having to translate the output data format from one model to the input data format of the receiving model, but in general, models would still have to be run in series and therefore the problem of not being able to model feedbacks between the models would still



exist, though in some cases, for example the DeltaShell environment, these environments may include some means of directly coupling models (Krause *et al.*, 2005).

One method of enabling two models to run in parallel would be to modify two or more specific models to communicate with each other either directly or via a common data repository on a timestep-by-timestep basis. When coupling two or more models in this manner, the computation order and protocols for data format and transfer have to be considered (Krause *et al.*, 2005). In order for this to work each model must have some means of being instructed to run each individual time step and there needs to be some sort of controller that commands each model or a component of each model to run for the next time step. Alternatively, for the models to communicate with each other directly they each need to provide some sort of publicly accessible interface, for example the Component Object Model (COM) interface standard, and the interface type selected needs to be compatible with the operating platform and programming language of all of the models to be linked. The .Net programming platform has, in some respects, replaced COM by enabling compatibility between software modules written in different .Net programming languages. This linking approach requires the models to be modified to implement the interface standard, which may not be possible if the models are proprietary. This approach has the advantage that feedbacks can potentially be modelled, and though the models will need to be modified, legacy models can be linked without being completely re-written and thus retain their identity and in-built integrity. A disadvantage of this approach is that, though the specific models have been linked, further modifications may be required if another model needs to be linked into the suite of models.

As noted by Krause *et al.* (2005), one of the recent model integration development paths for integrated modelling has been the development of model interface specifications such as OpenMI (Blind and Gregersen, 2005; Gregersen *et al.*, 2005; Moore and Tindall, 2005; Gregersen *et al.*, 2007; Knapen *et al.*, 2009) and the High Level Architecture (HLA) (Lindenschmidt *et al.*, 2005). A model interface standard consists of a set of software interfaces that must be implemented by the model that is to be made compliant with the standard. This concept of some sort of interface standard which must be adhered to is in some ways similar to the modularisation approach, except that it does not require the modularisation of legacy models. Implementation of the interface standard can be achieved in two ways, either by implementing the interface directly in the model code or creating a wrapper around the model. In the latter, the wrapper is compliant with the standard and has internal links to the wrapped model; however, the model may still need to be modified to some extent. Each model must declare sets of publically visible input and output variables.

Feedbacks may be modelled if the model interface standard permits this. The models would be configured individually through their respective user interfaces. Links would then be created between appropriate variables in each model. It is important to note that it is specific applications of each model that are linked, not the model engines themselves.

There are two approaches to indirectly controlling the flow of a simulation, pull mechanisms and push mechanisms. Pull mechanisms start at the point where a result is required and requests for variable values filter up through links and processes are called until the required result has been calculated. Push mechanisms start at the point where a piece of information is available and filters down through the links with each process being run when all its input variables are available. The linked model run is initiated by an external trigger. Krause *et al.* (2005) conclude that though coupling models by means of model interface standards requires some effort to adapt the models, the advantages are increased flexibility, the ability to model more complex interactions and the ability to perform more detailed analyses of the coupled models. Other advantages of this approach are that the identity and integrity of legacy models is maintained, and their marketability is improved through their ability to link to other models obeying the same interface specification. Krause *et al.* (2005) conclude that at that time the OpenMI approach to model coupling was the most sophisticated.

The other main development path for integrated modelling noted by Krause *et al.* (2005) has been the modularisation of models and the development of modular modelling frameworks such as MMS (Leavesley *et al.*, 2002), OMS (Ahuja *et al.*, 2005; Kralisch *et al.*, 2005) and LIQUID (Branger *et al.*, 2010a; Branger *et al.*, 2010b). Water resources models are typically structured into software components of some description that represent one or more hydrological processes. Thus, the concept of modularising legacy model into collections of modules representing individual hydrological processes makes a certain amount of sense. The modular modelling frameworks typically specify some sort of interface which each module must adhere to. Each module must declare sets of publically visible input and output variables. Several modules can then be linked within the appropriate modelling framework to create a custom-built model. Some sort of controller is usually required to configure the model and to coordinate calls to the various modules. The advantage of the modularisation approach is that custom-built models can be created to meet the requirements of specific modelling projects. The disadvantages of this approach are that there is a difference in the skills required by a model builder and a model user, and that the developers of legacy models must buy into one modular modelling framework. Feedbacks can be modelled if the controller and the module interface permit this, though the modularisation in itself may be sufficient for feedbacks to be modelled.

Each modelling exercise and combination of models will have unique requirements and the most appropriate linking mechanism will have to be selected for each case. The requirement for integrated assessment of water resources and advances in computer programming technologies has resulted in numerous innovative endeavours to provide the ultimate modelling system. This is evident from the proceedings of the International Environmental Modelling and Software Society (iEMSs) 2010 International Congress on Environmental Modelling and Software, from which many of the papers referenced in this review were obtained. There does not appear to be any ultimate modelling system that stands out above the rest and, as a general rule, each individual research or commercial development group tends to continue along its own development path as it has control over its software and is able to continue development to meet its own specific needs. For this project it is important to keep in mind that adoption of a standard would be preferable to buying into a proprietary system, as this project is meant as a test case for future model linking exercises.

Several model linking mechanisms from the interface specification and modular modelling system and approaches are reviewed below. The mechanisms reviewed were those that appeared from the literature to have undergone on-going development and implementation. Some of the requirements were that the linking mechanism should:

- be suited to water resource type models;
- be capable of linking models with different spatial and temporal scales;
- be applicable to most water resource type models;
- enable parallel processing to account for feedbacks between modelled components to be represented;
- require minimal changes to the models in which it is implemented;
- ideally enable linking of models written using different software platforms;
- have a minimal impact on the speed of model runs;
- ideally be based on some sort of standard or be regarded as a standard in its own right;
- be adequately supported by the developers of the linking mechanism;
- have been widely applied and adopted by developers of water resources models;
- not place a large financial burden on users of the models in which it is used; and
- be suitable for linking the *ACRU* model and the selected flow network model.

### 3.1.1 Open Modelling Interface (OpenMI)

#### 3.1.1.1 Overview

The purpose of OpenMI is to provide a standard to facilitate the linking of models, operating at various spatial and temporal scales, and to enable new and existing models to interact with each other to represent catchment process interactions (Blind and Gregersen, 2005; Moore and Tindall, 2005; Gregersen *et al.*, 2007). Gregersen *et al.* (2007) define OpenMI as “a standardised interface to define, describe and transfer data on a time basis between software components that run simultaneously, thus supporting systems where feedback between the modelled processes is necessary in order to achieve physically sound results”.

Initial development of OpenMI was completed in the HarmonIT project funded by the European Commission as part of the Water Framework Directive (2000/60/EC) approved by the European Parliament and Council in 2000 (Blind and Gregersen, 2005; Moore and Tindall, 2005). Hutchings *et al.* (2002) conducted an extensive review of the state of the art with regard to model integration as part of the HarmonIT project, and the conclusion of the review was that none of the existing model integration initiatives met the requirements of the Water Framework Directive. Hutchings *et al.* (2002) found that many of the existing developments did not seem to have been used outside of the institution that developed them, suggesting that their design and functionality were targeted to the developers own needs. Version 1 of OpenMI was launched in 2005 with the aim that it would become a worldwide standard for linking environmental models and tools, though the initial focus had been on water resources modelling (Gijsbers *et al.*, 2010). Subsequent to the launch of OpenMI, the OpenMI Life project under the EU-Life programme was initiated to transform OpenMI from a research output to an operationally viable standard (Gijsbers *et al.*, 2010). Gijsbers *et al.* (2010) report that an organisation known as the OpenMI Association was established in 2007 as a legal entity, to take ownership of, support and promote the OpenMI standard and associated software utilities and tools (Gijsbers *et al.*, 2010). The website for the OpenMI Association can be found at <http://www.openmi.org>. Since the inception of the OpenMI Association, additional model and software development groups have joined the association and created new implementations of OpenMI. Based on feedback from members of the association regarding their experiences with implementing OpenMI and requests for additional features, the design of OpenMI was refined and further developed, resulting in Version 2.0 being released (Donchyts *et al.*, 2010; Gijsbers *et al.*, 2010). This review is largely based on Version 1 of OpenMI as more literature was available for this version, and the changes to OpenMI to create Version 2 are only mentioned briefly.

The HarmonIT project acknowledged that there were a variety of approaches to model integration, and the two criteria that directed their chosen solution were, that it should be suitable for application to existing models, as it would not be feasible or desirable to recode a large number of existing models, and that the time, skill and cost required to implement it in a model should not be a deterrent to its use (Moore and Tindall, 2005). Gregersen *et al.* (2007) state that OpenMI is designed to be easily implemented in existing models and modelling systems, in which substantial investment in development and testing has already taken place, and for which recoding may not be an option for economic reasons. The objective was to create a model linking architecture that, due to its high quality and wide support, would become a European standard, or even a world standard, for linking water and environmental models (Moore and Tindall, 2005; Gregersen *et al.*, 2007). The project aimed to simplify the linking of models to enable better modelling of process interactions, enable simple swapping of linked models to facilitate sensitivity studies and benchmarking, enable representation of feedbacks and process interactions, reduce development time for decision support tools, provide model users with a wider selection of models, and provide model developers with a bigger market (Moore and Tindall, 2005). One of the strengths of the project was that the development team comprised individuals from a broad collection of end users, research organisations and several usually competing commercial modelling software developers, representing 14 organisations and 7 countries (Blind and Gregersen, 2005; Moore and Tindall, 2005; Gregersen *et al.*, 2007). Gregersen *et al.* (2007) noted that it was significant that three significant and usually competitive commercial partners, DHI – Water and Environment, WL – Delft Hydraulics and Wallingford software had contributed to the development and promotion of OpenMI. Blind and Gregersen (2005) make the point that this broad collaboration was important if the objective to create a standard was to be achieved.

In order to promote wide adoption of the OpenMI standard, Gregersen *et al.* (2007) state that one for the general requirements for the OpenMI architecture, was that implementation of the standard in models should be cost effective, and at the same time enable model developers to create their own software solutions around the standard. They also state that it was important that legacy models should not be required to include an unnecessary amount of OpenMI related code and that implementation of the standard should not require maintaining two versions of a model. These requirements indicated the need for a lean standard that is, in essence, just an interface definition, but which is supported by software libraries that include wrapper classes and other tools. Moore and Tindall (2005) list the following key requirements identified for the development of OpenMI:

- ability to link models from different domains, such as hydraulics, hydrology, ecology, water quality and economics;
- ability to link models from different environments, such as atmospheric, freshwater, marine, terrestrial, urban or rural;
- ability to link models based on different concepts, for example, deterministic and stochastic models;
- ability to link models with different dimensional representations (0, 1, 2, 3D);
- ability to link models working at different spatial and temporal scales;
- ability to link models using different projections, units and categorisations;
- ability to link models to alternative data sources, such as databases and monitoring equipment;
- ability to link both new and existing models with the minimum of code changes by people and without requiring unreasonably high level IT skills;
- not adversely affect model performance;
- based on proven and available programming technologies;
- the architecture should be component based and multi-layered;
- enable linking of models developed in different programming languages and running on different operating platforms and networks; and,
- that the model interface specification should be placed in the public domain.

Gregersen *et al.* (2007) and Moore and Tindall (2005) define the following terms used within the OpenMI documentation:

- *model application* – all the parts of a modelling system's software that is installed on a computer, which typically include a user *interface* and an *engine*;
- *engine* – generic representation of a process or processes, consisting of the algorithms or calculations used to model the process or processes;
- *user interface* – graphical or command line tools that enable the model user to specify or input data required by the *engine* and which describe a specific scenario to be simulated by the process;
- *model* – when the *engine* is run it reads the data for specific scenario to be simulated and becomes a *model* of the system for which the simulation is being run (a *model* is an *engine* that has been populated with data);
- *engine component* – and *engine* becomes an *engine component* if can be instantiated as a standalone software entity and has a well-defined interface enabling it to accept and provide data;
- *model component* – an engine component that has been populated with data;

- *linkable component* – if the *engine component* implements the OpenMI standard interface then it becomes a *linkable component*, and can be linked to other *linkable components*;
- *quantity* – a *engine variable* whose value can be accepted or provided during an exchange between models;
- *element* – a location at which a quantity is calculated, for example, a catchment or a river reach;
- *migration* – the process of implementing the OpenMI interface standard in an *engine*.

The OpenMI standard interface has two functions, a descriptive configuration-time function, in that it defines what items a linkable component can accept or provide and which of these items will be used in a particular modelling scenario to exchange data between linked models, and a runtime function, in that it defines the means for a model to request and accept exchanged data values at runtime (Moore and Tindall, 2005). Gregersen *et al.* (2007) explain that the OpenMI standard is a software component interface definition that can be implemented in the computational core of water resource models to enable their integration, and that without any additional programming these models can be configured to exchange data during model execution.

Blind and Gregersen (2005) state that the main problem to be overcome in the design of the OpenMI interface standard was how to enable the exchange of data between models, databases and other modelling software tools in a runtime environment, and that the solution was based on two principles, component based design and the '*GetValue*' mechanism for runtime data exchange. They explain that component based design implies that OpenMI compliant models, databases and tools must be structured as standalone components which all have a common set of properties methods and events. Existing models can be made OpenMI compliant by implementing the OpenMI interface standard directly or by creating a wrapper around the model. They explain further that the '*GetValue*' mechanism is pull driven, which means that the requesting model requests the requested model for a set of quantity values for a given time for one or more specified locations. Gregersen *et al.* (2007) describe this as a 'request and reply' mechanism, and that OpenMI has a 'pull-based pipe and filter' architecture made up of linked source and target components that exchange memory-based data in a predefined format and manner. They state that OpenMI is a purely single threaded architecture in which each linkable component can handle just one data request at a time. During a model run, data exchange is initiated by one component at the end of the chain of linked components, and after this the linkable components continue to

exchange data without any external supervision till the end of the simulation period reached. Gregersen *et al.* (2007) make the point that OpenMI is not based on a framework, it consists entirely of linkable components which exchange data directly at runtime. The standard interfaces are described by Gregersen *et al.* (2007) and are shown in Figure 3.1. The OpenMI standard includes a standard interface *ILinkableComponent* that must be implemented by each engine component to become an OpenMI *LinkableComponent*. Each *LinkableComponent* contains a list of *InputExchangeItem* objects and a list of *OutputExchangeItem* objects, in other words a list of the data inputs it requires, and a list of the data outputs it can provide. Each *InputExchangeItem* contains a *Quantity* object and an *ElementSet* object specifying what can be accepted at which locations. Each *OutputExchangeItem* contains a *Quantity* object and an *ElementSet* object specifying what can be supplied at which locations, and also a set of *DataOperation* objects describing details of how the exchanged data is to be calculated. A *Quantity* object describes what type of data is to be supplied or received, for example, water level or flow rate. An *ElementSet* object specifies the locations (physical entities) to which a *Quantity* applies. Two instances of *LinkableComponent* can only exchange data if they are linked by one or more *Link* objects which implement the *ILink* interface. A *Link* object contains information about the *InputExchangeItem* object and *OutputExchangeItem* object of the target and source *LinkableComponent* objects respectively.

Moore and Tindall (2005) provide the diagrams shown in Figure 3.2 and Figure 3.3 to help illustrate how the linking of two models works in simple terms. Two OpenMI compliant model applications, a rainfall-runoff model application and a river flow model application are shown in Figure 3.2, each with its own user interface and input datasets. As shown in Figure 3.3 each model engine declares the variables which it can accept and provide. In this example the rainfall-runoff model can provide runoff which can be used by the river flow model to satisfy its requirement for lateral inflow. These linkages between models must be specified by the user before the linked models can be run. These linkages must be configured for each model element, in other words, for each catchment and river reach in this example. One of the models must be nominated as the trigger to start the data exchange process, and in this example the river flow model would be triggered to start running, at its first simulation time step it would determine that it requires a value for the lateral inflow variable before it can proceed and so would call the *GetValues* method for the runoff variable in the rainfall-runoff model which would initiate calculation of runoff for the appropriate time step in the rainfall-runoff model. The runoff value for the requested time step would be returned to the river flow model which can then proceed with its calculations



for the first time step and this sequence of events would be repeated for each time step in the river flow model till the simulation is complete.

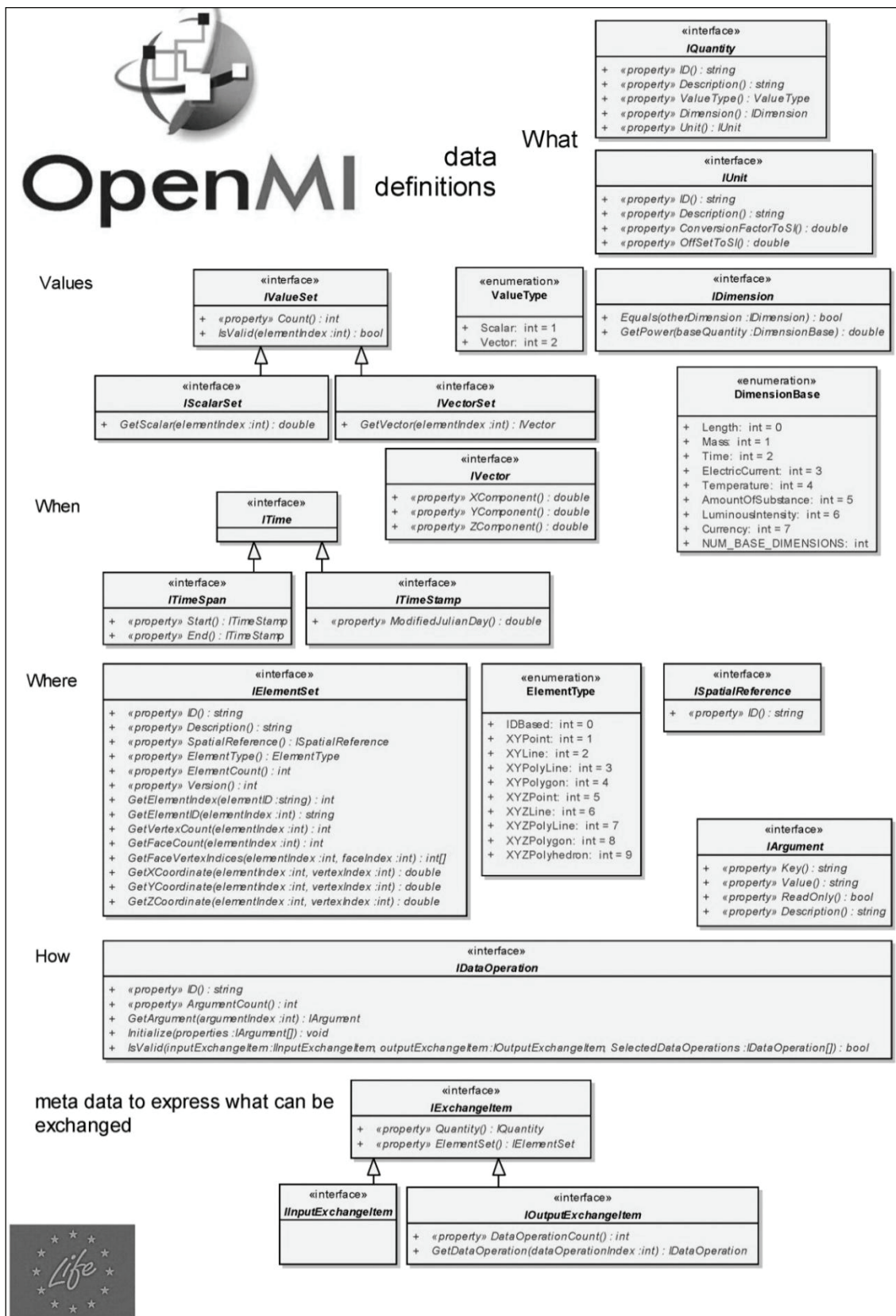
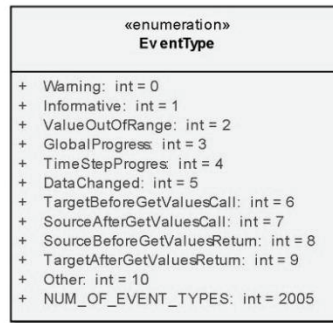
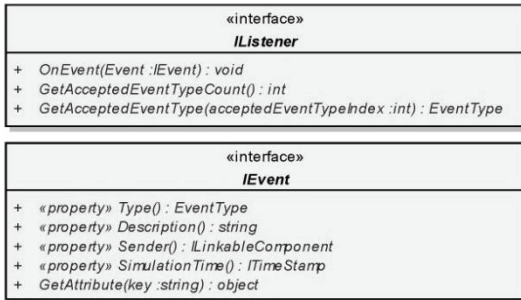


Figure 3.1 The OpenMI Version 1.4 standard interfaces (OpenMI, 2011a)

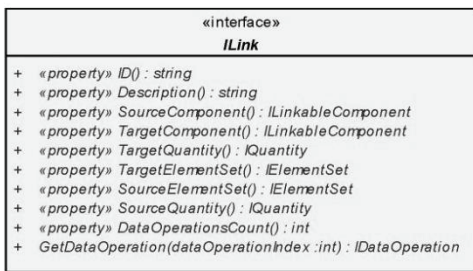
# org.OpenMI.Standard interface specification v.1.4.0

December 2007 © The OpenMI Association URL: www.openmi.org

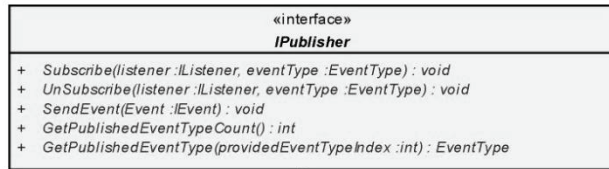
messaging definitions



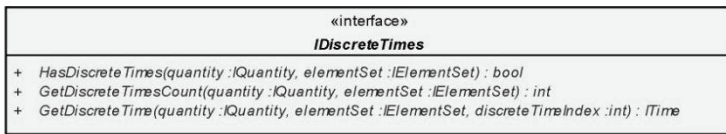
specification what will be exchanged and how



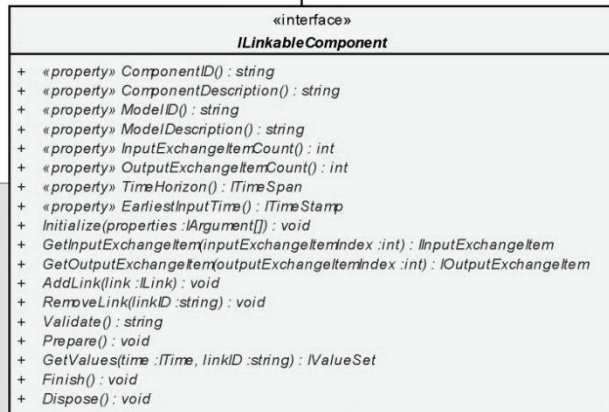
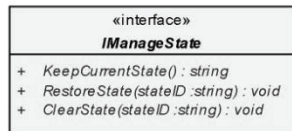
component interfaces for generic component access



optional extensions



obligatory interface



starting point:  
the OMI-file

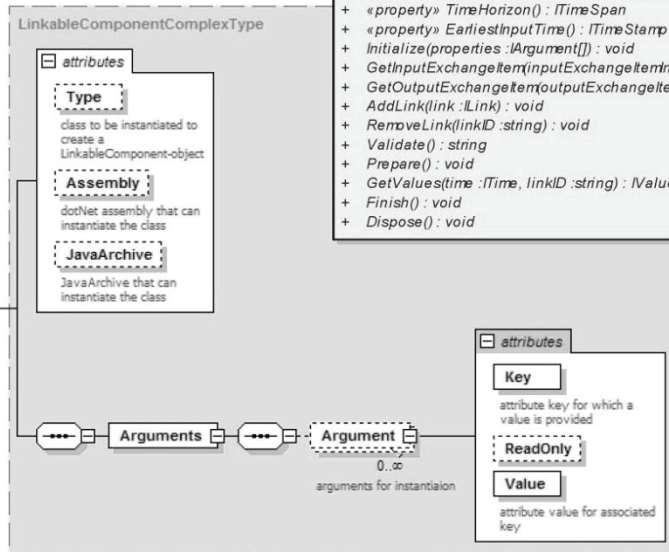


Figure 3.1 (continued) The OpenMI Version 1.4 standard interfaces (OpenMI, 2011a)

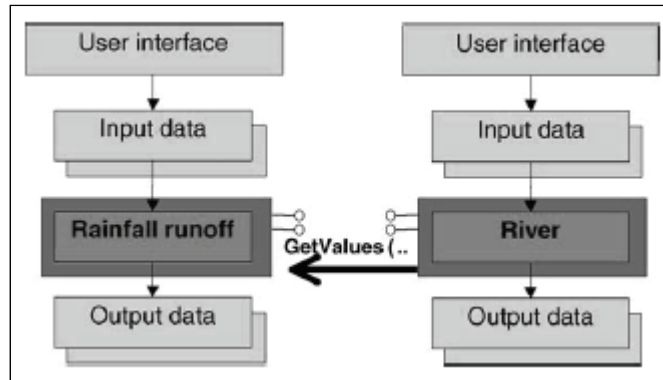


Figure 3.2 Example of two model applications linked after implementation of the OpenMI interface standard (Moore and Tindall, 2005)

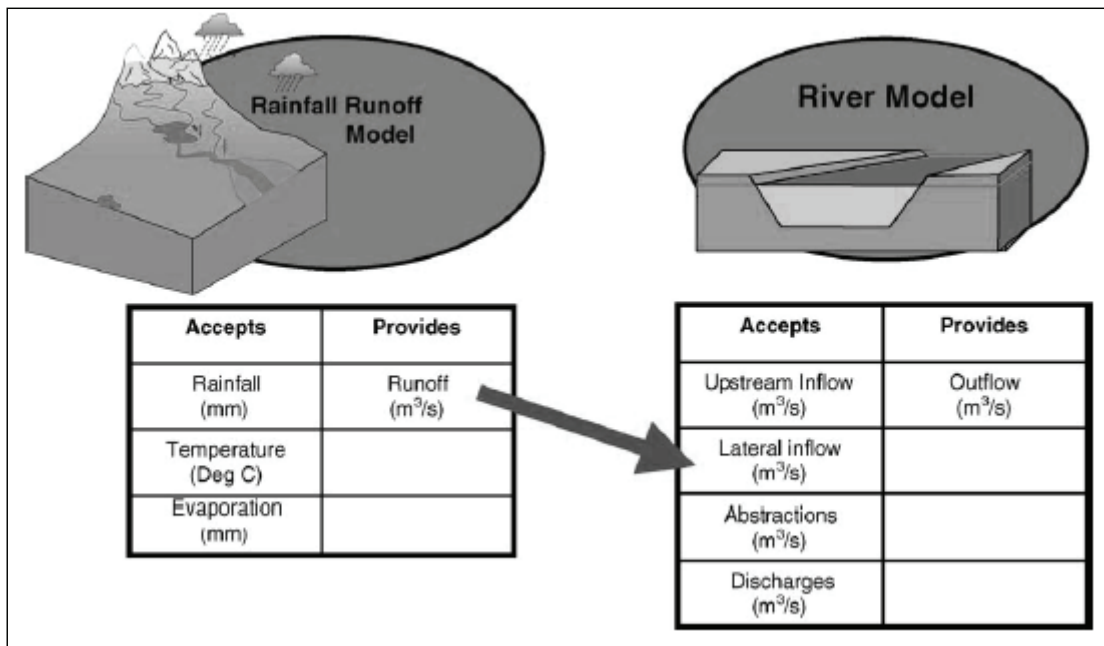


Figure 3.3 Example of the exchange of flow quantities between the two linked model applications shown in Figure 3.2 (Moore and Tindall, 2005)

The *GetValues* method is the key with regard to data exchange between models at runtime, and the way this works for typical modelling situations, including feedbacks, is shown in

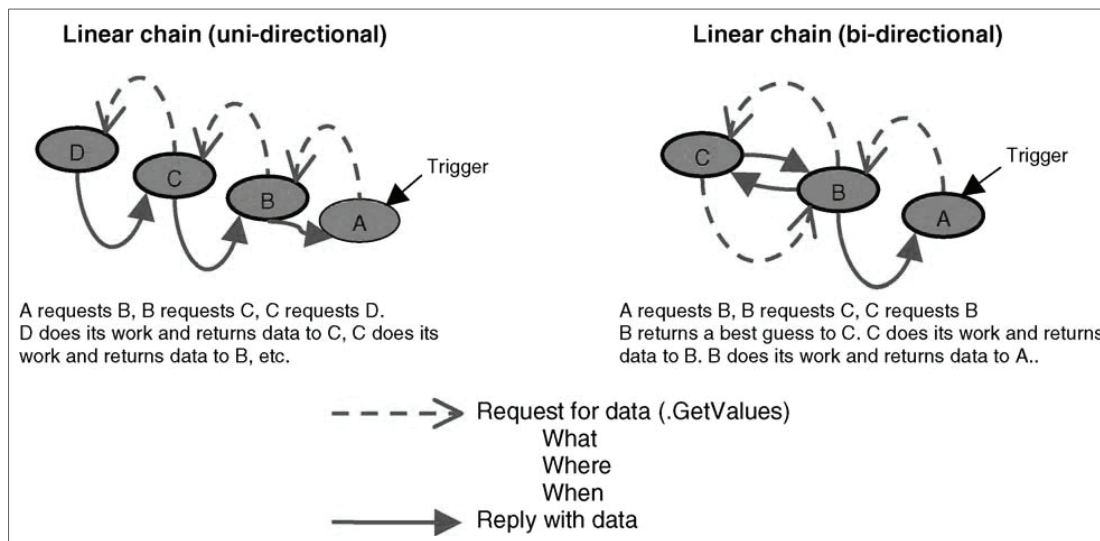


Figure 3.4 and explained by Moore and Tindall (2005). There are several scenarios that could arise when a model receives a request for data. First, if the requested data already exists or has already been calculated for the relevant time step, then it is simply returned to the model that requested it after any necessary, interpolation, unit conversion and mapping operations have been performed. Second, if not available, the requested model will run for one or more time steps, if necessary requesting other data inputs from other models, until it can return the requested value. In this case if the requested model cannot run, possibly because the data inputs it requires are not available, then it will attempt to return a data value extrapolated from existing previous values. Third, if the requested model in turn requires data from the requesting model, as is the case for backwater calculations, then iteration is required to reach a solution. OpenMI compliant models are required to save their status at each time step and if required be able to revert back to the status at a specified time step. Moore and Tindall (2005) make the point that the *GetValues* method handles requests for data strictly in order of request which prevents the calculation sequence becoming confused.

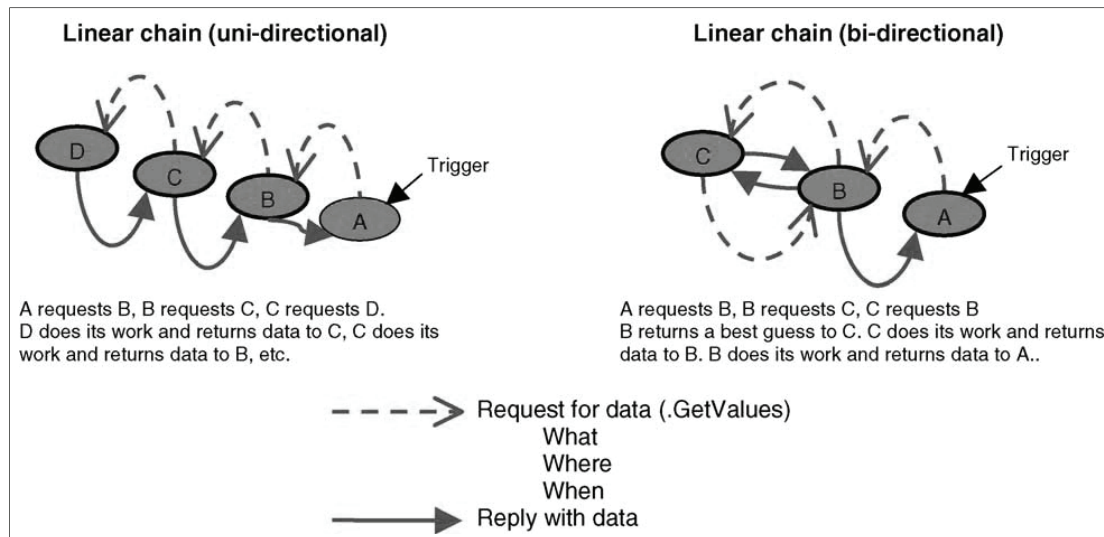


Figure 3.4 Illustration of data exchange between OpenMI compliant models using the *GetValues* method (Moore and Tindall, 2005)

The OpenMI architecture design is described by Blind and Gregersen (2005) as a layered architecture, and consists of two components, the OpenMI standard and the OpenMI environment, as shown in Figure 3.5. Moore and Tindall (2005) explain that the OpenMI standard is the definition of the model interface standard that an engine component must implement in order to be OpenMI compliant, and that this interface standard has been placed in the public domain. Blind and Gregersen (2005) describe the standard itself as being the system layer, which consists of a description of the mechanisms, interface definitions, model definitions and exchange variables, which is all that is required to work with the standard. Gijsbers *et al.* (2010) state that the OpenMI Association does not intend to develop a modelling framework, the OpenMI standard is its primary responsibility, but that it is necessary to provide an open source software implementation of the standard to promote wide adoption of the standard. The OpenMI environment consists of a collection of software tools and utilities that may be used by model developers to assist in making model engines OpenMI compliant and facilitate linking and running of OpenMI compliant models. The OpenMI standard consists of a set of interface definitions which must be implemented in code to create OpenMI compliant models. The OpenMI environment includes a backbone package (`org.OpenMI.Backbone`) that contains a default C# code implementation of each interface in the standard (Gregersen *et al.*, 2007). Blind and Gregersen (2005) explain that the OpenMI environment also includes three supporting configuration, utilities and tools layers. The configuration package (`org.OpenMI.Configuration`) provides tools to facilitate linking two or more models, and running the linked models. These tools include a graphical user interface, but use of this user interface is not prescribed and it is likely that developers of modelling frameworks will create their own custom user interfaces (Gregersen *et al.*,

2007). The utilities package (org.OpenMI.Utilities) provides utility classes, includes a default implementation of the *ILinkableComponent* interface, containing useful facilities for use when wrapping models, for example, keeping track of model links, converting units of measure, buffering, and interpolation or extrapolation between different spatial and temporal scales. The tools package (org.OpenMI.Tools) provides a set of useful tools including, visualisation tools, logging tools, optimisation controllers and iteration controllers. Versions of these tools and utilities were developed in C# and Java and are available as open source, and SOAP/WebServices were used to provide for communication across networks. Jagers (2010) states that although both the C# and Java implementations use only a single execution thread, this is not a requirement of the OpenMI standard which has been shown to be compatible with remote and multithreaded engines, and web services. The org.OpenMI.DevelopmentSupport package contains a generic set of low-level classes that can be used in the development of an OpenMI modelling environment. All aspects of the architecture were documented during HarmonIT project. Testing of the OpenMI interface standard and environment was done first by the design and development team, and then by an implementation team who implemented the standard in a wide range of models and ran linked models for a list of use cases developed to demonstrate its capabilities. The OpenMI interface standard, the OpenMI environment and all the accompanying documentation have been made available to the public as open source (Gregersen *et al.*, 2007).

Blind and Gregersen (2005) state that when migrating a legacy model the primary requirement is to ensure compliance with the OpenMI interface standard, however, there are many ways of going about doing this. One of the first decisions that need to be made is whether to migrate the whole model or individual functional components with the model. Blind and Gregersen (2005) state that smaller functional components are preferred, but larger components may be necessary for practical, financial and commercial reasons, though the larger component could itself be internally OpenMI compliant. Second, it needs to be decided for which model variables input or output data exchange items are required, where this will depend to a large extent on what the model is to be used for. Although the design of the OpenMI interface standard permits each model to run at its own spatial and temporal resolution, the *GetValue* mechanism requires, within reason, that model outputs be delivered at any requested time and location. Gregersen *et al.* (2005) estimate that, depending on how well structured the model code is, migration of a model to OpenMI will take from a few weeks to a few months to perform.

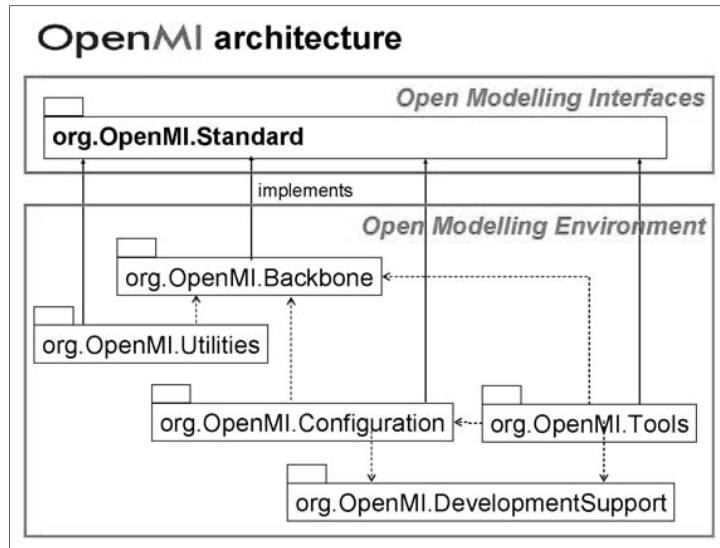


Figure 3.5 OpenMI architecture namespaces (Moore and Tindall, 2005)

Donchyts *et al.* (2010) explain that Version 2 of the OpenMI standard includes several new goals, some of these are technical in the sense of applying certain useful and established information technology concepts in the design of the standard to improve its quality and make it more intuitive, and some are intended to extend the scope of the standard to a wider set of applications than previously envisaged. Some of the main changes to Version 2 of the OpenMI standard are described by Donchyts *et al.* (2010) and Gijbers *et al.* (2010). The *ILink* interface has been removed as, in some respects, it was not very intuitive or efficient and ownership of Link objects was not clear. The connections between source and target *LinkableComponent* objects are defined in Version 2 by a provider/consumer relationship as a direct reference between an output exchange item and an input exchange item defined using new *IInput* and *IOutput* interfaces which are subclasses of *IExchangeItem*. The exchange items have taken on many of the responsibilities *ILink* responsibilities and are now completely self-contained. This arrangement means that, for example, a data provider object does not need to implement *ILinkableComponent* to be coupled to a *LinkableComponent*. In the *ILinkableComponent* the "Perform Calculation" and the "Query Value" steps of have been separated, the *ILinkableComponent* now includes an *Update* method which can be called independently, previously calculated values can be retrieved from an instance of *IOutput*, and the *GetValue* method can return a subset of the values available from the source exchange item. The *IDataOperations* interface used by the *IOutputExchangeItem*, has been replaced by a simpler *IAdaptedOutput* interface which can be wrapped around an output exchange item to perform data conversion operations between the data source and target. The pull mechanism of data flow control still exists, but a loop mechanism has been included as a different approach to running components which

enables control programs to be developed to be developed to, for example, run components in different threads or on different computers. Other changes include changes in the handling of spatial references, time and state data. Jagers (2010) states that the changes made in Version 2 of OpenMI will make it easier to use with implementations that are not models, for example in databases.

### 3.1.1.2 Application

As part of the development of OpenMI in the HarmonIT and OpenMI Life projects, the standard was implemented in numerous models and modelling tools. A list of models and other software known by the OpenMI Association to be compliant to OpenMI Version 1.4 is shown in Table 3.1. It will take some time for existing OpenMI compliant models to be updated to OpenMI Version 2. Gijbbers *et al.* (2010) report several new implementations of OpenMI including Delft3D, SWAT, HEC-RAS, Modflow, WaterOneFlow, Frames 3 and notably OMS. The following applications of OpenMI were found in the literature:

- In a study by Christensen (2004) the MIKE SHE hydrological model written in FORTRAN 90 and FORTRAN 77 was linked to the MIKE BASIN river management model written in C++. The implementation of OpenMI is reported to have required relatively little re-engineering of code in these models and enhanced the flexibility and applicability of the models. The performance of the OpenMI linking mechanism compared to other mechanisms was not evaluated.
- Reußner *et al.* (2009) report on the implementation of OpenMI in the SMUSI sewer system model and the BlueM.Sim rainfall-runoff and receiving waterbody model. Both models were coded in the FORTRAN 90/95 programming language. The OpenMI coupled system was found to significantly reduce the time and effort required for model configuration, compared to the previously used manual coupling procedure, and worked well for the case study. Reußner *et al.* (2009) state that the ability to reuse existing models and datasets is a significant advantage of the OpenMI approach. They state that another advantage of OpenMI is the scalability of the interactions between models and that model can be easily interchanged.
- Bulatewicz *et al.* (2010) report on the integration of agricultural (EPIC), groundwater and economic models using OpenMI. Wrappers written in C# were used to wrap the models written in FORTRAN or Scilab. OpenMI was selected as it represented a standard linking protocol as opposed to a specialised one, promotes collaboration, enables a model to be linked to a variety of other models and the integration of these models to be rapidly reconfigured. Bulatewicz *et al.* (2010) highlight the importance of model validation, not only the individual models but also the integrated models,



especially in instances where feedbacks occur. They conclude that the software development effort was minimised by being able to reuse existing models through the OpenMI linking mechanism, especially through utilising a standard which will enable the now OpenMI compliant models to be used in other studies.

- Castronova and Goodall (2010) investigated component-based modelling, in other words the modular modelling approach, and believe that OpenMI can serve as a foundation for a loosely coupled, component-based structure to enhance more tightly coupled approaches such as ESMF, CSDMS and OMS. They recognised that OpenMI was mainly intended for integration of legacy models but used it to create the Simple Model Wrapper (SMW). The SMW is intended to simplify the use of OpenMI for model developers and promote the development of process based model components.
- Makropoulosa *et al.* (2010) present a study in which two integrated modelling systems using different models, but both using OpenMI, were developed in parallel by two research groups for the same case study. Both modelling systems include hydrology, hydraulics and water quality models; MIKE-11(NAM), MIKE-11 and OTIS in one case and, MIKE-11(NAM), RISH-1D and RISQ-1D in the other case. MIKE-11 was already OpenMI compliant and OpenMI was implemented in the other models. The individual and integrated model runs corresponded well in both studies, demonstrating that integration using OpenMI did not adversely affect the accuracy of results from individual models. Not surprisingly, there were differences between the results from the two different modelling systems, but this highlighted the advantage of using OpenMI, which enables models to be compared by swapping them in and out of a suite of integrated models.

Table 3.1 A list of models and other software that are OpenMI Version 1.4 compliant (after (OpenMI, 2011b))

<b>Provider</b>	<b>Component</b>	<b>Description</b>
British geological survey & University of Birmingham	ZOOMQ3D	Finite-difference groundwater flow model
Wallingford Software Ltd	InfoWorks CS 10.0	Hydrological modelling for the urban water cycle
	InfoWorks RS 10.0	Flow simulation for rivers, channels and floodplains
	InfoWorks RS WQ 10.0	Water Quality analysis for rivers, channels and floodplains
	InfoWorks CS 9.5	Hydrological modelling for the urban water cycle
	InfoWorks RS 9.5	Flow simulation for rivers, channels and floodplains
	InfoWorks RS WQ 9.5	Water Quality analysis for rivers, channels and floodplains
TU Darmstadt – Section of Engineering Hydrology and Water Management	SMUSI.OpenMI	Hydrologic runoff and pollution load model (urban sewer systems)
	BlueM.Sim	Hydrological modelling
	BlueM.Analyser	Monitoring and evaluation tool (IListener)
Halcrow Group Ltd	ISIS Professional v.3.1	River and flood risk modelling system
	ISIS Free v.3.1	River and flood risk modelling system (free version)
UNESCO-IHE Institute for Water Education	SWAT, version IHE	River basin modelling tool for soil, water and pollution
National and Technical University of Athens	RiSH-1D	Fortran Hydraulic River Model
	RMM-NTUA	Delphi Reservoir Management Model
Deltares	Sobek-Rural-CF	0/1D hydraulic simulation software for rural applications
	Sobek-RE	1D hydraulic simulation software for Rivers and Estuaries
Dutch Rijkswaterstaat, Waterdienst	Waqua, version Simona0811	2D/3D hydraulic simulation software for Seas, Rivers and Estuaries
BAW, Bundesanstalt für Wasserbau	GEI	Generic access to initial and boundary condition data files
LicTek	RegularGrid	Facilitates testing of exchange items with ElementSets of type XYPolygon
DHI	MIKE 11	Hydraulic and hydrological model for river flow
	MIKE SHE	Integrated GroundWater/Surface Water model
	MIKE URBAN	Industry standard in modelling water distribution and urban drainage networks
KISTERS AG	WISKI-KiTSM	Water Information System KISTERS

### 3.1.1.3 Comments

Blind and Gregersen (2005) state that at the time OpenMI was initially developed, none of the existing modelling frameworks or integration systems appear to have been successful in being recognised as a standard. They believe that OpenMI is not just another attempt, for the reasons that, three significant commercial hydrological modelling groups have invested expertise, time and funds, the organisations that participated in the development all recognise the need for collaboration and flexible modelling solutions, from an early stage the design was open to comment from all interested parties and the project was widely supported by leading experts in the field, the development was well documented, during the project a significant collection of models were made OpenMI compliant, and that a collaborative organisation was being put in place to maintain and further develop OpenMI beyond the HarmonIT project. Gregersen *et al.* (2007) confirmed that an OpenMI association was being formed to support the OpenMI user community, that it would be responsible for the maintenance and further development of OpenMI and would be an open forum to the public. Gregersen *et al.* (2007) make the point that standards enable people to work together, by providing a common 'language', but will only be successful if technically sound, widely used, well supported and further developed in response to user needs. The technical soundness of OpenMI has been proven through extensive testing on a wide range of use cases, including complex systems (Gregersen *et al.*, 2007). Gregersen *et al.* (2007) correctly state that the usefulness, and therefore success of the OpenMI interface standard, will depend on the number and availability of OpenMI compliant models. Approximately 20 models, including some widely used commercial water resources models, were migrated to OpenMI as part of the HarmonIT project and subsequently other model developers have started migrating additional models (Gregersen *et al.*, 2007). Gijbbers *et al.* (2010) strongly believe that Version 2 of OpenMI is a major advance in enabling application of OpenMI to support interoperation between models, GIS applications, databases and web-services.

OpenMI is primarily a model coupling interface specification. Though default code implementations are provided in both Java and C#, model developers are not required to use these implementations. Model developers may either implement the OpenMI interface directly in a model or by means of wrapper classes for which examples are provided. One possible negative aspect of OpenMI is that it does not directly support linking between a Java and a .Net implementation of the interface, though wrappers may provide a solution to this. OpenMI does not provide any of the traditional modelling framework tools, such as graphical user interfaces and data analysis tools, and the use of OpenMI in the DeltaShell (Donchyts and Jagers, 2010) modelling framework is an interesting development. OpenMI

was originally designed with the intention that it would be used to link legacy models, but there is no apparent reason why it should not be used in a modular modelling context. The OpenMI interface specification and associated code implementation are well documented compared to almost all of the model linkage mechanisms mentioned in this review. OpenMI was and is being developed by a diverse collaborative team and is being maintained and further developed by the OpenMI Association. OpenMI has been widely implemented, as evidenced by literature and the documentation, and seems to be gaining momentum, though the non-backwards compatibility of OpenMI Version 2 may result in a delay in already compliant models being migrated to the new version. Based on the wide acceptance of OpenMI, its good documentation and the existence of the active OpenMI Association, OpenMI could be considered to be a standard for coupling of environmental models. OpenMI is open source and available in the public domain.

### **3.1.2 Object Modelling System (OMS)**

#### **3.1.2.1 Overview**

The Object Modelling System (OMS) is described by David *et al.* (2010) as a framework for developing environmental models, including facilities for data provision, testing, validation, and deployment. Ahuja *et al.* (2005) describe OMS as a modular modelling framework which enables single- or multi-process modules to be implemented, and then compiled and applied as custom models. Kralisch *et al.* (2005) elaborate that OMS is based on a concept where all system and model components are represented as independent reusable modules which are coupled by standardised software interfaces. David *et al.* (2004) explain that OMS consists of a library of process, control and database access modules, tools to assemble selected modules into a custom model, supporting utilities for data retrieval, and GIS, statistical and graphical tools for data analysis.

Development of OMS started in 1996 at Friedrich Schiller University (FSU) in Jena, Germany, and since 2000 development continued at the USDA-ARS Great Plains Systems Research Unit (Fort Collins, CO) and the USGS (Denver, CO), jointly with FSU (Ahuja *et al.*, 2005; Kralisch *et al.*, 2005). David *et al.* (2010) state that OMS was first released in 2004 and report on recent development of the framework resulting in Version 3 of OMS. OMS appears to support open source development and is available from <http://oms.javaforge.com>.

David *et al.* (2004) describe OMS as a Java based introspecting simulation framework, which uses metadata read from annotations in the code of the modules, where these annotations specify the spatial and temporal constraints of the module, and metadata about data variables and parameters which are used for range validation, unit conversion and automated testing. OMS uses this metadata during creation of models to ensure correct assembly with respect to spatial and temporal scales, and at model runtime to provide data linkages. As explained in David *et al.* (2004), modules, which are called “Components” in OMS are the building blocks used to create custom models and usually represent a unique concept within a model, for example, a physical process, a management process or a remote data input. A comprehensive set of metadata requirements are specified by OMS to be declared by components, which provide a full definition of each component. There are two levels of metadata annotation required for each component, the first being component metadata, which includes information about the component’s purpose, author, version, references to relevant literature, spatial and temporal scale. The second level is attribute metadata for each public attribute of a component, such as units of measure, data range and default values. Jagers (2010) explains that each component is a plain Java class, with an execution method and optional initialization and finalization methods, where input variables, output variables and methods are identified and described using Java annotations, for example, *@In* for input variables, *@Out* for output variables, *@Unit* for unit of measure, and *@Execute* for the execute method. OMS is built on top of the NetBeans platform which is a Java based open source software framework for building desktop application software, which provides OMS with access to NetBeans features such as GUI components, data storage access components and help file components (Ahuja *et al.*, 2005; Kralisch *et al.*, 2005). Ahuja *et al.* (2005) explain that OMS uses a dictionary framework which they describe as the “knowledge-backbone” of OMS. These data dictionaries are implemented in Extensible Markup Language (XML) and are used to specify parameter sets, model control information and details of component connectivity. Ahuja *et al.* (2005) mention that OMS includes some generic software tools that may be used to assist in extracting components from existing non-modular simulation models, and include them in the OMS framework.

The conceptual layout of OMS is shown in Figure 3.6. Kralisch *et al.* (2005) explain that there are two types of components in OMS: model components which are the building blocks from which models are created, and system components which are used to assemble and execute model components. The system component are used to provide the selected model components to a Model Builder which assembles them to create an application specific model that may be populated with data and then executed within the Runtime Environment. System components are comprised of system core, model builder, update centre and user

interface groups of components which are responsible for all coupling and execution of model components. The system core provides the environment for component development and execution and the model runtime environment. It also provides basic functionality for all other components, implements a set of simple and complex data types including temporal data types that may be used in model components and provides objects such as the OMS components which are used as base classes for implementing model components. The model builder provides a GUI that enables individual model component to be assembled and configured to create custom models, including mapping component input and output parameter linkages. The documented input and output parameters for each available model component are exposed enabling users to link the output parameter from one component to the input parameter of another component. The model builder enables storage and management of custom models which can be shared with other users and executed in other computing environments. The update centre is a standard Net-Beans tool that enables existing OMS components to be downloaded or updated via the internet. OMS model components can be encapsulated within NetBeans modules containing model components, parameter-sets and documentation. There are facilities for shared components to be protected by licence agreements. OMS provides a collection of GUI components including GIS and graphing facilities for developers and users to display modelling results, and additional user interface components may be added by implementing them as NetBeans modules within the OMS framework.

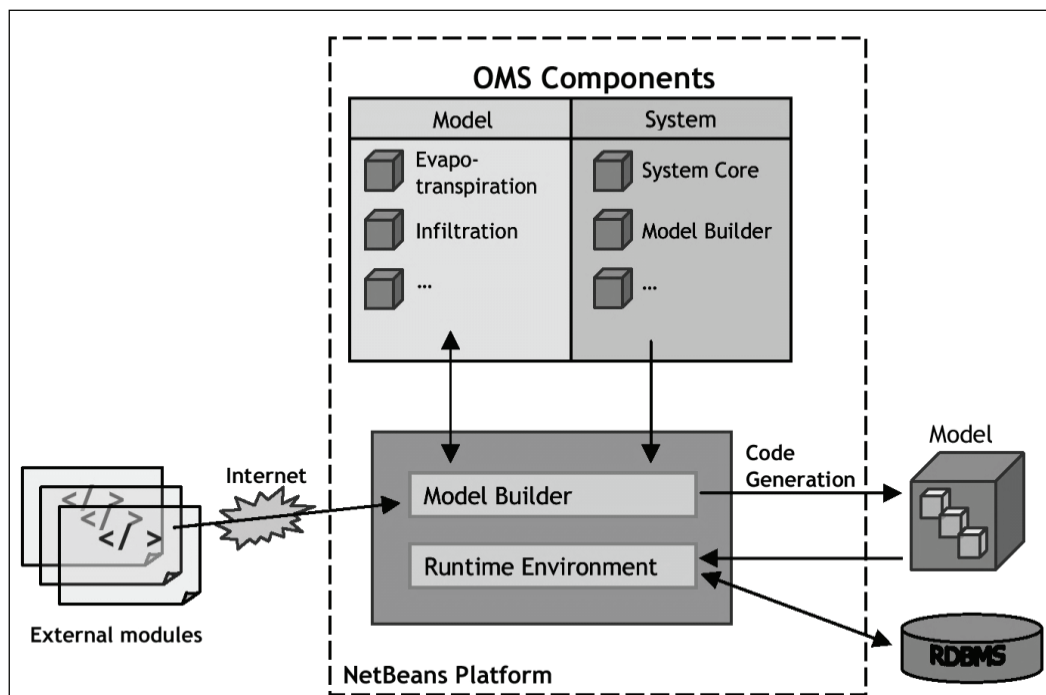


Figure 3.6 Conceptual layout of OMS (Kralisch *et al.*, 2005)

As explained previously OMS model components are the building blocks from which custom models are created. Kralisch *et al.* (2005) explain that developers must implement certain OMS specific properties in each model component. Each model component must implement four standard methods, *register*, *init*, *run* and *cleanup*. The *register* method should contain functionality that is required to be run once during model initialization. The *init* method should contain functionality to be run once at the first time the component is executed, such as setting initial values for parameters and variables. The *run* method should contain the algorithm calculations for the process represented by the component and is called every time the component is invoked. The *cleanup* method should contain functionality to be run during model finalisation, for example, to free any resources used by the component. In addition, for each input and output parameter or variable used by a component, the read and write access, which is supervised by each individual model component, must be specified. The model components are implemented as Java classes, but may access functionality from libraries outside the Java runtime environment using the Java Native Interface (JNI), enabling existing software developments written in other programming languages to be easily incorporated. OMS also makes provision for specialised compound components, within which other components may be executed, enabling control structures such as conditions and iteration. OMS provides two special predefined compound components *TimeCompoundComponent* and *SpatialCompoundComponent* which provide the temporal and spatial contexts required by most models. *TimeCompoundComponent* includes an attribute that represents a user specified time interval type and step size. *SpatialCompoundComponent* represents discrete points in space by means of a list of spatial entities such as HRUs or GIS raster cells. Examples demonstrating the use of *TimeCompoundComponent* and *SpatialCompoundComponent* to execute model components iteratively through time and space, including hierarchies of these components is shown in Figure 3.7.

Kralisch *et al.* (2005) explain that OMS provides the *OMSEntity* data type to represent spatial entities in model components, where *OMSEntity* is an abstract container for arbitrary attributes of spatial entities. *OMSEntity* objects store attribute data in tables that map attribute names to corresponding attribute values. The attribute set within an *OMSEntity* object may be easily expanded to accommodate additional data obtained from other model components or external data sources. Each spatial entity is represented by a unique ID in an *OMSEntity* object, and 'get' and 'set' methods are provided to access the attributes for a specified entity ID. The use of *OMSEntity* objects to represent spatial attribute data is illustrated in Figure 3.8.

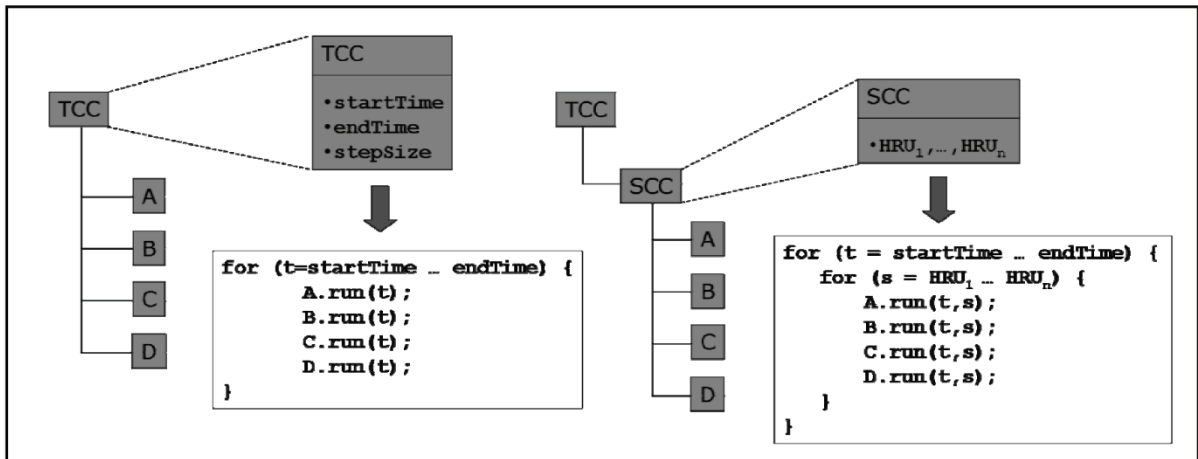


Figure 3.7 Application of compound components to control execution of model components in time and space (Kralisch *et al.*, 2005)

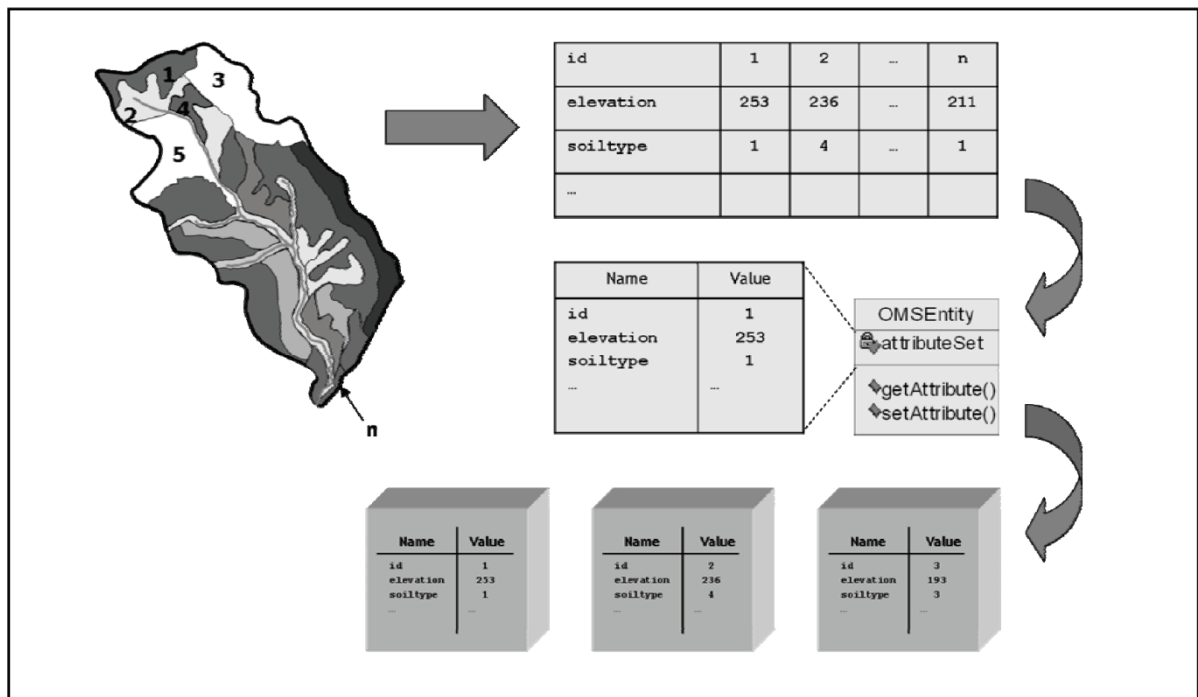


Figure 3.8 Representing spatial entities using OMSEntity objects (Kralisch *et al.*, 2005)

David *et al.* (2010) explain that OMS Version 1 was a heavyweight framework in that it offered a traditional library of classes, in the form of an Application Programming Interface (API), to be subclassed or instantiated directly by components or models, and a limited set of data types that could be used to exchanged data values between linked components. They state that this was a simple approach but that it was not suitable for legacy code integration and restricted the use and sharing of custom data types. David *et al.* (2010) explain further that in OMS Version 2 component design was simplified by requiring components to use interfaces instead of classes as is the case in many other frameworks, resulting in a better,



more robust and more lightweight framework, though supported data types were still fixed. Following from an in-depth analysis of successful framework designs and software engineering principles. David *et al.* (2010) report that in OMS Version 3 they have moved away from the traditional API-based approach to a lightweight, non-invasive approach, with a small API, by using programming language annotations to provide metadata at relevant points in the model code which is then read by the framework. Their objectives for Version 3 were to enable easier integration of model code, through the use of programming language annotations with the flexibility to integrate legacy models. These code annotations enable the specification of connections between components, data transformations, conversion of units of measure and automated model documentation. They conclude that the architecture of Version 3 provides an environmental modelling framework that is scalable, easier to use and more transparent. An interesting development is that Gijsbers *et al.* (2010) , citing David *et al.* (2009), state that OpenMI has been implemented on top of OMS. David *et al.* (2009) state that the use of code annotations provide a means of enabling interoperability between models and modelling frameworks.

### 3.1.2.2 Application

Kralisch *et al.* (2005) report that J2000, a distributed conceptual hydrological model, has been implemented in the version of OMS being used at Friedrich Schiller University in Jena, Germany. They report that the implementation was successful though performance in comparison to the original implementation of J2000 was slower, and concluded that this was related to the flexibility gained by the use of dynamic attribute sets for model entities inside the OMS. Ahuja *et al.* (2005) and David *et al.* (2004) indicate that the Root Zone Water Quality Model (RZWQM) and Precipitation-Runoff Modelling System (PRMS) have been implemented in OMS. Jagers (2010) mention that an OMS implementation of the Soil and Water Assessment Tool (SWAT) was being created. David *et al.* (2010) report some recent implementations of OMS that are being tested and applied as follows:

- Implementation of the J2K-S model for distributed simulation of water balance and Nitrogen dynamics in large watersheds using OMS version 3 is described in Ascough *et al.* (2010).
- A PRMS-based model family with associated methods and tools for water supply forecasting by the NRCS National Water and Climate Centre.
- The Conservation Delivery Streamlining Initiative (CDSI) model base to deliver science deployed as services available to USDA-NRCS field consultants.

The core components of the Horton Machine, a hydro-geomorphological toolbox and the NewAge semi-distributed hydrological model.

### 3.1.2.3 Comments

Ahuja *et al.* (2005) believe that OMS will leverage the sizeable investments of developer time and money, made in existing complex natural resource systems, to facilitate an interdisciplinary effort extract the best scientific routines from these models and enable integration and interoperability of new and existing modules and data resources, while reducing duplicate functionality across natural resource models. Ahuja *et al.* (2005) conclude that the component-oriented and modular approach of the OMS and its library of implemented modules and models will enable the collaborative, integrative and more efficient model development approach that is required to solve global challenges related to natural resource systems. David *et al.* (2010) state that the structure of OMS Version 3, being a lightweight and non-invasive modelling framework will enable quick development of new models and easier integration of legacy models for use on multiple platforms.

OMS has a dual purpose, part modelling framework and part model linking mechanism. The model linking architecture appears to be sound and is could be used to link both whole models and process modules. Although not much literature has been published by the developers, OMS appears to be fairly well known, judging from the times it is referenced as an example of a model linking framework. OMS does not appear to be well documented, but the source code is available in the public domain. Development and implementation seems to be largely restricted to the USGS and USDA-ARS modelling groups. The primary strength of OMS is its use of metadata tags and annotation which enable a lightweight model coupling mechanism.

### 3.1.3 Jena Adaptable Modelling System (JAMS)

#### 3.1.3.1 Overview

The Jena Adaptable Modelling System (JAMS) is described as an environmental modelling framework for component based model development and application, with a focus on water resources management (Kralisch and Krause, 2006; Kralisch *et al.*, 2007; Fischer *et al.*, 2009). It is stated by Kralisch and Krause (2006) that JAMS is based on OMS, but that the capabilities of OMS have been enhanced based on special requests from model developers. They go on to state that the focus of JAMS is on providing flexibility for the development of new model components and less on easy integration of existing models. Kralisch and Krause (2006) state that the aim of JAMS is to represent complex simulation models as sets of well-defined model components, whose functionality can include single

processes, complex sub-models, and data input and output. The architecture of JAMS and how it works is described in Kralisch *et al.* (2007) and Kralisch and Krause (2006) though it is not clear which parts are specific to JAMS and which parts are specific to OMS. Fischer *et al.* (2009) describe the calibration of environmental models using JAMS.

As with OMS, JAMS is implemented in Java. Kralisch and Krause (2006) explain further that a JAMS model is defined by a XML-based model description document listing the model components from the component library that are used, how the listed model components are assembled and what data must be exchanged between components. The organisation of the JAMS framework is shown in Figure 3.9. The JAMS core consists of the core library and the runtime system. The core library contains data types and core functionality such as I/O mechanisms and unit conversion. The runtime system is responsible for model configuration and execution including communication between JAMS components.

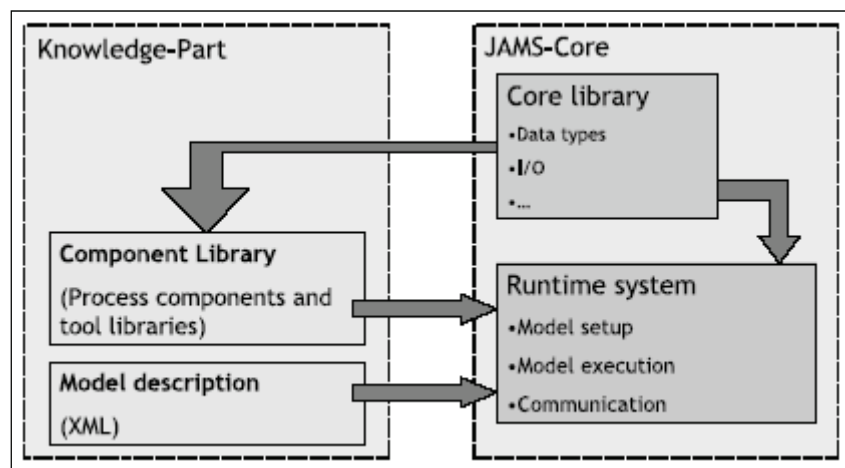


Figure 3.9 Organisation of the JAMS framework Kralisch and Krause (2006)

JAMS components must contain a set of data attributes through which data is exchanged with other components, and three methods, *init* which is executed once when the model starts, *run* which is executed repeatedly for different points in time and space, and *cleanup* which is executed once at the end of model execution. JAMS provides specialised components called 'context components' as a base for components in which the *run* method needs to be executed repeatedly for different points in time or space, where context components act as containers for other components. Model Context components represent a JAMS model and contain an ordered list of other components whose run methods are executed once during a model run. Temporal Context components provide the time related control during model execution. Temporal Context components contain an ordered list of other components to be executed, start and end date of simulation, and time step size.

Spatial Context components represent the spatial domain and contain an ordered list of spatial model entities, and an ordered list of components to be executed for each spatial entity.

A component may exchange data with another component or a spatial entity provided within a spatial context if the input and output data variables for the component have been declared and at configuration time the sources of input data have been specified. Component input and output data are specified using Java annotations that provide the following: the access type (input, output or both), the update type (at initialisation or run stages), units of measure, and minimum and maximum values.

### **3.1.3.2 Application**

Kralisch *et al.* (2007) state that a collection of JAMS modelling components have been developed covering various aspects of IWRM including hydrological and nutrient modelling and parameter optimization. They also mention several hydrological models that have been implemented in JAMS, including the Thornthwaite water balance model, HYMOD, J2000, parts of WASIM-ETH and PRMS, and the SNOW17 processes. Kralisch and Krause (2006) present an example application where the FAO reference evapotranspiration (refET) model was implemented as a model in JAMS. Kralisch *et al.* (2009) describe how JAMS and the River Basin Information System (RBIS), which is a web-based environmental information management system, were coupled to enable data sharing between the two systems.

### **3.1.3.3 Comments**

Kralisch *et al.* (2007) state that JAMS can be used to easily create custom models tailored to simulate a wide range of specific environmental problems while reusing existing solutions. From literature it appears that JAMS has been developed and applied within Department of Geoinformatics, Hydrology and Modelling at Friedrich-Schiller-University, Jena, Germany. As mentioned in Section 3.1.2, development of OMS started in 1996 at Friedrich Schiller University (FSU) in Jena, Germany and since 2000 development continued at the USDA-ARS Great Plains Systems Research Unit (Fort Collins, CO) and the USGS (Denver, CO), possibly jointly with FSU (Ahuja *et al.*, 2005; Kralisch *et al.*, 2005). It is not clear in the literature which version of OMS JAMS is based on, what the differences are between JAMS and OMS, and whether there are two divergent developments of OMS in Germany and the USA. These questions were clarified via personal communication with (Kralisch, 2011) at FSU who confirmed that JAMS and the current USDA-ARS and USGS version of OMS are

divergent versions of the original version of OMS, though the core platforms are the same and the JAMS and OMS developments groups do collaborate on further development of both frameworks. Kralisch (2011) reports that components can be exchanged between JAMS and OMS with minor changes to the components. Kralisch (2011) explains the differences between JAMS and OMS as follows:

- JAMS only supports Java components, while OMS also supports legacy code in other programming languages;
- JAMS includes additional tools to support the creation, application and analysis of models, for example, a graphical model builder, a model calibration assistant, a toolbox for parameter sensitivity and uncertainty analysis, and a toolbox for analysing simulation results; and,
- JAMS provides a large number of modelling components including hydrological modelling, nutrient modelling and erosion modelling.

Many of the comments that were made with respect to OMS apply to JAMS. It appears that JAMS is used by FSU more in a research capacity, and its development and implementation seems to be restricted to the environmental modelling group at Friedrich Schiller University.

### **3.1.4 The Invisible Modelling Environment (TIME)**

#### **3.1.4.1 Overview**

The Invisible Modelling Environment (TIME) is described as a model development framework for the creation, testing and integration of new model components and the development, application and deployment of environmental model applications (Rahman *et al.*, 2003; Rahman *et al.*, 2005; Murray *et al.*, 2007). The objective of the Catchment Modelling Toolkit is to provide a cohesive suite of environmental modelling applications, and this is achieved through the TIME framework which enables models to be developed and integrated quickly and consistently (Rahman *et al.*, 2003). TIME consists of a collection of .NET classes, libraries and visualisation components for use in the development of model components and applications. Rahman *et al.* (2003) state that TIME is different to most other frameworks, mostly in its use of metadata to describe and manage models, and that it gives model developers the flexibility to select the components of TIME that are relevant to a specific project. Argent and Rizzoli (2004) state that the primary features of TIME are that it has a thin architecture and a strong facility to utilise model metadata enabling the TIME system to be automated in many ways. TIME enables deployment of models as graphical applications, command line applications and active webpages (Murray *et al.*, 2004).

TIME was developed in 2001 within the Catchment Modelling Toolkit project by the Cooperative Research Centre for Catchment Hydrology (CRCCH) in Australia, and funded by the Commonwealth (Rahman *et al.*, 2003). The CRCCH includes several institutions including universities, research institutes, water services providers, catchment management organisations and government departments. The Catchment Modelling Toolkit is a system of environmental modelling software which integrates a new generation of catchment models and modelling support tools (Marston *et al.*, 2002). The aim of the Catchment Modelling Toolkit is to provide land and water managers, researchers and educators with an integrated collection of software tools and components to simulate catchment response to management and climate variability, at a range of scales and using a variety of approaches (Marston *et al.*, 2002). The requirements for TIME and the Catchment Modelling Toolkit in general were based on three surveys conducted in 2001 among catchment managers, model users, model developers and model coders in Australia to gather information about which models were being used, the types of model applications, and the design and development behind the models. The results of these surveys are documented in Marston *et al.* (2002). From the eWater Toolkit website (<http://www.toolkit.net.au>) it appears that prospective users of TIME would be required to first attend a training course and may then apply for access to the source code and is subject to a licence agreement.

TIME was developed on the Microsoft .Net platform, mostly using the C# programming language and supports model development in a variety of .Net programming languages including C#, VB.Net, Fortran95.Net, C++, Delphi.Net and Visual J# (Murray *et al.*, 2007). The models and tools forming the Catchment Modelling Toolkit are based on TIME and the approach seems to have been to restructure legacy models into a .Net programming language with a direct implementation of TIME.

Rahman *et al.* (2003) explain that the architecture of TIME is divided conceptually into five layers, as shown in Figure 3.10. Each layer consists of a family of classes, with the classes in the upper layers using services provided by classes in lower layers. Developers create models in the Model layer using classes in the Kernel and Data layers. The Tools and Visualisation and User Interface layers contain classes that interact with models and provide most of the framework functionality, such as user interface generation and model linking. Rahman *et al.* (2004) explain that with a thin Kernel layer, most framework functionality including user interface generation and model linking, is implemented in the Tools layer, enabling models, which mostly use the Kernel and Data layers, to remain independent of these tools. These layers are described further by Rahman *et al.* (2003) and Murray *et al.* (2007).

The Kernel layer contains the core classes of the framework, as shown in Figure 3.11, which support the other layers. *Model* is the abstract parent class for all TIME models, and it contains the abstract *runTimeStep* method which must be implemented by all child classes.

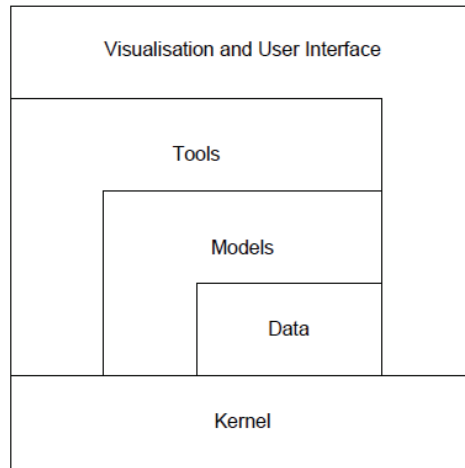


Figure 3.10 Architectural layers of the TIME modelling framework (Rahman *et al.*, 2003)

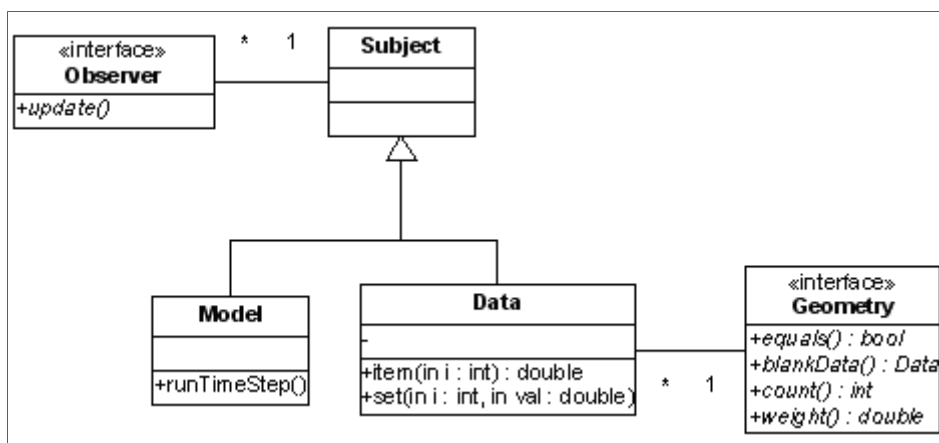


Figure 3.11 Core classes of the Kernel layer (Rahman *et al.*, 2003)

*Data* is the abstract parent class for all classes representing data types in TIME. The *Data* class includes the *item* and *setItem* methods for generic one dimensional access to data values for all data types. These two access methods must be implemented in all child classes of *Data* to provide a common interface. Child classes of *Data* may declare access methods more specific to the particular data type, enabling more convenient access to data values. The spatial or temporal context of a data object is specified by its association with a class implementing the *Geometry* interface. A group of data objects sharing a common geometry as specified through the *Geometry* interface are considered compatible for certain operations such as mathematical addition and subtraction, and custom operations such as regression analysis. This separation between *Data* objects storing data values and

*Geometry* objects enables efficient representation of spatial data such as a layer of polygon features linked to an attribute table. The shape and location of the polygon features is stored in a shared *Geometry* object, and the data values for each field in the attribute table are stored in a different *Data* object. Hence, most data types are represented by two classes, a class implementing the *Geometry* interface which stores the spatial or temporal context, and a *Data* class storing the data values specific to the data type.

The *Model* and *Data* classes are child classes of the *Subject* support class. An instance of the *Subject* class may be associated with one or more classes implementing the associated *Observer* interface. This allows *Observer* type objects to subscribe to *Model* and *Data* objects to receive notification of changes through events.

The Kernel layer includes definitions of the various custom metadata tags used to classify and document properties of TIME model components and models, and the scope of model parameters; these are listed in Table 3.2. Rahman *et al.* (2004) explain that the use of TIME custom metadata tags and the capacity for introspection that they offer, enables the code in models to remain independent of TIME's support operations, such as data and model management, model linking, IO and data visualisation, resulting in better model stability (Rahman *et al.*, 2004). TIME makes use of the language independent introspection mechanism in .Net for discovering components and their properties at runtime, including class structure, class member fields and methods as well as custom metadata tags.

The Kernel layer also includes a set of classes that may be used to represent common data units. As shown in Figure 3.12 units may be represented as a *Simple Unit* such as length, mass or time, or as a *Compound Unit* that combines two or more *Units* using multiplication or division.



Table 3.2 TIME custom metadata tags defined in the Kernel layer (Rahman *et al.*, 2003; Murray *et al.*, 2007)

Category	Tag	Description	Applied To
Classification	Input	Variable is a non-static model input	Fields
Classification	Output	Variable is a model output	Fields
Classification	Parameter	Variable is a static one off input to the model	Fields
Classification	State	Variable is an internal state	Fields
Constraints	Minimum	Minimum allowable value of a variable	Fields
Constraints	Maximum	Maximum allowable value of a variable	Fields
Display	Ignore	Exclude component or field from generic tools	Classes, Fields
Documentation	Aka	Alternative name for a variable	Fields
Documentation	Author	Author of the code	Classes
Documentation	LastModified	Last time code was modified	Classes
Documentation	Note	Describe an auxiliary aspect of a variable	Field
Documentation	Status	Release status of a variable or model	Classes, Fields
Documentation	Summary	General description of fields or classes	Classes, Fields
Documentation	URL	URL to further documentation	Classes, Fields
Other	Minimise	Whether to maximise or minimise an objective function	Fields
Other	UserOption	Flags a field as a default the user can change and maintain across sessions	Field
Other	WorksWith	Class works with a particular type	Classes
Parameter attribute	CalculationUnits	Specifies the units a class uses internally for a variable	Fields
Parameter attribute	DecimalPlaces	Number of significant decimal places for a variable	Fields
Parameter attribute	Default	Value to be applied to a property when model initialised or reset	Fields
Parameter attribute	ExpectedUnits	Units that make sense for a variable	Fields
Parameter attribute	Fixed	Indicated variable should not be modified for calibration	Fields

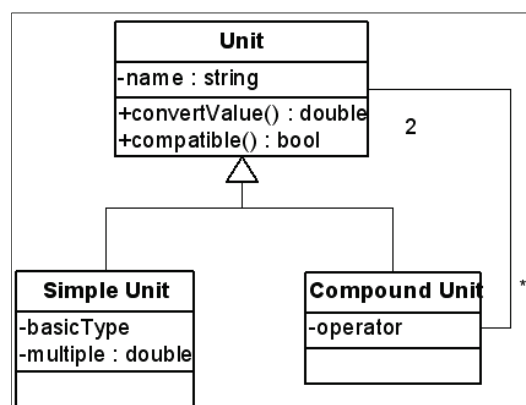


Figure 3.12 Representation of units in TIME (Rahman *et al.*, 2003)

The Data layer contains classes representing a range of data types, and some of the more specialised data types are listed in Table 3.3. Murray *et al.* (2007) describe data cubes representing a time continuum of two dimensional data, such as a time-series of rasters or a raster of time-series. The Data layer also contains classes that perform data input and output operations on a selection of text, database and spatial data formats.

Table 3.3 Some specialised TIME data types (Rahman *et al.*, 2003)

Data Type	Description
Raster	Two dimensional regular grid of data, located within a geospatial context.
Time-Series	Temporal arrangement of data on one of several fixed time steps.
Node Link Network	Abstract representation of physical networks, such as river systems.
Sites	Collections of points in space.
Poly Lines	Linked collection of multi segment lines.
Polygons	Collection of closed polygonal regions.
Cross Sections	Cross sections surveyed, or generated river cross sections.
Arrayed Data	Ordered list of values with no spatial or temporal context.

The Models layer contains the models and model components included in the TIME framework, and is the layer in which most developers will work creating their own models. Models typically only reference classes in the Kernel and Data layers and contain only core scientific algorithms. Models are then included in model applications which include user interfaces and data handling. All TIME models are implemented as a class which inherits from the *Model* class and are written in one of the .Net languages. Each model must implement the abstract *runTimeStep* method in the *Model* class, this method is called for each iteration of the model. Each model class will contain fields specifying input, output, parameter and state variables that are each documented using TIME custom metadata tags. A simple model written in C# is shown in Figure 3.13 and illustrates the use of TIME custom metadata tags.

The Tools layer includes various classes that may be used for generic processing of data and models, data statistics and parameter optimisation. These classes may be used by developers when writing and testing models and model applications. The Tools layer contains classes that make use of the TIME custom metadata tags in models to provide a set of model processing tools. These tools include automatic generation of graphical user interfaces and command line interfaces for models, linking of models, and several parameter optimisation tools. Other tools provide support for attaching data to model input and output variables. Automatically generated user interfaces are designed to enable model developers to quickly and easily test models. Modelling applications generally include custom designed user interfaces.

```

using System;
using TIME.Core;

public class ToyModel : Model {
    [Input,Minimum(0.0)] double rainfall;
    [Input,Minimum(0.0)] double actualET;
    [State] double netRainfall;
    [Parameter,Minimum(0.0),Maximum(1.0)] double coefficient;
    [Output] double runoff;

    public override void runTimeStep( ) {
        netRainfall = Math.Min( 0.0, rainfall-actualET );
        runoff = coefficient * netRainfall;
    }
}

```

Figure 3.13 Example of a simple model including TIME custom metadata tags (after Rahman *et al.*, 2003)

The Visualisation and User Interface layer contains a collection of classes that provides users with a visual interface to the data, models and tools in the other layers. A group of classes known as the Visualisation Framelet is shown in Figure 3.14, which are the parent classes for all classes created for the visualisation of data. Subclasses of Layer are used to draw a particular representation of a data type onto a *Canvas* and may be used to draw both graphs and spatial maps. A *Canvas* may contain several superimposed layers. A *Canvas* manages the drawing of individual layers and translates data coordinates for a layer to screen coordinates. *ViewDecorators* are used to draw axes, titles, labels and legends onto a canvas. *ViewControl* objects are used to display a *View* object in a graphical user interface. *View* objects may also be used for printing, bitmap generation and in web-based mapping tools. The Visualisation and User Interface layer contains numerous classes for graphical display based on the classes in the Visualisation Framelet. These classes include standard *Layer* classes for each of the main data types except arrayed data, and graph layers displaying scatter plots, cumulative frequency graphs, flow duration curves and probability density plots. TIME also provides support for the use and visualisation of rasters.

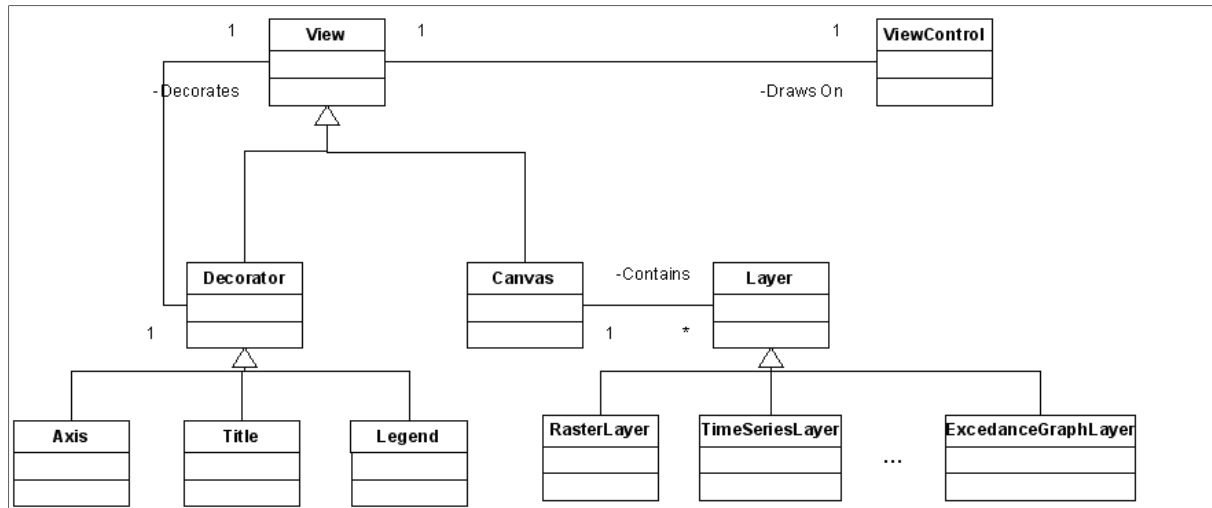


Figure 3.14 TIME Visualisation Framelet classes (Rahman *et al.*, 2003)

### 3.1.4.2 Application

TIME is part of the Catchment Modelling Toolkit and is the foundation on which the models, model applications and other modelling tools included in the Catchment Modelling Toolkit are built. Argent *et al.* (2009) mention that TIME is used as the base for over 30 integrated catchment process modelling tools, most of which form part of the Catchment Modelling Toolkit. A list of modelling tools in the Catchment Modelling Toolkit reproduced from the eWater Toolkit website (<http://www.toolkit.net.au>) is shown in Table 3.4. One of the most noteworthy implementations of TIME is in the Source Catchments modelling framework (formerly called WaterCAST and E2) described in Argent *et al.* (2009) and Cook *et al.* (2009).

### 3.1.4.3 Comments

Rahman *et al.* (2005) state that TIME has a growing user base and that its evolution and maturation were mostly stakeholder driven with much of the functionality being dictated by the requirements of the modelling tools forming the Catchment Modelling Toolkit. Rahman *et al.* (2004) make the point that the use of markup, and the introspection of this markup, simplifies the development of modelling tools, including model linking engines and user interface generators, while enabling greater flexibility by keeping models and model components decoupled from details of a framework and its tools.

From the literature it seems that TIME almost has a dual purpose, part modelling framework and part model linking mechanism, although the model linking mechanism is not explained. TIME appears to have many similarities to OMS, particularly in its use of metadata and

introspection, except that TIME is .Net based while OMS is Java based. The .Net platform gives developers using TIME the advantage of being able to use several object-oriented .Net programming languages. As with OMS, TIME seems to be suitable for use both in legacy models, and in the modular approach of small process modules or components that may be linked together by users to create custom models. TIME is well documented and is maintained and supported by the eWater Cooperative Research Centre. TIME seems to have been mainly developed and implemented within the water resource management fraternity in Australia, and has been implemented in a wide range of models developed in Australia. The source code is available to the public subject to training and licencing.

Table 3.4 Modelling tools within the eWater Catchment Management Toolkit (after <http://www.toolkit.net.au>)

<b>Tool</b>	<b>Description</b>
Aquacycle	Aquacycle is a total urban water balance model gaming tool.
BC2C	BC2C (Biophysical Capacity to Change) is a tool for estimating catchment scale water and salt export quantities, following changes in landuse in upland catchments.
CatchmentSIM	CatchmentSIM is a 3D-GIS topographic parameterisation and hydrologic analysis model.
CHUTE	CHUTE is a spreadsheet program for the design and analysis of rock chutes.
CLASS-CGM	CLASS-CGM (Crop Growth Model) can be used to simulate growth of main C3 field crop types.
CLASS-PGM	CLASS-PGM (Pasture Growth Model) can be used to simulate growth of composite pasture types.
CLASS-SA	CLASS-SA (Spatial Analyst) is a spatial modelling tool.
CLAS-U3M-1D	CLASS-U3M-1D (Unsaturated Moisture Movement Model) can be used for estimating recharge, plant water use and soil evaporation across the soil profile at daily time steps using the Richards' equation.
CMSS	CMSS (Catchment Management Support System) predicts average annual loads of pollutants (usually Total Phosphorus and Total Nitrogen) at the subcatchment level, according to different land use types.
Concept	Concept is a conceptual diagram drawing package that can be used to communicate dynamic relationships between multiple elements.
E2	E2 is a whole-of-catchment model building and running application that can simulate the effects of scenarios (e.g. land use or climatic change) on the flow and load of constituents (e.g. sediments, nutrients, salt) at defined points in a river network over time.
Eco-Modeller	Eco Modeller is a tool for building, storing and running quantitative models of ecological responses to physical and biological factors, for use in comparing the merits of alternative natural resource management scenarios.
eFlow-Predictor	eFlow uses environmental flow objectives to generate an altered flow regime and determine how much additional water would be required to achieve the new flow regime.

Table 3.4 (continued) Modelling tools within the eWater Catchment Management Toolkit  
(after <http://www.toolkit.net.au>)

<b>Tool</b>	<b>Description</b>
FCFC	FCFC (Forest Cover Flow Change model) is used to adjust daily time series observed or simulated flow records for significant changes in forest cover.
IHACRES	IHACRES (Identification of unit Hydrographs And Component flows from Rainfall, Evaporation and Streamflow data) is a catchment-scale, rainfall-streamflow, modelling methodology that characterises the dynamic relationship between rainfall and streamflow, using rainfall and temperature (or potential evaporation) data, and predicts streamflow.
LIZA	LIZA (Landcover for the use Zone of Australia) is a collection of maps and GIS data that provide landcover type for 1990 and 1995 for the intensive use zone of Australia.
MCAT	MCAT (Multi Criteria Analysis Tool) is an investment decision support tool that optimises environmental expenditure using multi-criteria analysis and combinatorial optimisation techniques.
MELS	MELS (Minimum Energy Loss (MEL) Structures) is a hydraulic design and analysis suite that enables designers to quickly trial several alternative MEL culvert designs, checking for basic structure dimensions and performance under adverse conditions such as high or low flow and sedimentation issues.
MUSIC	Model for urban stormwater improvement conceptualisation
NSFM	NSFM (Non-parametric Seasonal Forecasting Model) forecasts continuous exceedance probabilities of streamflow (or any other hydroclimate variable).
RAP	RAP (River Analysis Package) is a collection of 3 tools: Hydraulic Analysis – examines the hydraulic characteristics of river channels to determine the optimal discharge for a river reach based on specified rules. Time Series Analysis – calculates summary statistics of time series data, including hydrological metrics. Time Series Manager – manipulates and manages time series data.
RIPRAP	RIPRAP is a spreadsheet program for the design of rock lining (rip-rap) bank protection. It provides a range of rock sizes to be used depending on bank angle and depth chosen.
RRL	RRL (Rainfall-Runoff Library) simulates catchment runoff by using daily rainfall and evapotranspiration data.
SCL	SCL (Stochastic Climate Library) is a source of models for generating climate data, including rainfall, evaporation or temperature, at multiple timescales, across single or multiple sites.
SedNet	SedNet identifies sources and sinks of sediment and nutrients in river networks and predicts spatial patterns of erosion and sediment load.
SHPA	SHPA (Soil Hydrological Properties of Australia) is a collection of maps and GIS data that provide estimates of soil hydrologic properties across Australia based on the Atlas of Australian Soils and interpretations by Neil McKenzie.
Source-Catchments	Water quality and quantity modelling framework that supports decision making and a whole-of-catchment modelling approach.
TREND	TREND facilitates statistical testing for trend, change and randomness in hydrological and other time series data, utilising 12 different statistical tests.
Urban Developer	Tool for urban water management.
WRAM	WRAM (Water Re-Allocation Model) simulates water allocation and trading between irrigation areas.

### 3.1.5 LIQUID<sup>®</sup> Modelling Framework

#### 3.1.5.1 Overview

LIQUID<sup>®</sup> is described as a modelling framework for modelling hydrological processes (Branger *et al.*, 2010a; Branger *et al.*, 2010b). The LIQUID<sup>®</sup> framework was developed for the purpose of providing easy integration of hydrological processes while maintaining their individual spatial and temporal scales, enabling integrated models composed of reusable modules to be built and run. The framework includes templates for creating new modules, module coupling mechanisms and connections to work with input and output data in GIS and databases. LIQUID<sup>®</sup> is claimed to be able to represent complex interactions between modules, including feedbacks, different time steps and irregular geometries.

The LIQUID<sup>®</sup> framework is proprietary software and has been under development since 2005 by Hydrowide, but can be made available for research purposes by means of a partnership contract, and modules are subject to licences imposed by the individual module developers (Branger *et al.*, 2010a). There does not appear to be much literature related to LIQUID<sup>®</sup> and the documentation on the Hydrowide website (<http://www.hydrowide.com/liquid/current/>) was incomplete. Development and use of the LIQUID<sup>®</sup> framework seems to be confined to researchers from Hydrowide, Cemagref and Grenoble University.

Branger *et al.* (2010a) explain that LIQUID<sup>®</sup> consists of two main sections, the framework, consisting of core components and some general utilities, and the platform, consisting of modelling components and their documentation, as illustrated in Figure 3.15. LIQUID<sup>®</sup> manages a library of modelling modules representing hydrological processes and provides a build system enabling custom hydrological models to be built using selected modules in the library (Branger *et al.*, 2010b). Three categories of user are envisaged for LIQUID<sup>®</sup>, module developers, model developers and model users. One of the most important core components of the LIQUID<sup>®</sup> framework is the Scheduler which manages the interactions between modules within a model at runtime and controls the simulation time steps. Another important core component of the LIQUID<sup>®</sup> framework is the model build system which enables operating system and compiler independent code compilation, and creation of executable files using the Build tool provided by Boost (<http://www.boost.org>). A set of third party libraries for numerical analysis, geometry calculations and connecting to databases are included in the LIQUID<sup>®</sup> framework for use by module developers. The LIQUID<sup>®</sup> framework also includes a test framework that enables module developers to build and run tests for

modules, and a system to automatically generate module code documentation is also available. LIQUID<sup>®</sup> is programmed using the C++ programming language, including template and meta-template programming, and makes use of the standards ANSI C++, OpenGIS, Open DataBase Connectivity (ODBC) and DocBook standards. (Branger *et al.*, (Branger *et al.*, 2010a)a). Most of the modules currently in the library were implemented in C++ but it is possible to implement modules in other programming languages such as FORTRAN.

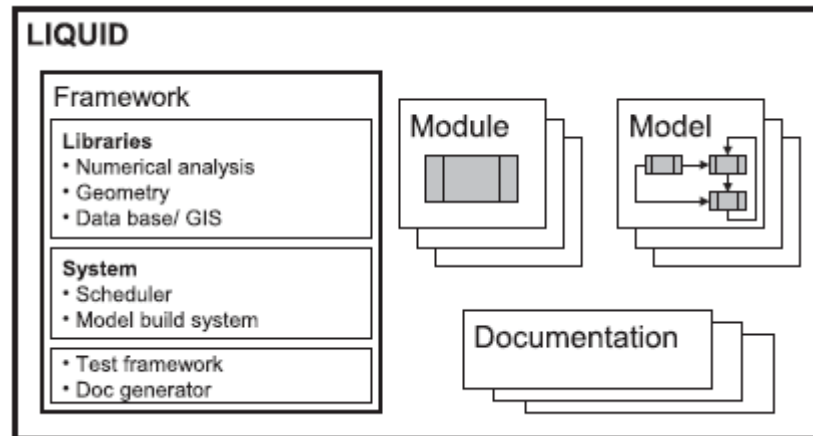


Figure 3.15 The main sections of LIQUID<sup>®</sup> (Branger *et al.*, 2010a)

Branger *et al.* (2010a) and Branger *et al.* (2010b) explain that each module is autonomous and represents one or more hydrological processes occurring on one or more spatial entities at a specific temporal and spatial scale. Each module includes five main components: a data scheme, a pre-processor, a solver, test cases and documentation as shown in Figure 3.16. Each module contains its own spatial data scheme which describes the time-independent data required by the module including parameters, initial values and the spatial entity types. The data used in the modules is accessed from a PostgreSQL/PostGIS database through an ODBC connection. Based on the data scheme for each module empty tables are created in the database and the model user populates these tables with appropriate data which is then read by the pre-processor and used to initialise a solver with parameters and initial values for each spatial entity being modelled. The hydrological simulation takes place within these solvers. Each solver is responsible for managing its own time steps and these time steps are re-evaluated each time the solver is executed. Each solver can receive input through its slots and provide output through its signals. There is one signal for each output variable and output values are delivered each time the solver executes. A slot is a method that is called when a new input value is received by the solver, and is the means through which a solver responds when new input value become available. In other words LIQUID<sup>®</sup> uses a push-driven mechanism to govern the progression of the



hydrological processes being modelled. The modules making up a model communicate through their slots and signals, and through these, both simple one-way coupling and couplings representing feedbacks can be established. Modules are created using LIQUID<sup>®</sup> templates in which the module developer first defines the slots, signals and data scheme, then writes code for the signals, slots, pre-processor and the solver, and finally provides metadata describing each slot and signal. One important constraint is that solvers must be time step independent, so that they can be run with a variable time step. SI base units are used by convention to prevent having to convert between units of measure, but it is ultimately up to the model developer to ensure units are consistent between associated signals and slots. The module library within LIQUID<sup>®</sup> contains Input modules for reading time series input data and Output modules for writing output data to ASCII files. The architecture of the Input and Output modules is the same as for the hydrological process modules. When creating a model one Input module is required for each time series input variable and can handle fixed interval or variable interval time series. Each time an Input module reads a new value from the database it sends a signal to the appropriate slots of connected process modules. The Output module can perform operations such as time series aggregation.

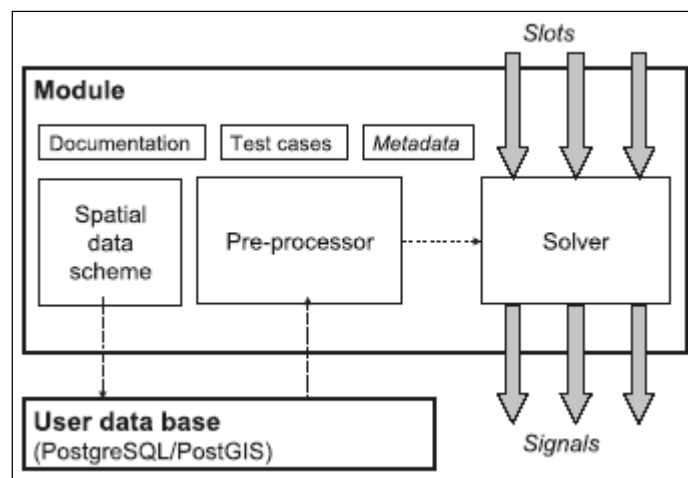


Figure 3.16 Architecture of a LIQUID<sup>®</sup> module (Branger *et al.*, 2010a)

Branger *et al.* (2010a) and Branger *et al.* (2010b) explain that a model is created by selecting suitable modules and connecting their slots and signals. This model configuration information is stored in a XML file which is read by the build system. The build system uses this configuration system to compile the specified modules and create an executable file for the model. During a model run the Scheduler manages the progression of the simulation. At each time step during a simulation each module solver estimates what its next time step will be and schedules the time of its next execution with the scheduler. When the next

scheduled execution date is reached for a particular module, the Scheduler calls the module and it executes. Based on signals received by a solver the slots can decide to reschedule the next execution of the module. This design enables simulations to progress at variable time steps. For most LIQUID<sup>®</sup> models it is envisaged that a study catchment will be divided into hydrological response units (HRUs), depending on the objectives of the study and data availability, and during pre-processing of a model run a separate solver instance would be created for each HRU on which a particular module is applied.

### 3.1.5.2 Application

Branger *et al.* (2010a) describe several applications of LIQUID<sup>®</sup> to create models as follows:

- PESTDRAIN simulates pesticide transport in tile-drained agricultural fields at an event scale. It is made of three modules, SIRDA which simulates water flow in the saturated zone, SIRUP which simulates surface runoff and water flow in the unsaturated zone, and SILASOL which simulates pesticide transport in the saturated and unsaturated zones. The model has a variable time step ranging from 3 minutes or less during rainfall events to a week during dry periods, providing a compromise between accuracy and computation time.
- ELIXIR-D2D was developed to estimate the effect of temporary pressurization of drainage pipes on the discharge agricultural field drainage systems during heavy rainfall events. ELIXIR is a 1D Saint-Venant module that computes flow and hydraulic head in pipe and channel networks, and D2D is a 2D shallow water table module based on Boussinesq approximations used to compute water table elevation and discharge into buried drains.
- BVFT is used to assess the influence of landscape management on water fluctuations in small agricultural catchments. It uses SIDRA and SIRUP modules, the FRER1D module for undrained agricultural zones, the HEDGE module for modelling the influence of hedgerows and the RIVER1D module for simple flow routing in ditch and river networks.
- CVN was developed for the analysis of hydrological responses during flash flood events. The FRER1D is used to calculate infiltration and is linked to the PEF module which calculates runoff based on surface ponding, and the RIVER1D module simulates water flow in the flow network.
- BALANCE model which models sensitivity of the long term water balance of large catchments to land-use changes. It uses the FRER1D module for infiltration, RIVER1D module for flow routing, and the BOUSS2D module for groundwater flow.

Jankowfsky *et al.* (2010) describe the Peri-Urban Model for Landscape Management (PUMMA) created with LIQUID<sup>®</sup> to assess water fluxes in suburban areas for different landscape management practices. The PUMMA uses the URBS module for urban hydrological elements representing cadastral units, the FRER1D module for natural area such as forests and fields, HEDGE for vegetated field borders and riparian zones, SIMBA for retention basins and lakes, and RIVER1D for the drainage network. Each module has its own time step, enabling different hydrological processes to be simulated at different temporal scales.

### 3.1.5.3 Comments

Branger *et al.* (2010b) state that the strong aspects of LIQUID<sup>®</sup> are the flexibility provided to module developers to develop modules based on any type of process representation, the efficiency of the module coupling system, and the ability to synchronize time steps and data exchange between modules with different process conceptualizations. According to Branger *et al.* (2010a), LIQUID<sup>®</sup> was originally designed for research in hydrology and thus far had only been used by researchers from Cemagref and Grenoble University for research applications. Together with Branger *et al.* (2010b) they go on to state that LIQUID<sup>®</sup> has entered a “*mature development stage*” and future development will include better computational efficiency, parallel processing, parameter optimisation, stabilisation of numerical schemes, adapting traditional numerical methods to complex geometries, display of simulation results and user friendliness.

The details of the model coupling architecture used in LIQUID<sup>®</sup> were not clear in the limited amount of literature that could be found. LIQUID<sup>®</sup> is intended to be used in a modular modelling context, but could potentially be applied to whole models. In addition to its model coupling mechanism LIQUID<sup>®</sup> also offers many of the attributes of a traditional modelling framework. LIQUID<sup>®</sup> is intriguing in that it seems to build module code on the fly, which enables models to be easily ported to different operating platforms. LIQUID<sup>®</sup> is supported by a small collaborative research group in France, but does not appear to have been used outside this group and is proprietary software.

### 3.1.6 Earth System Modelling Framework (ESMF)

The ESMF is a specialised, standards-based, open source modelling framework and model architecture for coupling grid based climate and atmospheric models in a high performance computing environment (Hill *et al.*, 2004; Collins *et al.*, 2005). ESMF resulted from a

collaborative multidisciplinary project to which many of the largest earth science modelling centres in the USA contributed, and is used as the basis for some of the general circulation models used by these centres. Models of large and distinct, but interactive, domains such as atmosphere land and ocean are each represented by an ESMF Gridded Component class, where each domain is represented by some form of physical grid. Instances of the ESMF State class are used to exchange data between components, where each component can accept one or more input ESMF States and produces one or more output ESMF States. ESMF Coupler classes receive one or more input ESMF States as input and map them by means of spatial and temporal conversion routines to one or more output ESMF States, making provision for coupling models at different scales and grid representations. For the reason that ESMF is a specialised framework focussed on coupling grid based climate and atmospheric models, it was not considered further for this project.

### **3.1.7 Modular Modelling System (MMS)**

The MMS is described by Leavesley *et al.* (2002) as an integrated modular modelling framework developed to provide a research and application framework required to enhance development, testing, and evaluation of physical-process modules, facilitate coupling of selected modules to form custom models, facilitate the coupling of models and to provide a range of modelling analysis and support tools. They further describe MMS as a modular modelling framework enabling members of the scientific modelling community to address complex issues associated with the design, development, and application of distributed hydrological and environmental models in a collaborative manner, and provides a means of sharing advances in modelling algorithms and techniques. MMS was developed by the US Geological Survey (USGS) and was one of the forerunners in the development of modular modelling frameworks. MMS was not considered further in this project as Markstrom (2011) states that the USGS no longer distributes or supports MMS.

### **3.1.8 High Level Architecture (HLA)**

#### **3.1.8.1 Overview**

The High Level Architecture (HLA) is described by Dahmann *et al.* (1997) as a specification of a technical architecture for use across all classes of simulations in the US Department of Defence (Dahmann *et al.*, 1997). The Defence Modelling and Simulation Office (DMSO) of the US Department of Defence have developed HLA to meet the requirement for interoperability among new and existing simulations within the US Department of Defence.

Jagers (2010) describes HLA as a general purpose architecture for distributed real-time training and simulation environments involving tightly coupled networks in which data exchanges are frequent, but usually small. Lindenschmidt *et al.* (2005) describe HLA more simply as a computer architecture for constructing distributed simulations, and explain that it facilitates interoperability between different simulations and simulation types and promotes reuse of simulation software modules. Jagers (2010) makes the point that HLA is just an architecture and does not include an implementation. Jagers (2010) mentions that the baseline definition for HLA was completed in 1996 and was accepted as a general IEEE 1516 standard in 2000.

In the HLA a model or simulation or entity implementing the HLA interfaces is referred to as a *Federate* and a system of linked *Federates* is called a *Federation* (Dahmann *et al.*, 1997). The HLA has four main components, the HLA Rules (federation rules), the HLA Interface Specification which specifies interfaces between components (federates), the Runtime Infrastructure (RTI) via which data exchange occurs and the HLA Object Model Template. The HLA Object Model Template for recording information that describes a federation object model such as possible data exchanges between components (federates) that can be queried at run-time (Dahmann *et al.*, 1997; Jagers, 2010). The HLA Rules need to be followed which define the general architecture of the HLA. The HLA interface specification provides information on the services provided to the models by the RTI and by models to the RTI. The HLA interface specification defines the way services are accessed, both functionally and in a programmer's interface. The HLA Object Model Template provides a standard way to document descriptions of object models. There are two types of object model descriptions, the first being the HLA Federation Object Model (FOM) and the second is the HLA Simulation Object Model (SOM). The HLA FOM provides information about the set of objects, attributes and interactions within a linked system (Federation). The HLA SOM describes the set of objects, attributes and interactions a linked system (Federation) can provide at runtime. The Runtime Infrastructure (RTI) is the system that provides services to carry out the interactions between models within the linked system. Lindenschmidt *et al.* (2005) describe the RTI as being the core of the HLA, providing services to start and stop a simulation execution, control data transfer between linked simulations and to control time stepping among the linked simulations. They mention that the source code of models to be integrated into HLA needs to be modified to include RTI functionality.

### **3.1.8.2 Application**

Lindenschmidt *et al.* (2005) describe an implementation of HLA to simulate river water quality for the management of large river basins. They implemented HLA in Water Quality Simulation Program version 5 (WASP5) to couple its three submodels, DYNHYD for hydrodynamics, EUTRO for eutrophication and TOXI for sediment and micro-pollutant transport into a HLA federation. Previously in WASP5 there was no interaction between the EUTRO and TOXI submodels, and feedback between these two models and DYNHYD was also not possible. They report that the HLA implementation of WASP5 enabled improved transfer of information between the three submodels which lead to better predictive ability and uncertainty analyses. Lindenschmidt *et al.* (2005) stated that they were not aware of any other implementations of HLA for water resources modelling but mention six other applications in other fields.

### **3.1.8.3 Comments**

Lindenschmidt *et al.* (2005) concluded that the HLA provides a fast, simple means of coupling models together in a simple modelling system, additional capabilities such as whole model uncertainty analysis and representation of interactions and feedbacks between submodels, and could potentially be used as a docking mechanism to modelling systems such as OMS.

Though the details of the model linking architecture were difficult to understand from the limited literature found, its design appears to be sound with many similarities to OMS and TIME. HLA was primarily intended for applications in the defence domain, there is no real reason why it should not be used for coupling environmental models as demonstrated by Lindenschmidt *et al.* (2005). One main reason why HLA should not be considered for use in this project is that it does not have a wide list of implemented environmental models.

## **3.1.9 Common Component Architecture (CCA)**

### **3.1.9.1 Overview**

The Common Component Architecture (McCartney and Arranz) is described as a component architecture for scientific high-performance computing (HPC), specifically high-performance parallel computing (Armstrong *et al.*, 2006). Jagers (2010) states that the objective of the CCA Forum, founded in 1998, was to define a standard for a scientific, high-performance component architecture that includes HPC features not available in other

generic component architectures such as CORBA, COM, .NET and JavaBeans. He further states that the objectives of the CCA specifications were to maintain the performance of components, provide platform independent inter-component communication mechanisms, enable parallel computing across components, and allow for configuration of components before and during execution. Bramley *et al.* (2000) state:

*“The philosophy of CCA is to precisely define the rules for constructing components (or, in the case of existing applications, the software wrapping that makes them into components) and the specification of the required behavior that a component must exhibit for it to coexist with other components within a CCA framework.”*

Bramley *et al.* (2000) explain that the CCA consists of two entity types: Components and Frameworks. Components are the basic software units that are created, composed together and managed within a Framework to form applications. Frameworks also provide the essential services that components require to operate and interact, such as dynamic instantiation, coupling and invocation of methods. The CCA does not provide any specifications as to how the Framework is constructed enabling different frameworks to be constructed for different purposes. A CCA framework should provide support for SIDL, services to handle communication, security, thread creation and management, memory management and error handling, ability to instantiate and couple components and a repository for components (Jagers, 2010). Armstrong *et al.* (2006) mentions SCIRun, Ccaffeine, and XCAT as examples of CCA compliant frameworks.

The concept of a *port* is fundamental in CCA, where ports provide the public interface of a component through which it communicates. In CCA there are two types of port, Provides-port and Uses-port, where one may be connected to the other. A Provides-port is an interface of functions that the component implements and are executed by the component on behalf of the component's “users”. A Uses-port is connection point on the surface of the component where functionality the component requires can be implemented. In simpler terms, Armstrong *et al.* (2006) describe a port in the CCA as a resource (collection of subroutines) that can be either exported or imported from a component. Each CCA component needs to have a *setServices* method in order to be used within a CCA framework, where the responsibility of the *setServices* method is to definition of the ports that a component provides and uses. When two components exist in the same address space, a direct connection can be made between a Provides-port and a Uses-port, and when two components exist in different address spaces, a Uses-port instead holds a proxy to the remote Provides-ports.

Armstrong *et al.* (2006) state that it was recognised that scientists use a variety of programming languages, but the CCA does not force scientists to use a particular language. This is made possible by the CCA specification being written in Scientific Interface Definition Language (SIDL) coupled with the use of appropriate programming language bindings. Bramley *et al.* (2000) state that CCA components may be written in Java, Fortran, C or C++, but it is the responsibility of the framework to provide suitable infrastructure to enable interoperability between components from different languages.

### **3.1.9.2 Application**

Jagers (2010) mentions that CCA has been demonstrated to be interoperable with other frameworks such as the Earth System Modelling Framework and the Modelling Coupling Toolkit, and that the Ccaffeine framework has been successfully combined with the OpenMI 1.4 Java implementation. Zhou (2006) reports using the CCA and ESMF to couple climate models.

### **3.1.9.3 Comments**

This short review does not even start to cover the technical details or the literature related to the CCA, but it does serve to highlight that coupling of models and process modules is not exclusive to the environmental modelling domain. In addition, this short review of the CCA is of interest for two other reasons, first in that it indicates that there are means of linking modules across programming languages and platforms, and second in that there are means of coupling modules for parallel processing in a high performance computing environment. There is no real reason why CCA should not be used for coupling environmental models though one reason why it should not be considered for use in this project is that it does not boast a wide list of implemented environmental models.

### **3.1.10 Discussion and recommendation**

Several model linking mechanisms from the interface specification and modular modelling system approaches have been reviewed in the previous sections. At the start of the review these two approaches appeared to be quite distinct. However, in general there is no reason why an approach intended for linking whole models, especially legacy models, should not be used to link process modules which in reality are just small models. When linking either whole models or process modules, it is critical for the person, or people, doing the linking to have a clear understanding of the respective models or modules. Linking of models or



modules should be done by experts to produce a sound integrated modelling system for use by suitable trained, but not necessarily expert, model users. Blind and Gregersen (2005) sum this up by correctly pointing out that an integrated modelling system created by linking individually valid models does not imply that the integrated system as a whole is valid, and that collaboration between model specialists will be required. Modular modelling is an attractive concept but it is beyond the abilities of most model users, and even experienced model developers will have to be careful when composing models to ensure that the modules on which they are based are compatible with each other. Whole legacy models build a reputation over time. While custom models may be useful for modelling individual case studies they have no reputation that gives confidence in the results, assuming of course that the model has been correctly parameterised. There needs to be a balance between too much flexibility, making an architecture hard to implement, and too little flexibility, which will reduce the number of situations in which the architecture can be applied.

The systems reviewed could be categorised into two main groups: CCA, HLA and OpenMI which are purely interface specifications, whereas OMS, JAMS, TIME, LIQUID, ESMF and MMS are modelling frameworks which include a mechanism for linking models or process models. Jagers (2010) confirms this by pointing out that CCA, HLA and OpenMI in essence only define architectures and interfaces, HLA doesn't even have a reference implementation, but that the developers of CCA and OpenMI are creating reference implementations. OpenMI, OMS, JAMS, TIME, LIQUID, ESMF and MMS are designed primarily for use in the water and environmental modelling domain, though ESMF is specific to the climate, atmosphere domain. HLA was designed for use in the defence domain. CCA is a general purpose linking mechanism and is suitable for use in a high performance computing operating environment. The coupling interfaces defined by CCA, ESMF, OMS, OpenMI and TIME are similar in that they all use *initialize*, *run*, *finalize*, *get* and *set* method concepts, but differ in the amount of code needed to implement the interface, and in run time performance (Lloyd *et al.*, 2009; Jagers, 2010).

A simple quantitative assessment was performed on the systems reviewed, relative to the broad requirements stated at the beginning of Section 3.1, and the results of this assessment are shown in Table 3.5. Each system is rated for each requirement using a rating with the following scale: 1 = strong, 0 = average, -1 = weak.

Table 3.5 Simple quantitative evaluation of the systems reviewed where 1 = strong, 0 = average, -1 = weak

	OpenMI	OMS	JAMS	TIME	LIQUID	ESMF	MMS	HLA	CCA
Suited to water resource type models	1	1	1	1	1	-1	1	0	0
Ability model feedbacks	1	1	1	1	1	?	?	1	1
Minimal changes to model code	1	1	1	1	0	?	0	1	1
Linking across software platforms	0	1	0	1	1		?	1	1
Minimal impact on model run speed	?	?	?	?	?	?	?	?	?
Regarded as a standard	1	0	0	0	0	-1	?	1	1
Adequately supported	1	0	0	1	-1	0	-1	0	1
Widely adopted	1	0	0	0	-1	0	-1	0	1
Minimum financial burden	1	1	?	?	-1	?	?	?	1
Suitable for this project	1	1	0	1	-1	-1	-1	-1	-1

Based on this review and the simple evaluation presented above it was the opinion of the authors of this review that, in order of preference, the OpenMI, TIME and OMS systems should be considered for use in this project. The advantages of OpenMI are that it is generally accepted as a de facto standard, is strongly supported by the OpenMI Association, has been widely adopted by key research and commercial players providing a useful set of compliant models, and has been well documented. The advantages of TIME are its lightweight architecture, it is strongly supported by the eWater CRC, has been extensively implemented by the developers, providing a useful set of compliant water and environmental models even if they are tailored to Australian requirements, and has been well documented. The advantages of OMS are its lightweight architecture, that it has been moderately implemented by the developers, providing a small set of compliant water and environmental models even if they are tailored to USGS and USDA-ARS requirements. The selection of a model linking architecture and system for use in this project will be strongly influenced by the flow network model selected, and whether it supports one of the linking architectures reviewed.

### 3.2 Evaluation of Linkage Mechanisms in Selected Models

The review model linking mechanisms in Section 3.1 concluded that the OpenMI interface specification standard was the most appropriate linking mechanism for use in the project. The *ACRU* model does not provide a mechanism for linking and would need to be made OpenMI compliant. A review and technical evaluation of the linking mechanisms supported by the river network models MIKE BASIN, MODSIM and RiverWare was conducted in order

to determine the feasibility of linking these models to the *ACRU* model, preferably using OpenMI. This review and evaluation, together with the review and evaluation of river network model presented in Chapter 2, was intended to lead to the selection of a river network model for use in the project.

### 3.2.1 Evaluation criteria

The linking mechanism supported by the river network models needs to meet certain requirements to be considered for this project, as shown in Figure 3.17, which in order of preference are:

Requirement 1 - The river network model is OpenMI compliant.

Requirement 2 - The river network model is not OpenMI compliant, but the source code for the model is accessible, enabling modification of the code to make the model OpenMI compliant.

Requirement 3 - The river network model is not OpenMI compliant but provides access to the model engine, including access to parameters, input data and output data, which may enable an OpenMI compliant wrapper to be developed around the model.

Requirement 4 - The river network model has a linking mechanism which is not OpenMI compliant, but enables non-OpenMI links to be created with other models.

In the case of *ACRU*, the source code of the model is available and can be changed to make it OpenMI compliant. For a river network model that meets Requirement 1, no code changes will be required as it is OpenMI compliant. In the case of Requirement 2 a non-OpenMI compliant model, whose source code can be accessed, can be made OpenMI compliant by making code changes to the source code.

There is no access to the source code of the river network models being reviewed. Therefore, these river network models themselves cannot be made OpenMI compliant. In terms of Requirement 3 access is provided to the model engine of the non-OpenMI compliant model. In this case, wrapper code can be written around the model, which will interact with the river network model and provide the required functionality for OpenMI compliance, as illustrated for Requirement 3 in Figure 3.17. The wrapper code controls how the non-OpenMI compliant model will run and will enable the non-OpenMI compliant model to link to OpenMI compliant models. It is important to note that this type of solution may provide limited linking abilities, where only a small set of selected variables may be made

available for exchange between the models. This limitation will depend on the model selected. For Requirement 4, a non-OpenMI compliant model that offers a non-OpenMI linkage mechanism can still be linked to an OpenMI compliant model to meet the goals of this project.

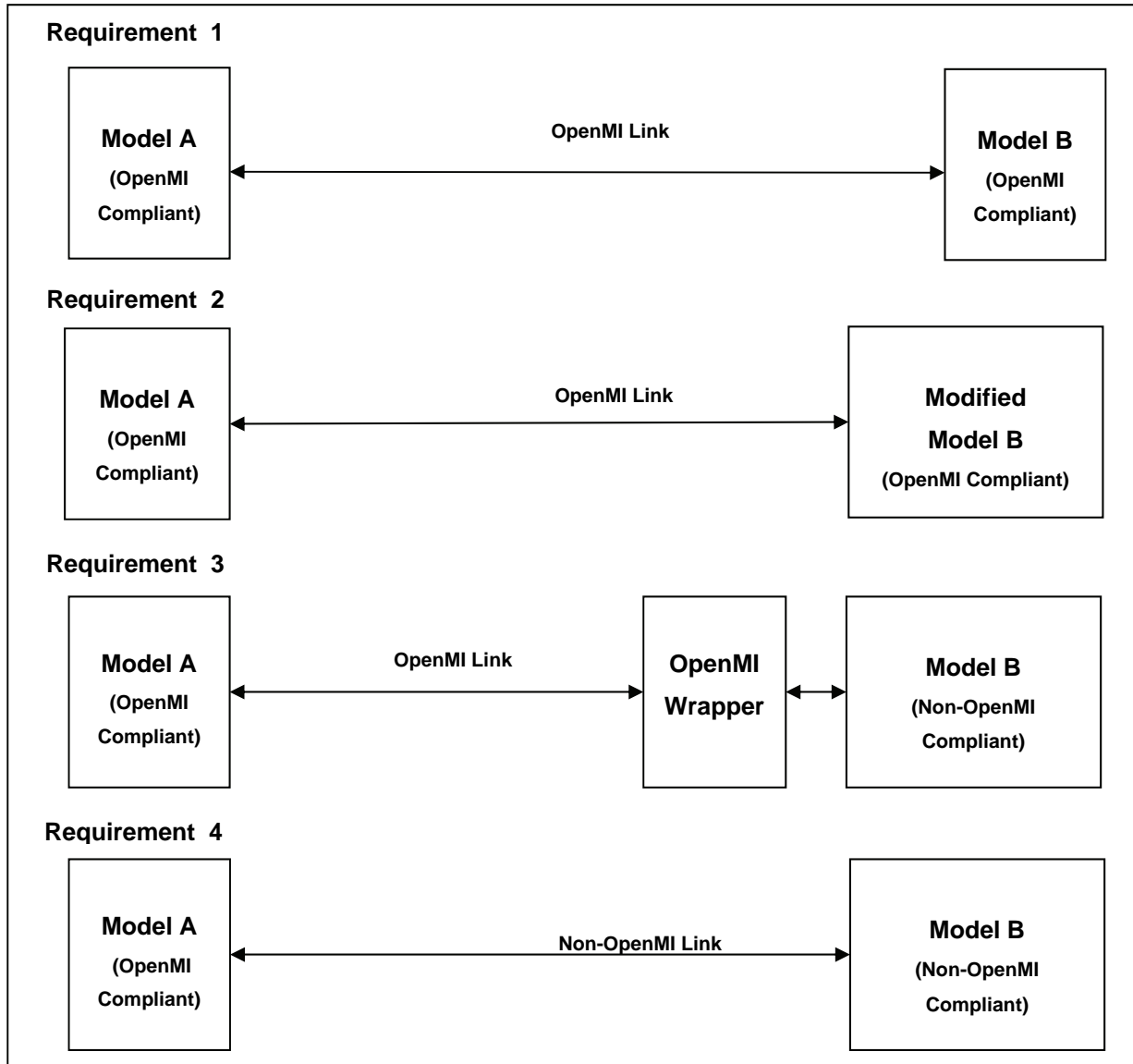


Figure 3.17 Illustration of river network model linkage mechanism requirements

For existing models to be suitable for migration to become OpenMI compliant requires a list of criteria to be met as defined in the guidelines for OpenMI, which can be found in Gijssbers *et al.* (2005). The criteria for a model to be suitable for migration to become OpenMI compliant are:

Criteria 1 - The model initialization should be separate from computation, with the ability to set the boundary conditions during the computation phase.

Criteria 2 - The model should allow access to information detailing the modelled quantities (variables) it can provide.

Criteria 3 - The model should allow access to the values of the modelled quantities it can provide for any requested point in time and space.

Criteria 4 - The model should provide run-time control to any outside entity and importantly it should provide methods to dynamically control the simulation (step-wise simulation, hot-start from any previous time-step).

Criteria 5 - A model that is time independent should still respond to a request and, in the case where it requires data from another model that is time dependent, it should pass a timestamp on in its request.

The key concept is the ability to dynamically control the simulation and the manipulation of data. These criteria have been defined for the conversion of a model itself. The river network models being reviewed will not be converted, as there is no access to the source code of these models. However, the river network models may provide access to their model engine, or have a linking mechanism which is not OpenMI compliant, but has the potential to be adapted or modified to support linking to OpenMI compliant models. In both cases, the access to the model engine and the non-OpenMI linking mechanism needs to be evaluated using these criteria. If these criteria are met by the river network models, then an OpenMI compliant wrapper can be implemented around the model, thus making it OpenMI compliant. For this evaluation the three river network models being evaluated do not need to satisfy Criteria 5, as they are all time dependent models.

Criteria 3 for OpenMI compliance, which requires the return of a value when requested, has resulted in the following conditions that need to be satisfied by a model being migrated as defined in the guidelines for OpenMI, which can be found in Gijsbers *et al.* (2005):

Condition 1 - The model needs to know at which point in time it is within its simulation. It needs to determine if it has not yet reached the requested time, it is at the requested time, or it has passed the requested time. Depending on the model and the context, this will further enable the model to know whether to simulate up to the requested time, to extrapolate a value if it's not possible to obtain a value for the requested time, or to search its buffer if it already has a value for the requested time.

Condition 2 - A requested value for a given time will need to be interpolated if the model requesting and the model providing the value are not using the same temporal or spatial scales.

Condition 3 - A model, when waiting for data to be computed, will need to return an extrapolated value.

These conditions are needed to enable the OpenMI architecture to work. The model should provide the functionality to meet these conditions. If a model does not meet these conditions then they can be met through the implementation of code, such as wrappers, but this would result in greater effort and time spent in converting the model to OpenMI.

The MIKE BASIN, MODSIM and RiverWare models are evaluated based on the requirements, criteria and conditions described above. Though the criteria are focussed on the OpenMI interface specification standard, the first four criteria would need to be met for any linking mechanism that links complex interactions between processes in two or more separate models.

The evaluation of each model consisted of two parts: (i) a review of information about the linking mechanism from literature and personal communications with the developers, and (ii) a technical evaluation in which a simple software implementation of the linking mechanism was created to better understand the mechanism and verify its ability to meet the criteria.

### **3.2.2 MIKE BASIN**

MIKE BASIN is a river network model developed by DHI Water and Environment. The version of MIKE BASIN used for this evaluation was MIKE BASIN 2011.

#### **3.2.2.1 Review**

MIKE BASIN 2011 is not OpenMI compliant and does not have an alternative linking mechanism (Hallowes, 2011). MIKE BASIN provides access to its computational core engine through Microsoft COM and .NET class libraries (DHI, 2011b). This enables the creation of customized solutions with any program or programming language that supports these technologies, such as Visual Studio .NET, Microsoft Excel or ArcGIS®. MIKE BASIN also enables users to write customized solutions using Visual Basic macro programming, as a part of its graphical user interface (DHI, 2011b). The access provided to the computational

core engine of MIKE BASIN is described briefly in this section. The information that follows on the COM and .NET class libraries has been derived from DHI (2011b) and DHI (2011c).

To use the computational core engine of the MIKE BASIN model, two assembly files need to be referenced, namely the DHI MIKE BASIN COM / .NET Engine class libraries (DHI.MikeBasin.Engine.tlb/dll) and the DHI MIKE BASIN Data Access Component class libraries (DHI.MikeBasin.Data.tlb/dll) (DHI, 2011b). The class libraries contain a workspace called *DHI\_MikeBasin\_Engine*, which contains two main classes, the *DHI\_MikeBasin\_Engine.Engine* class and the *DHI\_MikeBasin\_Engine.ModelObject* class. An instance of the *DHI\_MikeBasin\_Engine.Engine* contains instances of the *DHI\_MikeBasin\_Engine.ModelObject* class, where each instance represents a physical component, such as a node, reach or catchment, within the system being modelled.

The MIKE BASIN river network model's access to its model engine meets Criteria 1 to 4 needed for OpenMI compliance. The *DHI\_MikeBasin\_Engine.Engine* class provides initialization of MIKE BASIN separate to its computation, with the ability to modify boundary conditions during computation, thus satisfying Criteria 1. Criteria 2 and 3 are met by the *Engine* and *Data* classes which provide access to information on the quantities the model can provide and the values of those quantities for any requested point in time and space. The *DHI\_MikeBasin\_Engine.Engine* class also provides methods to dynamically control the simulation (step-wise simulation, hot-start from any previous time-step or even iteration within a time-step), which is required for Criteria 4. The MIKE BASIN river network model does not provide the functionality to meet the Conditions 1-3 defined above. Additional wrapper code would be needed, to implement the functionality to meet these conditions. Further details on the COM and .NET class libraries can be found in DHI (2011c). It is important to note that a MIKE BASIN model still needs to be configured using the ArcGIS® user interface prior to running it using a customized solution such as wrapper code. The class libraries do not enable the model structure to be changed, such as adding or deleting nodes.

The COM and .Net class libraries, provides access to the model engine of MIKE BASIN, which enables the development of an OpenMI wrapper for MIKE BASIN to support linking to OpenMI compliant models. This is demonstrated in a study of linking MIKE BASIN to MIKE SHE using OpenMI described by Christensen (2004). Christensen (2004) states, that customization of MIKE BASIN to be OpenMI compliant took a small re-engineering effort. The study conducted by Christensen (2004) proves that MIKE BASIN, through its model engine access, can be made OpenMI compliant. According to Hallows (2011), the latest

version of MIKE BASIN is not OpenMI compliant, but he does not foresee a problem with implementing an OpenMI compliant wrapper to support linking to OpenMI compliant models.

### 3.2.2.2 Technical Evaluation

From the review, the COM and .Net class libraries provided for MIKE BASIN, meet the criteria to enable the development of an OpenMI compliant wrapper for MIKE BASIN. A technical evaluation was carried out to verify that the .NET class libraries meet the criteria for OpenMI, which will aid in the evaluation. The technical evaluation involved the creation of a simple test application that interacted with the .Net class libraries.

The version of MIKE BASIN used for testing was sourced from the MIKE by DHI software 2011. It required the installation of ESRI ArcMAP™ 10.0 version of ArcGIS®, as the MIKE BASIN release 2011 runs as an extension within ESRI ArcMAP™ 10.0. It is important to note, both MIKE BASIN release 2011 and ESRI ArcMAP™ 10.0 require licenses. The test application was written in the Visual Studio 2008 development environment, using the programming language C#. The two assembly files referenced in the test application were DHI.MikeBasin.Engine.dll and DHI.MikeBasin.Data.dll. The MIKE BASIN model used for testing was initially set up and saved using the MIKE BASIN software's ArcGIS® interface.

The MIKE BASIN .NET class libraries were tested and they were found to meet Criteria 1-4. The *DHI\_MikeBasin\_Engine.Engine* .Net class has a method called *Initialize* that sets up the model separate from its computation. A *DHI\_MikeBasin\_Engine.Engine* object contains *DHI\_MikeBasin\_Engine.ModelObject* objects, representing entities such as dams, reaches and water users. These objects can be accessed using the *GetModelObject* method contained within the *DHI\_MikeBasin\_Engine.Engine* class. The *DHI\_MikeBasin\_Engine.ModelObject* class provides methods such as *SetInput* to change boundary conditions and input quantities during computation. The *Initialize* and *SetInput* methods enable Criteria 1 to be met.

The details of the input and output quantities of a MIKE BASIN model can be accessed using the *DHI\_MikeBasin\_Engine.ModelObject* class methods *GetInputSpecs* and *GetResultSpecs* respectively, which satisfies Criteria 2. The values of input and output quantities of a MIKE BASIN model are retrieved using the *DHI\_MikeBasin\_Engine.ModelObject* class methods *GetInputOriginalValue* and *GetCurrentResult* respectively. Similarly, the values of input time series and output time



series quantities are retrieved using the *DHI\_MikeBasin\_Engine.ModelObject* class methods *GetInputTSObject* and *GetResultsTSObject* respectively. Criteria 3 is met by being able to retrieve the values for both input and output quantities of a MIKE BASIN model, for the current time-step, using *GetInputOriginalValue* and *GetCurrentResult* methods and for a requested point in time and space, using the *GetInputTSObject* and *GetResultsTSObject* methods.

The MIKE BASIN .Net class libraries submit control to an outside entity, satisfying Criteria 4. An important aspect of this control with regards to OpenMI, is the ability to run the model on a time-step basis with the ability to hot-start from any previous time-step. In the test application the *SimulateTimeStep* and *AdvanceTimeStep* methods of the *DHI\_MikeBasin\_Engine.Engine* class were used to run the model on a time-step basis.

The test application successfully interacted with a MIKE BASIN model using the MIKE BASIN .Net class libraries verifying that Criteria 1-4 are met. The functionality provided by the class libraries was sufficient to satisfy Criteria 1-4 and no additional coding was needed. The support provided by the DHI group was good, with a response time to e-mail queries being on average one week. The literature describes MIKE BASIN as having the potential to be made OpenMI compliant and the technical evaluation has verified that MIKE BASIN meets the criteria for OpenMI.

### **3.2.3 MODSIM**

MODSIM is developed and maintained by Colorado State University (Labadie, 2006b). MODSIM 8.1 has been written using Microsoft Visual C++ .NET (Labadie, 2006b).

#### **3.2.3.1 Review**

MODSIM is not OpenMI compliant and does not provide an alternative linking mechanism, but like MIKE BASIN, does provide access to its model engine. MODSIM provides access to its model engine using public classes and variables. These classes can be accessed using custom code written using any of the supported programming languages in the .NET framework. Labadie (2006b) states that MODSIM can be accessed by external applications such as models running concurrently with MODSIM, without having to modify the original code.

To implement a solution to make MODSIM accessible through an OpenMI compliant wrapper requires access to the model engine which should meet Criteria 1-4. MODSIM provides this functionality through the *Model* class and the *TimeManager* class (Labadie, 2010c). The following brief descriptions of these two classes was found in Labadie (2010c). The *Model* class is the main class used to access the model engine of MODSIM. The *Model* class is structured such that MODSIM can be initialized separately from its computation. Labadie (2010c) states that through code customisation, access is provided to all model variables, prior to computation, during computation and after computation, and that boundary conditions can be modified during computation. This separate initialization and access to all model variables during computation meets Criteria 1. The *Model* class contains public methods and variables used to perform a MODSIM network simulation and access the data of the model. Access to time series information is provided by the *TimeSeries* class. The *TimeManager* class controls the model simulation time-steps in MODSIM. MODSIM therefore provides access to information about quantities and values of these quantities at a requested point in time and space, satisfying Criteria 2 and Criteria 3. The *Model* class and *TimeManager* class of MODSIM provides run-time control to any outside entity, which is required for Criteria 4. All public variables and object classes for MODSIM, such as the *Model* class and *TimeManager* class can be found within the *Csu.Modsim.ModsimModel* namespace. To adapt MODSIM to be OpenMI compliant, custom code, such as wrappers, will be required to access the model engine and implement the requirements needed for OpenMI compliance. MODSIM does not provide the functionality to meet the Conditions 1-3 defined above. Thus, additional wrapper code would be needed to implement the functionality to meet these conditions

### **3.2.3.2 Technical Evaluation**

The technical evaluation involved the creation of a simple test application that interacted with the MODSIM classes, to verify that the access to the model engine provided by MODSIM meets Criteria 1-4. MODSIM version 8.1 was used for testing and was downloaded from the MODSIM-DSS website [<http://modsim.engr.colostate.edu/version8.shtml>]. It required no additional installation files and did not require a license to be used. The test application was written in the Visual Studio 2008 development environment, using the C# programming language. The assembly files referenced in the test application were *ModsimModel.dll*, *NetworkUtils.dll* and *XYFile.dll*. A simple MODSIM model configuration was created using the MODSIM user interface for the purposes of testing.

A MODSIM model can be loaded by creating a new instance of the *Model* class, and loading the model into the new instance of the *Model* class, using the *XYFileReader* class found in the *XYFile.dll* library. The *Model* class also provides access to the boundary conditions of the model which can be set prior to and during running of the model, essentially the computation phase of the model. The ability to initialize and load a model and modify boundary conditions at computation time satisfies Criteria 1.

The *Model* class does not have methods that return a description of the quantities a MODSIM model can provide, but the quantities are accessible. In this simple test a list of the public variables of the *Model* class was created and displayed by the test application using reflection. The variables represented the input and output quantities of the model. A possible solution is to create two predefined lists of variables, with one list representing the input quantities a MODSIM model requires and the other list representing the output quantities a MODSIM model can provide. These lists would be defined in the wrapper that would be written around MODSIM to make it OpenMI compliant. Each quantity would have some description associated with it. An expert on the MODSIM model would be required to define the lists of input and output quantities, and their associated descriptions. The definition of these lists would be once-off and would apply to all MODSIM models. Therefore, although Criteria 2 is not met explicitly, Criteria 2 could be met when creating an OpenMI wrapper.

The time series quantities of a spatial object, known as nodes within the MODSIM model, can be accessed and updated through the *Node* class, which is contained in the *Model* class. These time series quantities are of type *TimeSeries* class and it provides functionality to access a time series value. The MODSIM model classes provide access to the value of a quantity for a given point in time and space satisfying Criteria 3. The *Model*, *Node* and *TimeSeries* classes provide access to its quantities, but a developer creating a custom solution that uses these classes would require knowledge on the quantities in order to use them effectively. The solution proposed above for the creation of lists of input and output quantities, with their associated descriptions, will aid in this process.

The MODSIM model loaded into a *Model* object can be run using the *RunSolver* method of the *Modsim* class. The *TimeStepManager* class within the *Model* object is responsible for the model simulation time-steps in MODSIM. Initially the model was run for one year by setting the simulation start date to 01/01/1981 and the simulation end date to 01/01/1982 using the MODSIM user interface. An attempt was then made to run the model on a single daily time-step basis. To run the model for a single day the *startingDate* and *endingDate* variables of the *TimeStepManager* class were set to 01/01/1981 and 02/01/1981 respectively. The model

was then saved and the *RunSolver* method was then called to run the model for one day (i.e. 01/01/1981). The model ran for the single day (i.e. 01/01/1981), however, it was not possible to repeat the process of setting the *startingDate* and *endingDate* variables to run the model for subsequent days in the simulation period. The access to the model engine of MODSIM does submit runtime control to an outside entity but the model could not be run on a daily time-step basis. The MODSIM model engine classes do not appear to fully meet Criteria 4. It is possible the approach taken to run the MODSIM model on a daily time-step basis was incorrect, but there is no documentation providing an example or instructions on how to do so. The approach taken was from the knowledge gained from the tutorial document on MODSIM by Labadie (2010c). Numerous attempts over an eight week period were made to get support from the developers of MODSIM on the approach required to run the model on a daily time-step, but there has been no response to date of the writing of this document.

MODSIM is not OpenMI compliant but provides access to its model engine using the classes *Model*, *Modsim* and *TimeManager* included in the MODSIM shared libraries. The access to the model engine was evaluated using a simple test application and it was established that Criteria 1-3 could be largely met, though additional code would need to be implemented to fully satisfy these criteria, which was beyond the scope of this technical evaluation and would require assistance from an expert on the model. The *Model*, *Modsim* and *TimeManager* classes submit runtime control to an outside entity, but the MODSIM model could not be run on a time-step basis. The review of MODSIM indicated that it had the potential to be made OpenMI compliant, however the technical evaluation indicated that MODSIM may not fully meet the criteria for OpenMI.

### **3.2.4 RiverWare**

RiverWare is a general river basin modelling tool developed by the Center for Advanced Decision Support for Water and Environment Systems (CADSWES) at the University of Colorado (CU) (Zagona *et al.*, 1998). This evaluation was conducted using the RiverWare 6.0.3 version of the software.

#### **3.2.4.1 Review**

RiverWare is not OpenMI compliant but has been linked to MODFLOW to model groundwater-surface water interaction, and provides an alternative linking mechanism (Valerio, 2008; Zagona, 2011). Changes had to be made to the MODFLOW and RiverWare

code, to implement a link between the models, to enable parallel execution of the models and data transfer between the models at run time (Valerio, 2008; Zagona, 2011). This type of linking mechanism can be described as a hard-coded link (Zagona, 2011). A hard-coded link requires code changes to be made to the participating models to implement the link. In this case the link implemented between RiverWare and MODFLOW will only work for that combination of models.

RiverWare provides a Data Management Interface (DMI), which is a mechanism that enables the transfer of data from RiverWare by directly linking to a data source or using external executable software (CADSWES, 2010b). The configuration of a DMI is done through a user interface, which allows for flexible configuration. Once the DMI has been set up, RiverWare calls the DMI to automatically export data from RiverWare or import data into RiverWare. There are two types of DMI, namely, the Control File-Executable approach and the Database DMI (CADSWES, 2010b).

The Control File Executable approach is of more interest than the Database DMI, as it has the potential to be used to link to a model by transferring data. The Database DMI is a direct connection between RiverWare and an external database (CADSWES, 2010b). The following details on the Control File Executable DMI were found in CADSWES (2010b). The Control File Executable approach enables the transfer of large amounts of data between RiverWare and an external data source. Data can be exported from or imported into RiverWare using this approach. A control file is a list describing the data that will be exported for an export DMI or imported for an import DMI. An external executable can be called by the RiverWare DMI facility which acts as a mediator for the transfer of data between RiverWare and external sources.

RiverWare also provides a batch mode option, which enables the execution of RiverWare without using the graphic user interface (CADSWES, 2010a). The following description of the batch mode can be found in the technical documentation by CADSWES (2010a). When used in batch mode RiverWare reads and executes the commands contained in a RiverWare Command Language (Rcl) script file. This enables external applications to control the execution of RiverWare by creating these Rcl script files and calling RiverWare in batch mode to execute the commands. Rcl supports basic model run commands such as, setting up a model run, calling DMIs to transfer data, executing the model and saving the model.

The DMI and batch mode capability can be described as the linking mechanism offered by RiverWare. According to Zagona (2011), RiverWare, through its DMI and batch mode

capability, can transfer data to other models at each time-step and wait for values to be returned. This capability also enables RiverWare to be called by external software and for data to be transferred between the external software and RiverWare. RiverWare has been integrated into the U.S. Army Corps of Engineers' Corps Water Management System (CWMS) using the DMI and batch mode capability (Evans *et al.*, 2006). CWMS is a model support framework which provides standard support interfaces or utilities to models contained within the framework (Evans *et al.*, 2006). Cotter *et al.* (2006) states that CWMS controls the execution of RiverWare using Rcl script files and the transfer of data using DMI's between itself and RiverWare. This integration provides an example of the adaption or modification of the linking mechanism provided by RiverWare to link to an external application.

RiverWare does not meet the Criteria 1-4 or Conditions 1-3, as it does not provide access to its model engine. The batch mode capability enables the execution of RiverWare using Rcl scripts. The initialisation is not separate from computation, and boundary conditions cannot be changed during computation. The DMI does provide access to input and output data, but this is not direct access to the quantities provided or the values of the quantities. Additional code could be written to determine the quantities that are provided and the values of the quantities. The batch mode capability does not directly submit control to an outside entity. Rather the outside entity has to control the execution of RiverWare using Rcl scripts. This again would require additional code to automate this process.

The DMI and batch mode capability of RiverWare has the potential to be used together with code to support linking to OpenMI compliant models. A possible solution would be to write wrapper code in conjunction with Rcl script files and Control File Executable DMIs, which controls the execution of RiverWare and the transfer of data between RiverWare and the OpenMI compliant model it is linked to. The wrapper code would need to provide the necessary functionality to ensure OpenMI compliance of RiverWare.

#### **3.2.4.2 Technical Evaluation**

RiverWare is not OpenMI compliant but does provide an alternative linking mechanism by using its DMI and batch mode capabilities. The DMI and batch mode capability do not fully satisfy the Criteria 1-4. A possible solution is the use of the RiverWare DMI and batch mode capability in conjunction with custom code to attempt to meet these criteria. This section

describes the prototype implementation and testing of a proposed solution. The proposed solution will be referred to as the custom RiverWare solution in the rest of this document.

The custom RiverWare solution required the implementation of code that will interact with RiverWare using the DMI and batch mode capability, and will provide the functionality to outside entities required to meet Criteria 1-4. The custom RiverWare solution would provide methods similar to the methods of the model engines of MIKE BASIN and MODSIM. This enables the implementation of wrapper code to make the custom RiverWare solution OpenMI compliant, therefore enabling the linking of RiverWare to other OpenMI compliant models. The proposed solution is represented in Figure 3.18. The custom code is essentially wrapper code mimicking a model engine similar to MIKE BASIN and MODSIM, but to avoid confusion with the wrapper code used to make a model OpenMI compliant, it will be referred to as a custom engine. It is also important to note that the term used in RiverWare for boundary conditions and quantities is slots.

The custom RiverWare solution is not efficient, but the purpose of this evaluation was to determine if it is possible to meet Criteria 1-4 for OpenMI. The technical evaluation will involve a simple implementation of the custom RiverWare solution, and a test application to verify whether it meets Criteria 1-4.

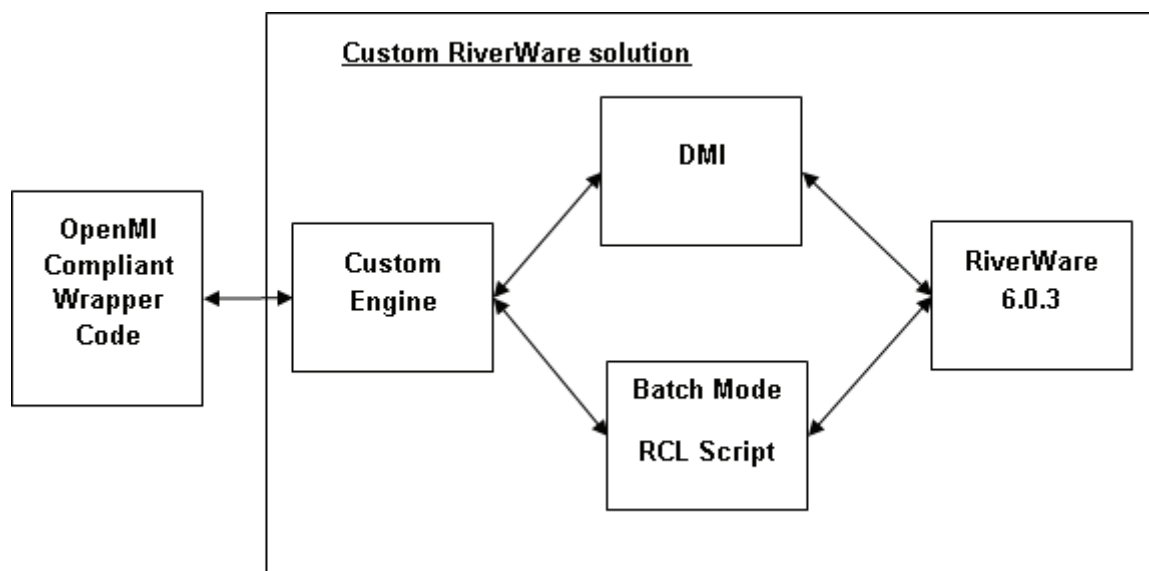


Figure 3.18 Structure of the proposed custom RiverWare solution

The version of RiverWare used for testing was Version 6.0.3. RiverWare is a standalone application and does not require additional software packages to be installed but it does require a license. The custom code and the test application are both written using the Visual

Studio 2008 development environment, using the C# programming language. A basic model was set up using RiverWare, for the purposes of testing.

RiverWare does not provide any initialization options as it is an executable. The custom engine was coded as a shared library, with a method called *Initialize*, which initializes the custom engine and requires the file path to the RiverWare model file (.mdl) as a parameter. This parameter will be used when interacting with RiverWare using the DMI and batch mode capabilities. The custom engine's *Initialize* method satisfies the first part of Criteria 1, which is, initialization should be separate from computation

To satisfy Criteria 2, a private method called *RetrieveModelSlots* was written, which is called from the *Initialize* method which retrieves the input and output slots (quantities) of the RiverWare model. This was achieved by creating a RCL script, which loads the model, retrieves the slots and writes this information to an output file. The custom engine creates and executes this Rcl script file using the RiverWare batch mode capability. The custom engine then reads the output file and stores the slots' details. A method *GetSlotSpecs* was defined to return the description of the quantities the RiverWare model provides, which has been read into the custom engine using *RetrieveModelSlots* method. The test application calls the *GetSlotSpecs* method and displays the description of the quantities in the RiverWare model.

Data is imported to and exported from RiverWare using the DMI. A control file is used to specify the data that will be either exported from or imported to RiverWare. An external executable is called by the RiverWare DMI which acts as a mediator for the transfer of data between an external data repository and RiverWare. For an import DMI, the executable provides the data files in a format which RiverWare understands and can import into the model. For an export DMI, the executable reads data files exported by RiverWare from a model and carries out processes such as storing the data in an external source or creating a report. The custom engine does not require an executable for the DMI because there is no transfer of data to an external data source.

The custom engine retrieves data from slots in a RiverWare model using a DMI and a RCL script. The method *GetSlotData* was coded to retrieve data from the model. The data is returned by the method *GetSlotData* in the form of a string to the calling code. In order for this method to work a DMI had to be set up in the model called *DMI\_Output\_Data* which points to a blank control file called *Output.Control*. For every RiverWare model that interacts with the custom engine, the DMI *DMI\_Output\_Data* needs to be created, which points to a



blank control file called *Output.Control* using RiverWare. The name of the DMI has to be *DMI\_Output\_Data* and the control file has to be named *Output.Control* in order for the custom engine to work. An object within RiverWare represents a spatial feature. A spatial feature (or object) and a slot need to be provided as parameters for the *GetSlotData* method. A new *Output.Control* file is created using the spatial object and slot, to specify the data to be exported and the file the DMI needs to write the exported data to. The file is called *TempOutput.dat*. Once the DMI is configured, a RCL script file is created and executed using the batch capability of RiverWare to invoke the *DMI\_Output\_Data*. The *GetSlotData* method then extracts the data in the file, and for this custom solution returns the data as a string. The test application then displays this data. In a proper implementation of this solution, the data will be converted and returned as a time series or a scalar type depending on the description of the slot. The *GetSlotData* method does not take a date-time value but this does not mean that a value from a time series type slot cannot be returned for a specified time. This was not implemented in the prototype custom engine as the work required to carry this out is beyond the scope this technical evaluation. The time series type slot values returned are for the date-time the model was run up until, which is the end date-time of the last simulation run. For example, if the model was run from 01/01/1981 00:00 to 21/01/1981 00:00, the values of the time series type slot returned will be for a simulation from 01/01/1981 00:00 to 21/01/1981 00:00. Values that have not been calculated for a time series type slot as yet, have the value "NaN" (Not a Number). The time series type slot values returned can be accessed and therefore it is possible to return the value from a time series slot for a particular time. If the value has not been calculated a null value could be returned. In a full implementation of this solution, the method returning data would accept a date-time parameter and either return a value or a null value indicating value has not been reached. Using the DMI and RCL scripts it is possible to return a value for a requested point in time and space, satisfying Criteria 3.

An initial attempt was made to meet Criteria 4 by trying to run the RiverWare model using code in the custom engine and RCL script commands in a daily stepwise manner. The RiverModel ran for the first day but failed to run for the days that followed. The reason is that the state of slots of a previous step of a simulation are not maintained. A simulation step in the model requiring values of slots from a previous step causes errors in the model. According to Neumann (2011a), the reason that the state of slots is not maintained, is that the RCL command *StartController* which is used to run the model, clears output slot values which have been saved in a previous run.

A solution to this problem was proposed by Neumann (2011b). The solution entails the running of the model for the first time-step, and then saving the slots or data whose state needs to be maintained, using an export DMI. The export DMI would be invoked after the *StartController* RCL command. In the second time-step an import DMI could be used to import the slots or data exported in the first time-step. This import DMI is invoked between the *OpenWorkspace* and *StartController* command. This would ensure the state of the values of slots are now maintained and can be used in the second time-step simulation. Similarly, once the model has run for the second time-step, the slots whose state needs to be maintained would be saved using an export DMI. Again the export DMI is invoked after the *StartController* RCL command. This process would be followed for each time-step. This solution would work as the state of slot values are maintained and also enables boundary conditions to be changed, but this solution requires more effort and time as it is more complex than a simple implementation of the custom RiverWare solution. The complexity comes in deciding which slots' state needs to be maintained. This would require additional code and knowledge of the slots, and how they are used in calculations in the model.

For the purpose of this evaluation, a prototype version of this solution was implemented. At initialization all slots are exported from the RiverWare model, which the custom engine reads and stores. The method to run the RiverWare model on a time-step basis, reads the slots that are required for the next time-step, which are stored in the custom engine. It then imports these values into the RiverWare model, runs the model and then exports the slots to files, which the custom engine reads and stores the slots with a date-time stamp flag. For each time-step the slot values imported into the RiverWare model are the previous time-step values, and in the case of the first time-step the "Initial" values. This allows the custom engine to maintain the state of slots and the model to step through its entire simulation period. The solution implemented succeeded in running the model in a stepwise fashion, satisfying Criteria 4. In addition, the maintenance of the state of slots within the model engine, specifically slots representing boundary conditions, provides the ability to change or set the boundary conditions during the computation phase, thus satisfying the second part of Criteria 1.

The results of the daily stepwise simulation of the RiverWare model needed to be verified against results from a simulation run for a year of the same RiverWare model. A comparison was carried out on a single result slot in the model for the two types of simulation. It was found that there was a difference between the values calculated for the stepwise simulation and the simulation run for a year. According to Neumann (2011a) the reason that the results could be different is there could be slots that are not being maintained, and also the model

could behave differently for a standard simulation and the stepwise simulation. Further investigation and code implementation would be needed to ensure that the daily stepwise simulation of the RiverWare model calculated the correct values, but was beyond the scope of this technical evaluation due to its complexity and expert knowledge that is required. It is rather left for the full implementation of the custom RiverWare solution if the model is chosen for use in this project.

The custom RiverWare solution for this evaluation could not meet all the criteria required for OpenMI compliance. An alternative solution to create a non-OpenMI link between RiverWare and an OpenMI compliant model was also explored conceptually, which required a RiverWare model to only import output data from an OpenMI compliant model into an input slot. A RiverWare model could import data from an OpenMI compliant model on a time-step basis using an import DMI using rules. The import DMI would interact with an executable that triggers an OpenMI compliant model to run for the requested time-step. Once the OpenMI compliant model had run for the time-step, the executable would read the returned data and convert it into the format required for the import DMI. The import DMI would then import the data into the RiverWare model. The RiverWare model would continue its execution and interact in future time steps with the OpenMI compliant model to retrieve input data. A problem is foreseen with this solution which is, for each time-step the executable is called by the import DMI, the OpenMI compliant model and any other OpenMI compliant model linked to it needs to be loaded into memory for every time-step. This solution would then also require some means of maintaining the state of values used within RiverWare.

In summary, RiverWare's DMI and batch mode capability do not fully satisfy Criteria 1-4. A proposed custom RiverWare solution involving the DMI and batch mode capability was described and an attempt at a prototype version of the solution was carried out. The prototype custom solution was also not able to meet all the criteria. The main problem encountered was the use of the RiverWare executable using RCL commands. It was designed to execute batch tasks and not to maintain the state of slots. A solution to maintain the state of slots to enable the model to run in a step wise fashion was proposed by Neumann (2011b). A prototype implementation of the solution that maintains the state of slots was implemented and the RiverWare model was able to run in a daily stepwise simulation. It was found that the results of the daily stepwise simulation were incorrect. Further investigation and code implementation to get the correct result is beyond the scope of this technical evaluation. It is also important to note that the approach taken is not efficient as the RiverWare model had to be loaded into memory for any interaction between it and the custom engine, and then released from memory, making the custom RiverWare solution

simulation slow. A more efficient solution was discussed using just an import DMI to link to an OpenMI compliant model and it was foreseen the OpenMI compliant model would suffer the same pitfalls as the batch mode of RiverWare. The support provided for RiverWare was impressive, with a short response time of one day and valuable feedback was provided by the developers.

### 3.2.5 Results and recommendation

None of the three river network models (MIKE BASIN, MODSIM and RiverWare) evaluated is OpenMI compliant. MIKE BASIN and MODSIM do not support an alternative linking mechanism, but do provide access to their model engines, satisfying Requirement 3 as shown in Table 3.6. RiverWare provides an alternative linking mechanism through its DMI and batch mode capabilities, satisfying Requirement 4 as shown in Table 3.6.

Table 3.6 Comparison of river network models based on requirements

Requirements	MIKE BASIN	MODSIM	RiverWare
<b>Requirement 1</b> OpenMI compliant.	✘	✘	✘
<b>Requirement 2</b> Non-OpenMI compliant, access to source code.	✘	✘	✘
<b>Requirement 3</b> Non-OpenMI compliant, access to model engine.	✓	✓	✘
<b>Requirement 4</b> Non-OpenMI compliant link.	✘	✘	✓

MIKE BASIN is the only model that satisfies Criteria 1-4 necessary for the creation of OpenMI compliant wrappers, as shown in Table 3.7. MODSIM satisfied Criteria 1-3 but not Criteria 4. In the review of RiverWare, it was found that it did not satisfy Criteria 1-4. Through the implementation of the custom RiverWare solution it was demonstrated that Criteria 1-3 could be satisfied, but not Criteria 4. None of the models evaluated meet the Conditions 1-3, therefore additional code would be required when developing the wrappers in order to meet these conditions.

MIKE BASIN is the only model that satisfied OpenMI Criteria 1-4 and did not require customization. The support offered by DHI for this evaluation was good, with an average response time of one week. The technical evaluation of MIKE BASIN was the least complex and time consuming.

Table 3.7 Comparison of river network models based on OpenMI criteria

	<b>MIKE BASIN</b>	<b>MODSIM</b>	<b>RiverWare</b>	<b>Custom RiverWare Solution</b>
<b>Criteria 1</b>	✓	✓	✗	✓
<b>Criteria 2</b>	✓	✓	✗	✓
<b>Criteria 3</b>	✓	✓	✗	✓
<b>Criteria 4</b>	✓	✗	✗	✗

MODSIM could not fully satisfy Criteria 4. It is possible that the approach used to get a MODSIM model to run on a time-step basis was incorrect. Attempts have been made to contact the developers of the MODSIM model to determine if it is possible to run the model on a time-step basis, using the classes provided by MODSIM, but there has been no response to date of writing of this document.

In the case of RiverWare a possible solution was the use of its DMI and batch mode capability and custom code in the form of a custom engine and an OpenMI compliant wrapper to adapt RiverWare for linking to other OpenMI compliant models. A prototype version of the solution proposed to make RiverWare meet the OpenMI criteria was implemented, but did not result in the criteria for OpenMI being fully met. RiverWare was the most complex and time consuming evaluation. The technical evaluation reached a point that, if it were to continue further, it would go beyond the scope of a technical evaluation and would rather be a full implementation of the solution. The custom RiverWare solution was inefficient in terms of its use of memory resources. It would be possible to meet all the criteria for OpenMI, using the DMI and batch mode capability, but it would probably be an easier task to migrate RiverWare to OpenMI, though this would have to be done by the developers of RiverWare. The support provided for RiverWare was excellent, with an average response time of one day, which aided in the implementation of the prototype solution.

The best model for use in this project would be the one that satisfies all the OpenMI criteria, requires minimal or no customisation to satisfy these criteria and has a good support system for the model. If all these requirements are satisfied, OpenMI compliant wrapper code can be written around the model to support linking to OpenMI compliant models. Based on this evaluation it can be concluded that RiverWare and MODSIM do not satisfy these requirements but MIKE BASIN does, making MIKE BASIN the strongest of the three river network models evaluated for this project.

## 4 ACRU MODEL DEVELOPMENT

DJ Clark

The requirement for integrated water resources management will require the integration of models representing specific domains to provide a systems perspective for water management decisions to support the implementation of the National Water Act. One of the objectives of this project was to provide a daily time step hydrological model that is capable of modelling the varying hydrological responses within the terrestrial hydrological system at suitable spatial and temporal scales to represent real world complexity. The *ACRU* hydrological model was proposed as a suitable model to represent land based hydrological processes for the following reasons:

- It has been developed and applied extensively in South Africa and is on the South African Department of Water Affairs (DWA) list of recommended models;
- The physical conceptual nature of the model makes it suitable for modelling a variety of land use scenarios;
- The object oriented model structure and object oriented Extensible Markup Language (XML) input file structure is capable of representing real world complexity;
- It operates at a daily time step, which makes it suitable for operational modelling;
- The object oriented model structure enables parallel processing which enables feedbacks between catchments to be modelled;
- It includes water quality modules for sediment yield, salinity, and nitrogen and phosphorus modelling;
- It can be easily adapted to provide additional functionality required for operations modelling; and
- It includes the concept of water ownership which is necessary for water accounting.

A brief background to the *ACRU* model is given to provide the context for *ACRU* development work within this project. Within this project it was necessary to make changes to the *ACRU* model and the design of its associated model input files to ensure that the *ACRU* model is suited for used in both water resources planning and operations modelling and is capable of representing real world complexity. Several changes were made to the ModelData and ModelConfiguration XML schemas used for *ACRU* model input to refine the design and include new functionality such as scenario management, the storage of state data required to hot-start the model, a means of storing dynamic data, use of forecast data and improved linkages to external data files. These changes to the schemas required

corresponding changes to be made to the .Net and Java *XmlModelFile* libraries and the *ACRU* model itself.

## **4.1 Background**

The *ACRU* model is described in Schulze *et al.* (1995b) as a physical conceptual agrohydrological model operating at a daily time step. The *ACRU* model is further described in Schulze *et al.* (1995b) as a versatile total evaporation model that is sensitive to climate, land cover/use and land management practices. These characteristics have resulted in *ACRU* being used for a variety of purposes including: climate change assessments, land use studies, crop yield modelling, water resource availability studies, reservoir yield analysis, crop water requirements and design hydrology (Schulze *et al.*, 1995b). The purpose of this section is to provide some background to the history of the development of *ACRU* model so that the context of the work included in this project is clearly understood.

### **4.1.1 The *ACRU* 3.00 version**

The *ACRU* model was originally developed as part of a distributed catchment evapotranspiration study conducted in the early 1970's in the Natal Drakensberg region of South Africa (Schulze, 1975). The early development and use of the *ACRU* model are further described in Schulze *et al.* (1995b). Though the *ACRU* model has been developed, tested and applied in South Africa it has also been tested and applied internationally. Since its inception the *ACRU* model has been under continual development and refinement by staff and post graduate students in what was the School of Bioresources Engineering and Environmental Hydrology (BEEH) at the University of KwaZulu-Natal. The Centre for Water Resources Research (CWRR) at the University of KwaZulu-Natal is now the custodian of the *ACRU* model. This development and refinement has taken place as part of numerous hydrological studies, many of which were funded by the South African Water Research Commission. The *ACRU* model was developed in the FORTRAN programming language. This FORTRAN version of the *ACRU* model as described in Schulze *et al.* (1995b) will be referred to in this document as the *ACRU* 3.00 version of the model. The *ACRU* 3.00 version of the model used text files for model input, consisting of the main model input file (or "menu" file) and Single, Composite and CompositeY2K format time series input files.

The *ACRU* 3.00 version is a stable computationally efficient version of the model but had some limitations which prompted its restructuring into an object oriented model structure in the Java programming language. Some of these limitations included:

- the need for a more modular model structure that enabled easier model development,
- the need to keep up with changes in computer programming technology,
- the need to be able to model more complex hydrological system configurations, and
- the need for different components of the hydrological system can be run in parallel (parallel processing), enabling the transfer of water between components of the hydrological system to be effected at each modelling time step.

#### **4.1.2 The *ACRU2000* version**

The restructuring of the *ACRU* model described in Kiker (2001) and Kiker *et al.* (2006) took place between January 1999 and March 2002 as part of the WRC Project 636. The restructuring resulted in the *ACRU2000* version of the model. The model structure was changed but the same hydrological process algorithms were used as in the *ACRU* 3.00 version of the model. The restructured *ACRU* model offers several advantages:

- the object oriented structure enables easier model development,
- the object oriented structure enables more complex hydrological system configurations to be modelled, and
- different components of the hydrological system can be run in parallel, though there are performance trade-offs for parallel processing due to the quantity of information loaded into memory at one time.

The object oriented structure of the model includes three main types of objects: *Component* objects which represent the physical components of the hydrological system being modelled (e.g. subcatchments, rivers, dams, vegetation, soil), *Process* objects which represent the hydrological processes through one or more algorithms (e.g. evapotranspiration, runoff, infiltration) and *Data* objects which contain the parameter or variable data values that describe the *Component* objects. Additional object types include *Model* which serves as the main object container for a model run and various model input, output and configuration objects.

The restructured *ACRU* model enabled several new modelling modules to be developed, including: *ACRUSalinity* for salinity modelling, *ACRU\_NP* for nitrogen and phosphorus modelling, *ACRU\_Cane* for modelling sugarcane yield under irrigation, and a module for



modelling dam and river operating rules. The development of these new modules demonstrated the robustness of the new object oriented structure, and also highlighted some areas for improvement.

The *ACRU2000* version of the model also used text files for model input, where the “menu” file was divided into a single “control menu” file and a set of separate “land segment menu” files, one for each subcatchment, together with the Single, Composite, CompositeY2K format time series input files and a new *ACRU-CSV* time series input file format. However, the restructured *ACRU* model could not be used to its full potential due to the non-object oriented nature of these model input files used to store the model input data and information.

#### **4.1.3 The *ACRUXml* version**

Between August 2004 and June 2008 the SPATSIM hydrological modelling framework was restructured and extended as part of WRC Project K5/1490 to produce the SPATSIM-HDSF hydrological modelling framework. The SPATSIM-HDSF framework includes a generic extensible database structure, tools to view, edit and analyse this data, an ArcGIS extension and several hydrological modelling tools and utilities. The *ACRU* model was modified to run from within the SPATSIM-HDSF framework and to be able to read from and write to SPATSIM-HDSF databases. The further development of the SPATSIM-HDSF framework and modifications to the *ACRU* model are described in Clark *et al.* (2009). This version of the *ACRU* model will be referred to as *ACRUXml* in this document.

The SPATSIM-HDSF framework includes a simple Graphical User Interface (GUI) tool that enables a user running a selected model within the framework to link a model input parameter or variable to the location of the relevant data value within a SPATSIM-HDSF database. These parameter/variable and data source links are stored internally within a SPATSIM-HDSF database. However, for a complex model such as *ACRU* which includes a large number of possible parameters and variables this means of linking model parameters and variables to a data source would not be suitable. Therefore, for *ACRU* a model specific GUI application is required to configure and edit model input data and data links used by the model. For this purpose the ConfigurationEditor application was developed. In addition some form of model input file was required for the *ACRU* model to store model configuration information and to enable the links between model variables and the location of the relevant data value within a SPATSIM-HDSF database to be recorded. For this purpose a prototype XML based model input file structure was developed. The XML based model input file structure sought to address four main requirements:

- Provide an object oriented input file structure to complement the object oriented structure of the *ACRU* model thereby enabling the restructured model to be used to its full potential;
- Provide a data model that is extensible such that new model parameters or variables can be accommodated without changes to the data model or to the software utilities that read from or write to the data model;
- Provide a structure for storing actual data values or references to where data values are stored, such as in a SPATSIM-HDSF database; and
- Provide a structure for storing additional information that describes the model parameters or variables for use in the ConfigurationEditor application.

These requirements were met with the design of two XML file schemas, the ModelData schema and the ModelConfiguration schema. The ModelData schema provides a data model for storing model input data in an object oriented structure suited to the *ACRU* model including *Component* elements and *Data* elements. A different implementation of the ModelData schema would be used for each configuration of the *ACRU* model. The ModelConfiguration schema provides a data model for storing information about model parameters and variables for use in *ACRU* and associated software utilities such as the ConfigurationEditor. A single implementation of the ModelConfiguration schema would be used for all configurations of the *ACRU* model. A different implementation of the ModelConfiguration schema would only be required if changes were made to the *ACRU* model. The details of the initial design of the ModelData and ModelConfiguration schemas can be found in (Clark *et al.*, 2009). Although the ModelData schema, ModelConfiguration schema and ConfigurationEditor were designed and developed primarily for the *ACRU* model, they were designed in such a way that they could easily be applied to other hydrological models. An advantage of using XML files is that they are programming language and platform independent. To facilitate the reading writing and editing of ModelData and ModelConfiguration XML files the XMLModelFiles software library was created in both the C-Sharp (C#) and Java programming languages. The main components and data flows for the *ACRUXml* model and the SPATSIM\_HDSF modelling framework are shown in Figure 4.1.

Following the completion of WRC Project K5/1490 a follow-on project WRC Project K5/1870 was initiated by the DWA to enable further development of the SPATSIM-HDSF modelling framework and the *ACRU* model and to provide user support. Software development work in WRC Project K5/1870 focused on debugging and further development of tools within the

SPATSIM-HDSF framework especially time series analysis tools and extensive further development of the ConfigurationEditor.

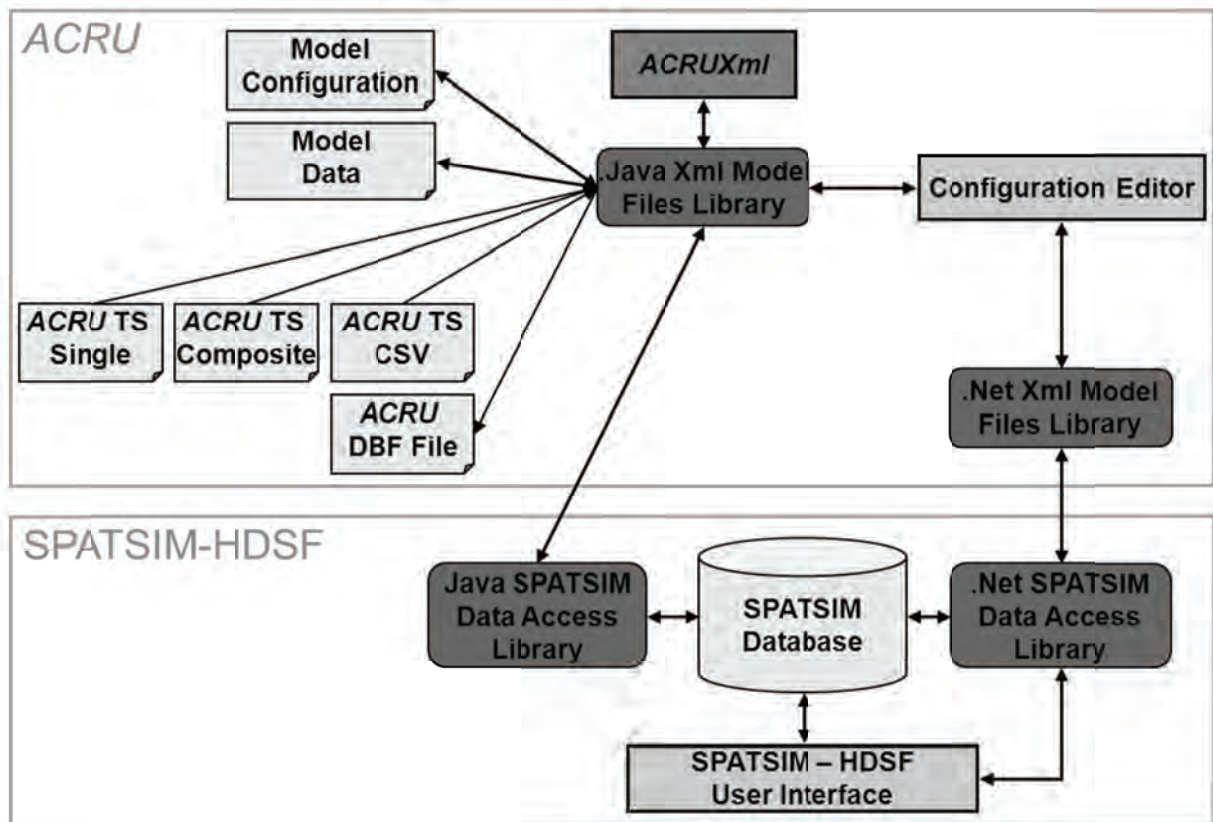


Figure 4.1 Main components and data flows for *ACRUXml* and SPATSIM\_HDSF

## 4.2 Development of XML Input Files

The design of both the ModelData and the ModelConfiguration schemas have been revised as part of this project and the design of these files are not expected to change substantially from this point. Some of the design refinements have been made in response to requirements and problems encountered in the implementation of these schemas for use with *ACRUXML*, and other refinements have been made to include proposed new functionality such as scenario management, the storage of state data required to hotstart models such as *ACRU*, use of forecast data and improved linkages to external data files. These changes to the schemas have required corresponding changes to be made to the XmlModelFiles libraries. A key requirement for the design of these model input data files was that they should be designed in such a way that if it was necessary to add a new variable to a model, then this could be done without changing the design of the ModelData and the ModelConfiguration schemas or the software utilities that use them. In this section the term “software utilities” will be used to refer generically to any software utility, for

example the ConfigurationEditor software, which may be developed to enable model users to interact with model input data stored in the XML based model input files described in this section.

#### 4.2.1 ModelData schema

Schema diagrams of the main elements of the initial and revised versions of the ModelData schema are shown in Figure 4.2 and Figure 4.3 respectively to help illustrate the design changes that have been made to this schema. An implementation of the ModelData schema will be referred to as a “ModelData file”, therefore a ModelData file is a populated XML file that obeys the ModelData schema. Each of the main schema elements will be briefly described to explain its purpose and important changes to the schema will be described in more detail.

The *ModelVersion* element will be used to store the version number of the *ACRU* model for which a ModelData file was created as a means of version control. Each ModelData file will need to keep a record of the ModelConfiguration file to be associated with it; this is another aspect of version control to ensure that the model version, ModelData file and ModelConfiguration file are all compatible. The *ModelConfigurationID* element in the initial schema has been renamed to *ModelConfigurationFile*. The new *ModelValidation* element in the revised schema will be used to hold information about whether the data in the ModelData file has been checked to be valid and when this was last checked. The purpose of the new *DefaultDataStore* element is to enable a default data store, for example a SPATSIM-HDSF database, to be specified.

A useful new element in the schema is the *ModelRuns* element which contains a list of zero or more *ModelRun* elements. A *ModelRun* element is used to store information about a particular model run so that it can be easily run again or so that a list of model runs can be configured and used in batch executions of the model. A *ModelRun* element stores an ID and description for the model run, the ordered scenario set to be used, the start and end dates of the simulation and optional start and end dates for the time series datasets to be used if different from the simulation start and end dates. Specifying the time series data start and end dates enables time series data preceding the simulation start date to be read into memory in situations where processes are influenced by model variable data values preceding the current simulation date.

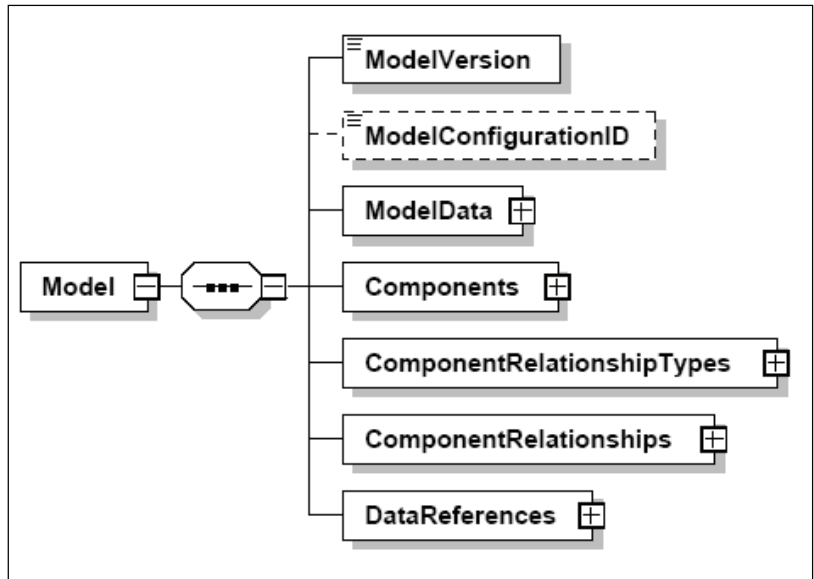


Figure 4.2 The initial design of the ModelData schema

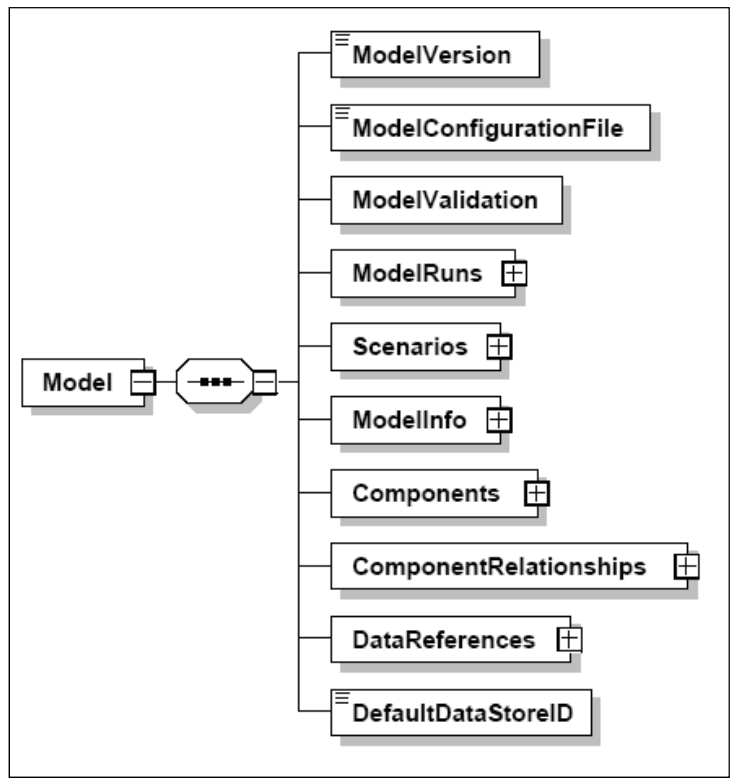


Figure 4.3 The revised version of the ModelData schema

In water resource planning it is often useful to be able to model two or more different scenarios. In the past this would have been done for the *ACRU* model in one of two ways, either changing the relevant data values in the model input file before running the model using the edited file, or copying the whole model input file and changing the relevant data values in the copy of the file. A drawback of the first method it that it can be difficult to keep

track of which data values were changed for which scenario and the second method results in duplication of data when only one or a few small changes are required for the scenario. Therefore a mechanism for setting up scenarios has been included in the data model and while this has added some complexity to the data model it is expected to be a useful feature. Each *Model* element will contain a *Scenarios* element which contains a list of one or more *Scenario* elements. A *Scenario* element is used to store information about a particular scenario it stores an ID and description for the scenario and an optional base scenario ID identifying the base scenario with which this scenario should be used. These *Scenario* elements are referenced by *Data*, *Component* and *Relationship* elements. The way in which scenarios have been designed to work is that typically a base scenario containing a full set of data would be configured by the user. The user would then configure additional scenarios which only contain the data values that change and these data values would override the data values in the base scenario. A scenario is only a base scenario if it does not itself have a base scenario. Scenarios which are not base scenarios can be superimposed over each other in the order specified in the scenario set specified in a *ModelRun* element, with the condition that they all have the same base scenario. Superimposed scenarios should typically not have overlapping parameter or variable values.

A model configuration may have parameters required to control model configuration and execution, these parameters are configured as a list of *Data* elements within the *ModelData* element. In the revised schema this *ModelData* element is contained within a new *ModelInfo* element, as shown in Figure 4.4, to be consistent with the *Component* element.

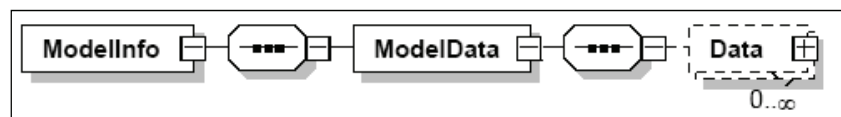


Figure 4.4 The new *ModelInfo* element containing a list of *Data* elements

The *Components* element contains a list of *Component* elements. The *Component* elements represent the physical components of the hydrological system being modelled (e.g. subcatchments, rivers, dams, vegetation, soil). Each *Component* element stores a unique ID for the component represented, a name for the component, the type of component being represented and a configuration component ID. The component type is a reference to a *ComponentType* element in the ModelConfiguration schema. The configuration component ID is a reference to a configuration *Component* element within the *ComponentConfiguration* element in the ModelConfiguration schema. A few changes have been made to the *Components* element as shown in Figure 4.5 and Figure 4.6 for the initial and revised

ModelData schemas respectively. The *ConfigurationInfo* element shown in Figure 4.5 has been removed as the ModelConfiguration schema has been changed to contain only one component configuration.

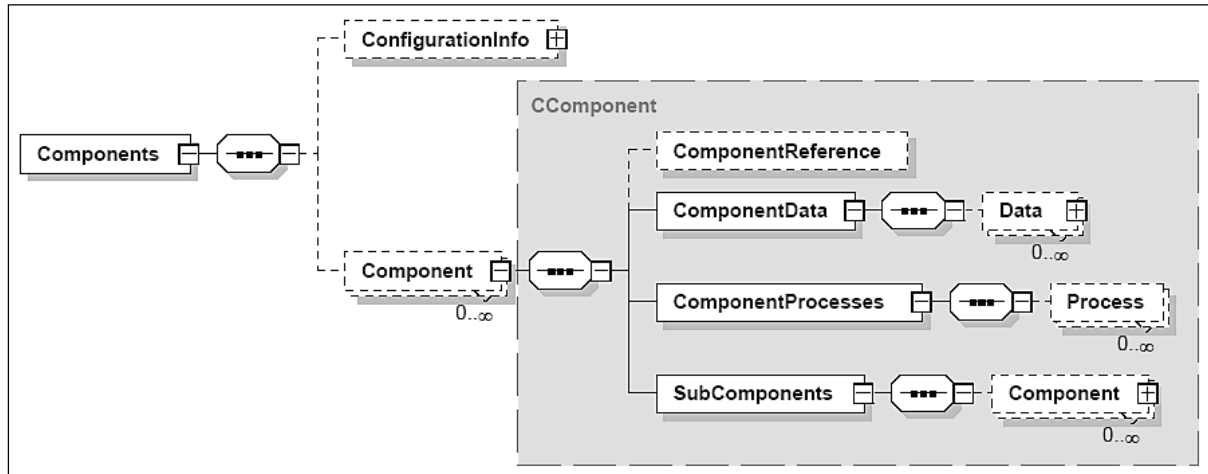


Figure 4.5 The *Components* element in the initial ModelData schema

In the revised ModelData schema shown in Figure 4.6 zero or more component *Scenario* elements may be included in a *Component* element. These component *Scenario* elements each contain two items of information, the first item is a scenario ID which is a reference to one of the model *Scenario* elements, and the second item specifies whether the *Component* element is active or inactive for the scenario. By default a *Component* element will be active unless it has a *Scenario* element that specifies that it is inactive for the specified scenario. Component scenarios enable certain components to be excluded from a simulation, for example when running simulations to determine the effect of a new dam in a subcatchment. Component scenarios should be used with caution and would typically be an option available to advanced users only. Setting *Component* element scenarios would require corresponding *Relationship* element scenarios to be set.

A new *SpatialRef* element has been added to the *Component* element to enable a spatial reference to be stored for a *Component* element, where this spatial reference may refer to a feature in an ESRI shapefile or geodatabase for example. These component *SpatialRef* elements each contain two items of information, the first item is a data reference ID which is a reference to one of the model *DataRef* elements which for example may store information for an ESRI shapefile, and the second item stores an ID that will be used to identify a particular spatial entity within the data reference, for example a particular feature in an ESRI shapefile.

The *SubComponents* element within a *Component* element contains a list of *Component* elements which are subcomponents of the parent *Component* element, for example a dam within a subcatchment. This structure allows for a nested hierarchy of parent and child *Component* elements.

The *ComponentProcesses* element within a *Component* element contains a list of *Process* elements belonging to the parent *Component* element. It is intended that a *Process* element will contain information about an algorithm for a hydrological process to be run for the parent *Component* element.

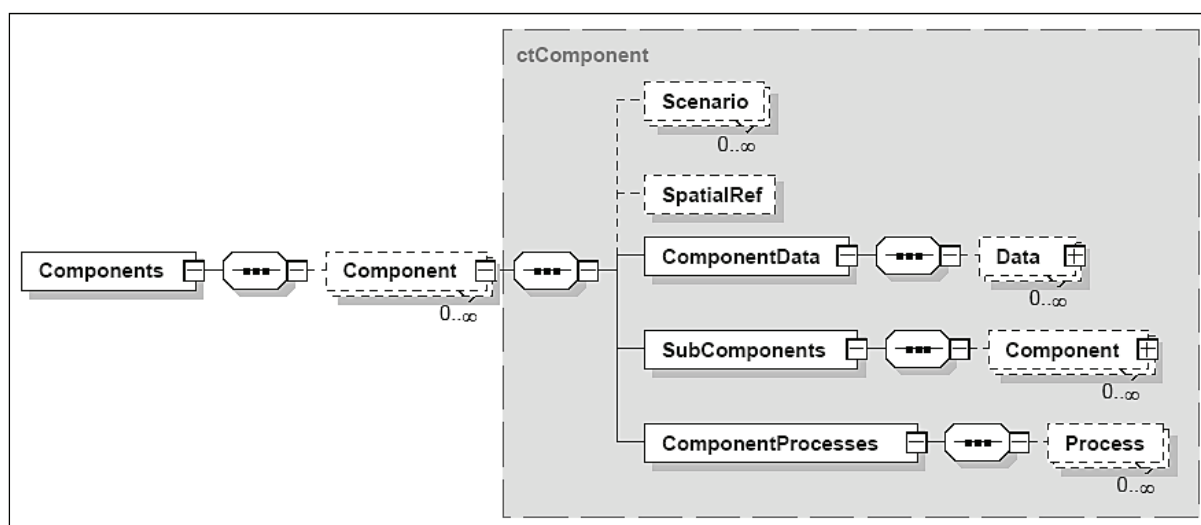


Figure 4.6 The *Components* element in the revised ModelData schema

The physical components making up a hydrological system to be modelled, for example hydrological response units (HRUs), dams and subcatchments, do not exist in isolation, they are related to each other in a one or more ways. For example an HRU may be related to a dam in that it is upstream of the dam. The *ComponentRelationshipTypes* element has been moved to the ModelConfiguration schema. Each *Model* element will contain a *ComponentRelationships* element which contains a list of zero or more *Relationship* elements. A *Relationship* element is used to store information about a relationship between two *Component* elements; it stores the relationship type, for example streamflow, and the IDs of the two *Component* elements which are the subjects of the relationship being stored. In the revised ModelData schema shown in Figure 4.7 zero or more relationship *Scenario* elements may be included in a *Relationship* element. These relationship *Scenario* elements each contain two items of information, the first item is a scenario ID which is a reference to one of the model *Scenario* elements, and the second item specifies whether the *Relationship* element is active or inactive for the scenario. By default a *Relationship* element will be



active unless it has a *Scenario* element that specifies that it is inactive for the specified scenario. Relationship scenarios enable certain relationships to be excluded from a simulation. Relationship scenarios would only be required when component scenarios are set and should be used with caution.

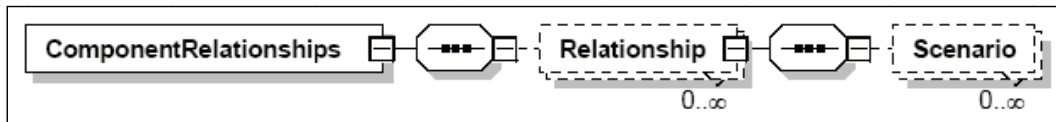


Figure 4.7 The *Relationships* element in the revised ModelData schema

A common problem when setting up a hydrological model is ensuring that the model input data is in the specific data format required by the model. Translation of data to a different data format is time consuming and can lead to errors in the translated data and therefore incorrect inputs to a model. This problem is further evident when using two different models in an integrated water resource assessment context to model two separate aspects of the water resource system as often output from one model needs to be used as input to the second model. In addition it is not efficient to store large time series data sets in XML such as in a ModelData file. It would also be advantageous to be able to specify that model output data be saved to a specific format. These considerations lead to the concept of data references in the ModelData schema. Each *Model* element will contain a *DataReferences* element which contains a list of zero or more *DataRef* elements as shown in Figure 4.8. A *DataRef* element is used to store information about a particular data reference; it stores an ID for the data reference and a data reference type which identifies the format of the referenced data store, for example *ACRU\_SingleFormat* or *SPATSIM-HDSF*. A *DataRef* element may also contain zero or more *Param* elements, where each *Param* element stores a parameter name and value pair, for example "FILENAME" as the parameter name and "C:\Myfolder\MyFile.txt" as the parameter value. The information stored in the *Param* elements would be used by third party software utilities to locate the referenced data store and open it for reading and writing. These *DataRef* elements are referenced by *Data* elements and *SpatialRef* elements belonging to *Component* elements.



Figure 4.8 The *DataRef* element in the revised ModelData schema

The design of the *Data* element has undergone substantial changes in conjunction with changes to the design of the *DataDef* element in the ModelConfiguration schema. The *Data* elements used in the *ModelInfo* and *Component* elements are identical. Schema diagrams of the initial and revised versions of the *Data* element are shown in Figure 4.9 and Figure 4.10 respectively. The combined design of the *Data* and *DataDef* elements has attempted to provide enough flexibility so that a particular model parameter may have a constant value or a value that changes dynamically during the simulation and to be able to store state data so that a model can be hotstarted.

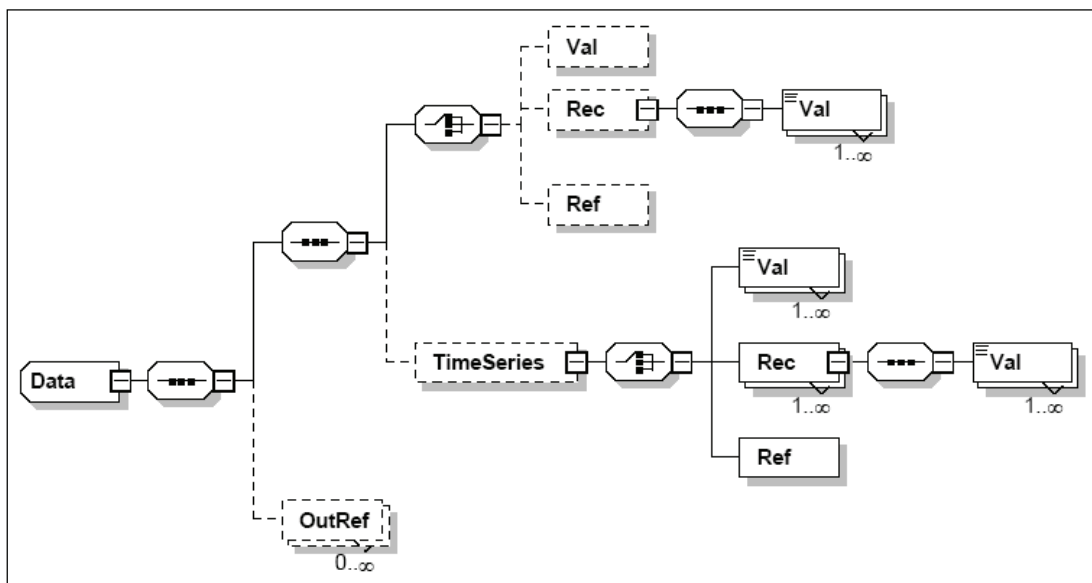


Figure 4.9 The initial design of the *Data* element

In the revised ModelData schema shown in Figure 4.10 each *Data* element will contain one or more *Scenario* elements. These data *Scenario* elements store a scenario ID which is a reference to one of the model *Scenario* elements. Each data *Scenario* element will contain a *Val*, *Rec* or *TimeSeries* element in which the data values for the scenario are stored. A *Val* element stores one data value or a reference to one data value. A *Rec* element stores a table of data values or a reference to a table of data values. A *Rec* element may contain a table (record) of data values in the form of either a 1-D array of values, a 2-D array of values or a dictionary of key-value pairs. A *TimeSeries* element stores a time series of *Val* or *Rec* elements or a reference to a time series. A *TimeSeries* element also contains two attributes, one stating the type of time series, such as daily, monthly or breakpoint, and the other stating the format of the timestamp used for the time series data/time values, for example “yyyy/MM/dd”.

In the initial design of the *Data* element if the data element stored a reference to an external data source then the information about the data reference was stored in the *Ref* element. This data reference information is now stored in the *Val*, *Rec* and *TimeSeries* elements so that when reading from the external data source it is known whether one data value, a table of data values or a time series is expected. The *Val*, *Rec* and *TimeSeries* elements may store either actual data values or a reference to data values stored externally but not both. References to data values stored externally require two items of information to be stored, the first item is the ID of the *DataRef* element that stores information about the data store itself, and the second item is an ID that identifies the location of the data value or values within the data store.

A *Scenario-Val* element stores only a data value. The *Rec-Val* and the *TimeSeries-Val* elements store a data value and also a key used to identify the data value within the set. A *TimeSeries-Val* element stores a data value, a time stamp and an optional data quality flag. A *TimeSeries-Rec* element is similar to a *Scenario-Rec* element but in addition stores a time stamp and an optional data quality flag.

Each data *Scenario* element may also contain zero or more *OutRef* elements. The purpose of *OutRef* elements is to store information about where model output for the scenario is to be stored. This information includes the ID of the *DataRef* element that stores information about the data store itself, and the location of where the data value is to be stored within the data store. The *OutRef* element also stores information regarding whether model output should replace or be appended to existing data values in the data store.

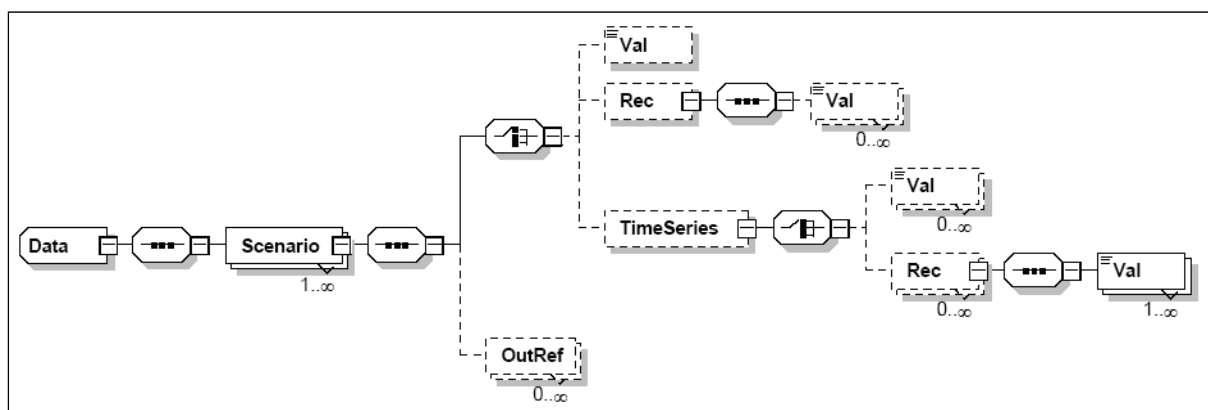


Figure 4.10 The revised version of the *Data* element.

The design of the ModelData schema has been revised to the point where it is expected to be stable and no substantial changes to the design are expected. The ModelData schema must be used in conjunction with the ModelConfiguration schema

#### 4.2.2 ModelConfiguration schema

Schema diagrams of the main elements of the initial and revised versions of the ModelConfiguration schema are shown in Figure 4.11 and Figure 4.12 respectively to help illustrate the design changes that have been made to this schema. Implementations of the ModelConfiguration schema will be referred to as a “ModelConfiguration file”, therefore a ModelConfiguration file is a populated XML file that obeys the ModelConfiguration schema. Each of the main schema elements will be briefly described to explain its purpose and important changes to the schema will be described in more detail. The primary purpose of a ModelData file is to store model input data values and model settings. The primary purpose of a ModelConfiguration file is to store information describing permitted component configurations and relationships and to store metadata about model parameters and variables. A large proportion of the information stored in a ModelConfiguration file is not required by the model but is required by software utilities used to display, edit and analyse data values stored in a ModelData file.

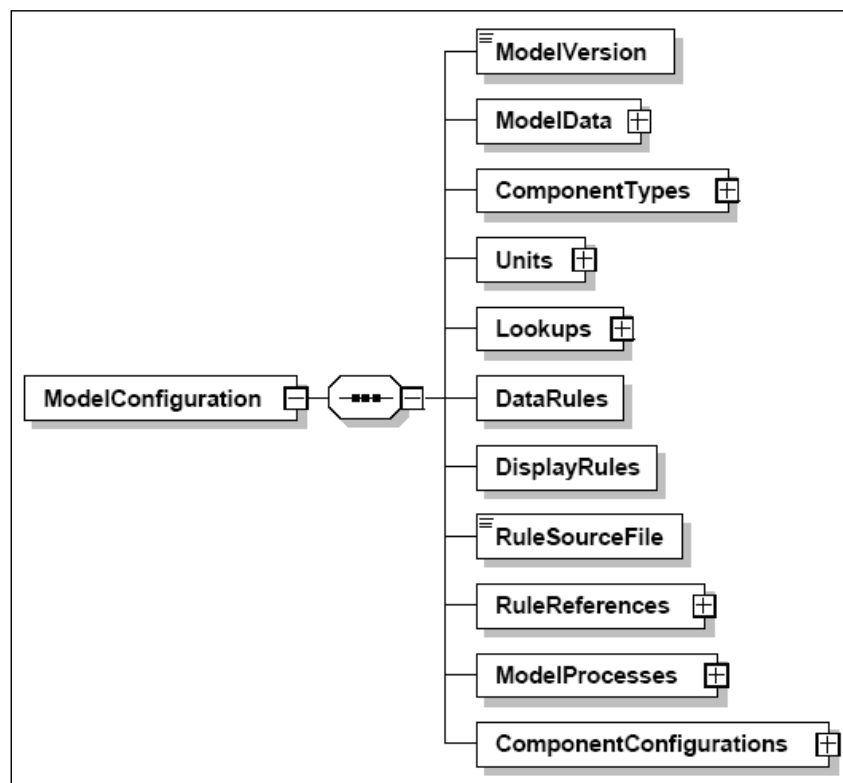


Figure 4.11 The initial design of the ModelConfiguration schema

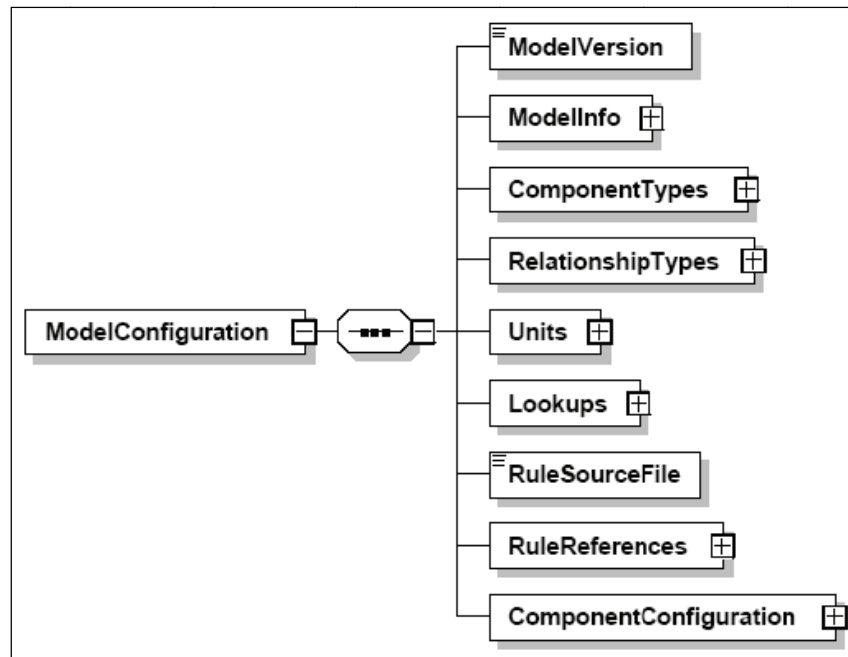


Figure 4.12 The revised version of the ModelConfiguration schema

The *ModelConfiguration* element has been extended to store a name and description for the model configuration it represents. The *ModelVersion* element will be used to store the version number of the *ACRU* model for which a *ModelConfiguration* file was created as a means of version control.

The *ModelInfo* element in the *ModelConfiguration* schema is related to the *ModelInfo* element in the *ModelData* schema and for similar reasons the *ModelInfo* element was created in the revised schema as a container for the *ModelData* element as shown in Figure 4.13. The *DataDefinitions* element contains a list of zero or more *DataDef* elements where each *DataDef* element contains metadata information about a general model parameter. The *DataGroups* element has been added to be consistent with the *ComponentType* element. Data groups provide a means of grouping model parameters or variables for display purposes in software utilities.

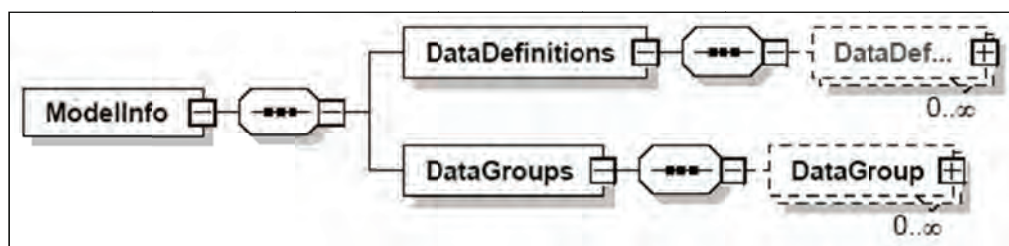


Figure 4.13 The new *ModelInfo* element in the ModelConfiguration schema

The *ComponentTypes* element contains a list of *ComponentType* elements. The *ComponentType* elements represent the different types of physical components making up the hydrological system being modelled (e.g. subcatchments, rivers, dams, vegetation or soil horizons). Each *ComponentType* element stores a unique ID for the component type represented, a name for the component type, the name of the model software class that is to be associated with the component type, and help text and description information for the component type. A few changes have been made to the *ComponentType* element as shown in Figure 4.14 and Figure 4.15 for the initial and revised ModelConfiguration schemas respectively. The *Identifiers*, *ComponentProcesses* and *SubComponentTypes* elements shown in Figure 4.14 have been removed as these are no longer required. The *ComponentType* element literally defines a type of component by means of the parameters and variables describing its characteristics. Component configuration is dealt with in the *ComponentConfiguration* element. For example, an in-channel dam and an off-channel dam may both be represented by the same “dam” component type but they will be configured differently in terms of streamflows.

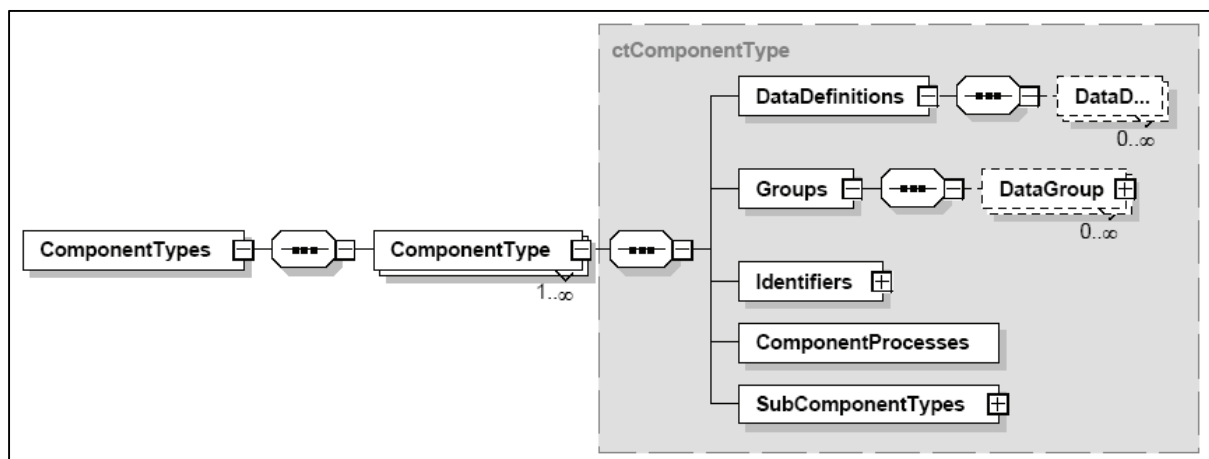


Figure 4.14 The *ComponentTypes* element in the initial ModelConfiguration schema

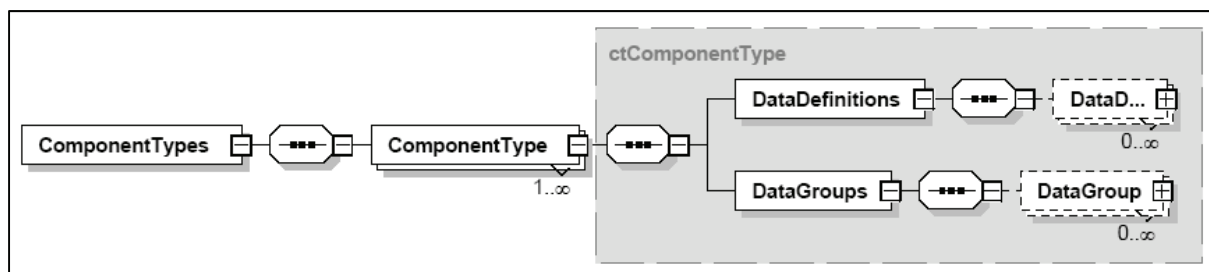


Figure 4.15 The *ComponentTypes* element in the revised ModelConfiguration schema

As explained in Section 4.2.1 the *ComponentRelationshipTypes* element shown in Figure 4.2 for the initial ModelData schema has been moved to the ModelConfiguration schema as shown in Figure 4.12 and has been renamed to *RelationshipTypes*. The *RelationshipTypes* element has been moved to the ModelConfiguration schema to prevent duplication between ModelData files but also to standardise the relationship types so that ModelData files do not each specify different relationship types that may not be recognised by the model. A *RelationshipType* element stores a unique ID for the relationship type, and a context and an inverse context for the relationship type. For example, a relationship type with the ID of “Streamflow” would have a context of “Upstream” and an inverse context of “Downstream”, thus if river reach RiverA flows into river reach RiverB, then if RiverA was the subject then RiverB would be related to it in a downstream context.

The *Units* element has been simplified in the revised ModelConfiguration schema. The *Units* element consists of a list zero or more *Unit* elements each representing a unit of measure for example cubic metres. The *Unit* element stores a unique ID for the unit of measure and a name and description for the unit. Each unit of measure is assigned to a category, for example, cubic metres may be assigned to a “Volume” category. Further information related to the dimensions for the unit and conversion to SI units is also included.

The *Lookups* element is unchanged. The Lookups element contains a list of zero or more *Lookup* elements and is used to store lookup lists for parameters that have a finite number of permissible data values. Each *Lookup* element contains a list of *LookupItem* elements each of which contains ID, name and description information about an individual lookup item.

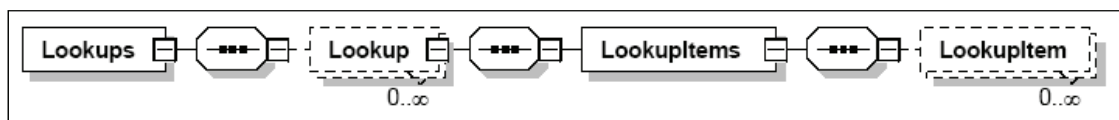


Figure 4.16 The *Lookups* element

The *RuleSourceFile* and *RuleReferences* elements are also unchanged. The *RuleSourceFile* element stores the name of the software class file containing the code for the data and display rules that are called for each model parameter and variable to determine whether the data values are valid and whether they should be displayed. The *RuleReferences* element stores references to software classes that are required by the file specified in the *RuleSourceFile* element to enable this file to be compiled on the fly.

The *DataRules* and *DisplayRules* elements shown in Figure 4.11 have been removed as data and display rules are now specified using elements within a *DataDef* element as in most cases the rules are specific to a particular data definition and by placing them in the data definition it removes the need to search for the relevant rule for a particular data definition. The *ComponentProcesses* element shown in Figure 4.11 has been removed as it is no longer required.

The *ComponentType* element shown in Figure 4.15 contains data definitions that describe the characteristics of the component type. Each *ComponentType* element represents a particular component type in isolation of all other component types even the subcomponents of the component type. Some means was required to enable the configuration of these isolated component types to be described to represent the hydrological system being modelled. This configuration needed to include not only parent-child component containment relationships but also other relationships between components. The *ComponentConfiguration* element shown in Figure 4.17 and Figure 4.18 for the initial and revised ModelConfiguration schemas respectively performs this purpose. The initial ModelConfiguration schema allowed for more than one component configuration to be specified, but this has been discontinued in the revised ModelConfiguration schema as it was decided that it would be better to create a new ModelConfiguration file to specify a different component configuration. The *ComponentConfiguration* element contains three sub-elements *Components*, *PermissibleRelationships* and *AutomaticRelationships*.

The *Components* element contains information describing the parent-child component containment relationships. As may be expected it has a similar structure of *Component* and *SubComponents* elements as for the ModelData schema as shown in Figure 4.6. Each configuration *Component* element stores a unique ID for the configuration component, a name for the configuration component, the component type ID, the minimum and maximum permitted occurrences of the configuration component within its parent configuration component, and whether the configuration component is permitted to recur within itself.

It was recognised that some means was required to be able to specify what types of relationships could be specified between two configuration components. The *PermissibleRelationships* element contains a list of permissible *Relationship* elements each containing information describing a relationship that is permitted between two configuration components. For example, an in-channel dam may be permitted to have a streamflow relationship with an upstream river reach, but an off-channel dam would not be permitted to have such a relationship. A permissible *Relationship* element stores the relationship type,



for example streamflow, and the configuration *Component* element IDs of the two configuration *Component* elements which are the subjects of the relationship.

It was further recognised that in addition to specifying permissible relationships some means was required to be able to specify what relationships must exist for a particular configuration component to enable software utilities to automatically configure some of the relationships in a *ModelData* file thereby helping to reduce model configuration time. The *AutomaticRelationships* element contains a list of automatic *Relationship* elements each containing information describing the target configuration component, the relationship type and context, and the related configuration component.

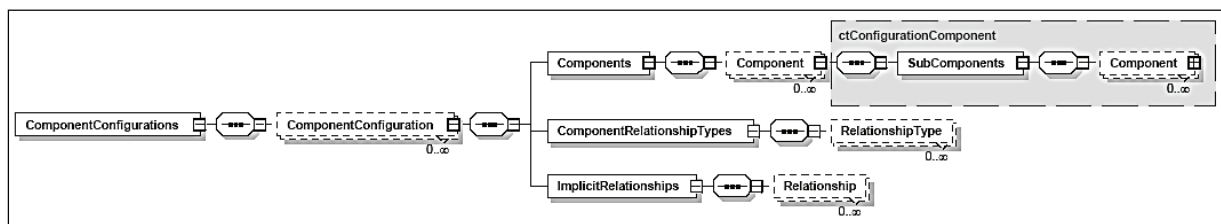


Figure 4.17 The *ComponentConfiguration* element in the initial *ModelConfiguration* schema.

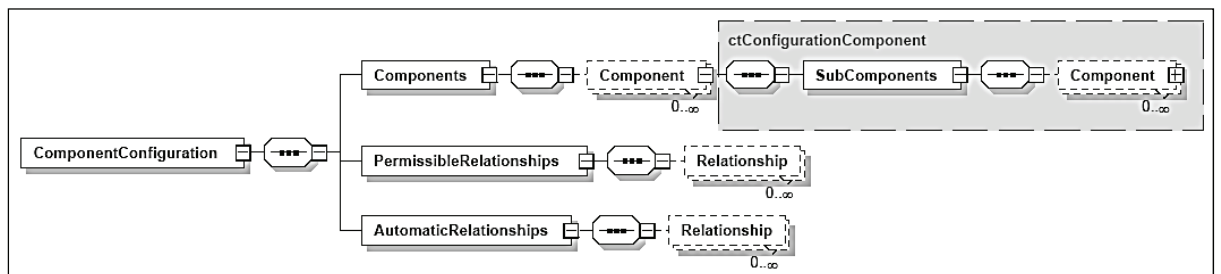


Figure 4.18 The *ComponentConfiguration* element in the revised *ModelConfiguration* schema.

The design of the *DataDef* element has undergone substantial changes in conjunction with changes to the design of the *Data* element in the *ModelData* schema. The *DataDef* elements used in the *ModelInfo* and *ComponentType* elements are identical. A schema diagram of the revised version of the *DataDef* element is shown in Figure 4.19. In the initial design the *DataDef* element did not contain any sub-elements. The *DataDef* element includes a long list of attribute information which is shown and described in

Table 4.1. The ID attribute of a *DataDef* element must contain an ID that is unique within the parent configuration *ModelInfo* or *ComponentType* element but need not be unique within

the ModelConfiguration file. The ID of a *Data* element in a ModelData file is identical to the ID of the corresponding *DataDef* element in the associated ModelConfiguration file to make the link between *Data* element and corresponding *DataDef* element. The *PType* attribute states whether the *DataDef* element represents input, output or state data, where state data can be regarded as both input and output data. The *PType*, *VType*, *SType* and *TType* attributes have been added to the *DataDef* element to describe the data values stored in a *Data* element as clearly as possible to make provision for all anticipated data structures that may need to be represented in the *ACRU* model and other similar models. The *VType* attribute stores the value type of the data values stored. The *SType* attribute stores the structure type of the data, an individual data value or a table of data values, where a constant would be one individual data value or one data table as opposed to and a time series of individual data values or a time series of data tables. The *TType* attributes states whether the data is always a constant, or always a time series, or whether the data may be dynamic. The *TType* attribute would be set to *Dynamic* if the data to be stored is in some instances modelled as a constant but in advanced modelling exercises the data may vary with time and a time series will be entered.

As stated previously the *DataRule* and *DisplayRule* elements are now situated within the *DataDef* element. A *DataDef* element may have more than one *DataRule* and *DisplayRule* element. A data rule is used to determine whether a model parameter or variable data value is valid or not. A *DataRule* element contains information about which software method in the rule source file is to be run and which other model parameters or variables may be required in determining the validity of the target parameter or variable. The display rules would be used by software utilities to determine whether a model parameter or variable should be displayed depending on user selected values for other model parameters or variables. A *DisplayRule* element contains information about which software method in the rule source file is to be run and which other model parameters or variables may be required in determining whether to display the target parameter or variable.

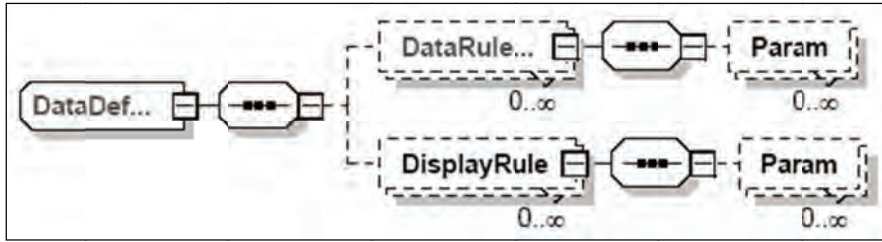


Figure 4.19 The *DataDef* element in the revised ModelConfiguration schema.

Table 4.1 Attributes of the *DataDef* element

Attribute	Use	Description
ID	required	A unique ID for the data definition
Name	required	A name for the data definition
Alias	required	An alternative name for the data definition
PType	required	Parameter type (Input, Output, State)
VType	required	Value type (String, Int16, Int32, Float, Double, DateTime)
SType	required	Structure type (Val, Rec)
TType	required	Time type (Constant, TimeSeries, Dynamic)
AType	optional	Aggregation type – only applies to time series data (None, Sum, Max, Min, Mean)
IType	optional	Interpolation type – only applies to time series data (Isolated, Step, StraightLine, Fourier, CubicSpline)
TSTypes	optional	Set of permitted time series types (Annual, Monthly, Daily, Hourly, Minute, Second, Breakpoint)
RType	optional	Record type – only applies to data with SType="Rec" (Array1D, Array2D, Dictionary)
RFormat	optional	Record format – only applies to SType="Rec" and RType="Array1D" or RType="Array2D"
UnitID	required	The ID of the unit of measure
DataClass	required	The associated software class
Decimals	optional	Default number of decimal places for numeric data
LookupID	optional	The ID of the lookup list to be used
ReadOnly	optional	Specifies if the data should be read-only
Description	required	Brief description of the data parameter or variable
HelpText	required	Help text for the data parameter or variable
MaxValue	optional	The maximum value for numeric data
MinValue	optional	The maximum value for numeric data
DefaultValue	optional	The default value to be used
ApplyDefault	required	Option to automatically set the default value

Data groups provide a means of grouping model parameters or variables for display purposes in software utilities. The *DataGroup* element shown in Figure 4.20 is unchanged from the initial design, it contains a unique ID for the data group, a name, a description, a parent data group if applicable and a list of *DataDef* element IDs which belong to the group.



Figure 4.20 The *DataGroup* element in the revised ModelConfiguration schema

The design of the ModelConfiguration schema has been revised to the point where it is expected to be stable and no substantial changes to the design are expected. The XmlModelFiles software library has been revised to implement the changes made to both the ModelData and ModelConfiguration schemas.

#### 4.2.3 XmlModelFiles libraries

The purpose of the XMLModelFiles software library is to facilitate the reading, writing and editing of ModelData and ModelConfiguration XML files. The XMLModelFiles software library was created in both the C-Sharp (C#) and Java programming languages. Substantial changes have been made to the library including:

- Consolidation and organisation of code within one software library;
- Changes and additions to the code to implement changes made to the ModelData and ModelConfiguration schemas;
- Creation of the *IDataContainer* interface so that *Data* elements in the *ModelInfo* and *Component* elements in the ModelData schema can be dealt with in the same way;
- Creation of the *IDataDefContainer* interface so that *DataDef* elements in the *ModelInfo* and *ComponentType* elements in the ModelConfiguration schema can be dealt with in the same way;
- Creation of new *DataRule* and *DisplayRule* classes;
- New code being written to enable data and display rules to be configured, run and accumulated up the data group and component hierarchy and for feedback to be provided regarding missing or invalid model parameters or variables; and
- New code to implement a new method of reading and writing XML files in the Java version of the library so that the *ACRU* model is able to write state data back to a ModelData file.

The XMLModelFiles library consists of three packages: XmlModelFiles.ModelData, XmlModelFiles.ModelConfiguration and XmlModelFiles.ModelRules. The elements within an XML file are often represented by matching code object, the process of saving the information within a code object to XML is referred to as serialisation and the reverse

operation is referred to as deserialization. A simplified Universal Modelling Language (UML) diagram of the main classes in the XmlModelFiles.ModelData package is shown in Figure 4.21. The *ModelDataDocument* class represents a ModelData file and contains code to serialize and deserialize between ModelData files and their matching classes. The *ModelElement*, *ModelRunElement*, *ScenarioElement*, *ModelInfoElement*, *ComponentElement*, *RelationshipElement* and *DataRefElement* classes represent the corresponding elements in the ModelData schema. The *IDataContainer* interface has been created so that the *ModelInfoElement* and *ComponentElement* classes can be referred to in a generic manner when working with the *DataElement* class. The *DataRule* and *DisplayRule* classes were created to contain code used to configured and run data and display rules based on the information stored in the *DataDef* elements of a ModelConfiguration file.

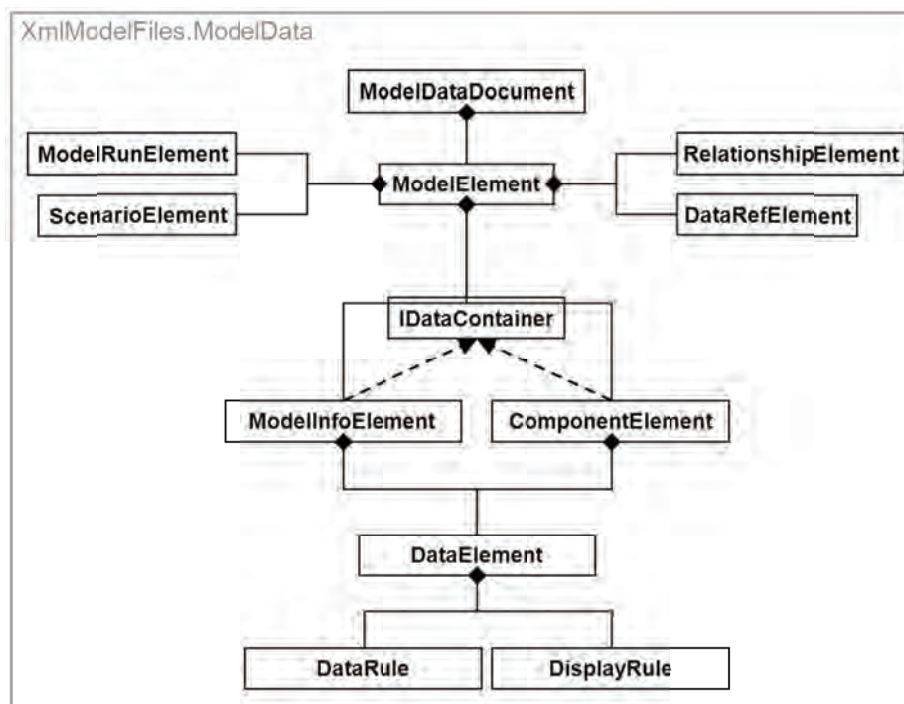


Figure 4.21 Simplified UML diagram of the XmlModelFiles.ModelData package

A simplified Universal Modelling Language (UML) diagram of the main classes in the XmlModelFiles.ModelConfiguration package is shown in Figure 4.22. The *ModelConfigurationDocument* class represents a ModelConfiguration file and contains code to serialize and deserialize between ModelConfiguration files and their matching classes. The *ModelConfigurationElement*, *ConfigurationModelInfoElement*, *ComponentTypeElement*, *DataDefElement*, *ComponentConfigurationElement*, *ConfigurationComponentElement*, *PermissibleRelationshipsElement* and *AutomaticRelationshipsElement* classes represent the

corresponding elements in the ModelConfiguration schema. The *IDataDefContainer* interface has been created so that the *ConfigurationModelInfoElement* and *ComponentTypeElement* classes can be referred to in a generic manner when working with the *DataDefElement* class.

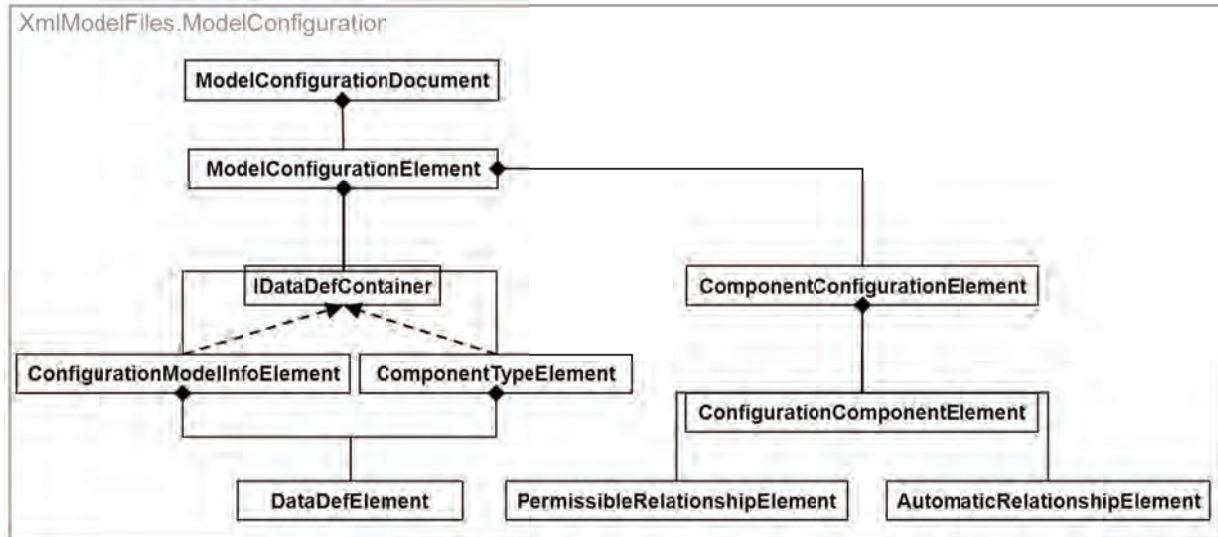


Figure 4.22 Simplified UML diagram of the XmlModelFiles.ModelConfiguration package

The XmlModelFiles.ModelRules package contains several low level classes used to compile and run the software class file containing the code for the data and display rules and will not be documented here. These classes were consolidated into this library as they had previously formed part of the ConfigurationEditor code set.

It should be noted that changes and additions to the code for the ConfigurationEditor software utility were required to implement changes made to the ModelData and ModelConfiguration schemas, however, this work was done within WRC Project K5/1870 within which the coding of the main functionality of the ConfigurationEditor was completed. An important aspect of the work completed in WRC Project K5/1870 was the creation of a set of forms and code classes to enable references to data stored in external data repositories to be configured and read data from and write data to these repositories. The data repositories currently included are: SPATSIM-HDSF databases implemented in Microsoft Access, *ACRU* Single format, *ACRU* Composite format and *ACRU* CompositeY2K format. These improved and more flexible linkages to data, especially time series data, stored in external data files are expected to be invaluable for including forecast data for operational modelling and for the integrated modelling.

The design changes to the *ModelData* and *ModelConfiguration* schemas and corresponding changes to the *XmlModelFiles* library formed the first component of this further development the second component involved the implementation of these new features into the *ACRU* model.

### 4.3 Model Development

As stated in the introduction to Chapter 4 the aim was to make necessary changes to the *ACRU* model and the design of its associated model input files to ensure that the *ACRU* model is suited for use in both water resources planning and operations modelling and is capable of representing real world complexity. The changes to the design of the model input files described in Section 4.2 of this document laid the necessary foundation for important new functionality to be added to the *ACRU* model. However, the changes to the code of the *ACRU* model are more difficult to document than the design of the model input files so this section will only contain brief descriptions of the changes made to the model.

Before the further development of the *ACRU* model completed in this project can be described it is necessary to explain a few *ACRU* related terms. The object oriented structure of the model includes three main types of objects: *Component* objects which represent the physical components of the hydrological system being modelled (e.g. subcatchments, rivers, dams, vegetation, soil), *Process* objects which represent the hydrological processes through one or more algorithms (e.g. evapotranspiration, runoff, infiltration) and *Data* objects which contain the parameter or variable data values that describe the *Component* objects. Additional object types include *Model* which serves as the main object container for a model run and various model input, output and configuration objects.

The changes to the design of the model input files reported in Section 4.2 laid the necessary foundation for important new functionality to be added to the *ACRU* model. The main classes of the *ACRUXml* version of the *ACRU* model are shown in Figure 4.23. A *MModel* object contains one or more primary *CComponent* objects which may in turn be composed of one or more *CComponent* objects, where *CComponent* objects represent the physical entities of the hydrological system being modelled. Each *MModel* class is associated with an *AModelInput* class, an *AModelCreator* class and an *AModelOutput* class to enable model input to be read, model configuration to be performed and model output to be written out. The *MModel* and the *CComponent* class both implement the *IDataOwner* interface which means that *MModel* and the *CComponent* objects may contain a list of parameters and variables represented by *DData* objects and which describe the attributes of the *MModel* and





simulation start date, the simulation end date and if necessary the data start and end date. It is expected that the concept of model runs together with the concept of scenarios will enable a range of scenarios to be run using batch executions of the model.

#### **4.3.1 Component types and component configurations**

The hydrological systems that models such as *ACRU* are created to represent can be viewed as being composed of a collection of distinct components such as: HRUs, dams, river reaches, vegetation and soil horizons. The *ACRU* 3.00 version of the model used a simple, and in most cases adequate, view of the components making up the hydrological system being modelled. The main spatial building blocks were subcatchments. Each subcatchment consisted of a land area plus an optional dam, an optional irrigated area, an optional adjunct impervious area and an optional disjunct impervious area. Subcatchments were selected such that the land area represented a region on which the climate, soil and land use of the land area were assumed to be homogeneous. In an effort to represent the real world in which climate, soils and land use can vary within a particular hydrological catchment, subcatchments were frequently used to represent HRUs rather than regions defined by a watershed. This component configuration structure was to a large degree hard coded into the model and its model input files.

The development of the object oriented *ACRU2000* version of the model was an important step towards enabling more flexibility in setting up the configuration of the spatial components representing the hydrological system. It is important to note that there is a distinction between providing more flexibility in the configuration of spatial components and providing complete flexibility for a user to in effect build their own model using components and processes as building blocks. Complete flexibility is difficult to achieve due to the complexity of the process algorithms and the fact that many of the process algorithms were developed based on certain assumptions, for example in the *ACRU* model the soil water process algorithms are based on concept of the soil being comprised of an A-horizon and a B-horizon. Complete flexibility would not only be difficult to achieve but also dangerous as the majority of model user's would not be likely to have a complete understanding of all the process algorithms and the feedbacks that may occur.

The design of the *ModelData* and *ModelConfiguration* schemas based on object orient concepts was a second important step towards more flexible configuration of spatial components in the *ACRU* model and therefore better representation of real world complexity. In addition the *SpatialRef* element added to the *Data* element in the *ModelData* schema will

enable spatial components to be given a spatial reference which will facilitate the development of GIS tools for setting up component configurations. For the *ACRUXml* version of the *ACRU* model a *ModelConfiguration* file has been created to include a component configuration that will enable more flexible configuration of spatial components without compromising the well-established process algorithms developed and tested for the model over many years. The new data structure including components and relationships has also enabled easier and more flexible configuration of flow networks within the model. The *ACRU* *ModelConfiguration* file is named “*AcrucConfiguration.xml*”. In the *ACRUXml* version there are four main aspects of the component configuration that need to be mentioned:

- As in previous versions of the model subcatchments are the main building blocks of the system being modelled, though a return has been made to the concept that each subcatchment should have a watershed as its’ boundary;
- The concept of HRUs has been introduced where each subcatchment may contain one or more HRUs representing the homogeneous land areas within it;
- The concept of catchments has been introduced to enable subcatchments to be grouped together within catchments; and
- The concept of nodes has been introduced to aid in flow routing.

In *ACRUXML* there are three main types of objects that are referred to: *Component*, *Data* and *Process*, where:

- *Component* objects represent are the real world physical entities (e.g. catchment, dam, irrigated area, soil, vegetation) that make up the hydrological system being modelled.
- *Data* objects represent the attributes of each *Component* object (e.g. catchment area, dam volume).
- *Process* objects represent the real world hydrological processes that act on the real world physical entities represented by *Component* objects (e.g. evaporation, runoff, infiltration).

The spatial *Component* object types represented in *ACRUXml* are briefly described below to explain how *ACRU* visualises the real world hydrological system:

- A *Catchment* is a spatial container for other *Catchments* and *Subcatchments*.
- A *Subcatchment* is a spatial container for other spatial entities including: *HRU*, *IrrigatedArea*, *AdjunctImperviousArea*, *DisjunctImperviousArea*, *Vlei*, *RiparianZone*, *River*, *RiverInflowNode*, *Dam*, *DamInflowNode* and *SubcatchmentNode*. A *Subcatchment* may not contain other *Subcatchments*, it is the smallest spatial

container representing a surface flow watershed. It is a container for entities representing segments of land and the flow network.

- A *HRU* (hydrological response unit) is a spatial segment of land for which the soil and land cover are assumed to be homogeneous. A *HRU* is used to represent a spatial segment of land that does not require specialised processes as is the case for *IrrigatedArea*, *AdjunctImperviousArea*, *DisjunctImperviousArea*, *Vlei* and *RiparianZone*.
- An *IrrigatedArea* is a spatial segment of land on which irrigation may be applied (currently an *IrrigatedArea* is a subcomponent of a *HRUs* because the area of the *IrrigatedArea* can vary monthly and the *ACRU* model needs to adjust the net area of the *HRU* accordingly).
- An *AdjunctImperviousArea* is a spatial segment of land that has an impervious land cover and is adjacent to and flows directly into the flow network.
- A *DisjunctImperviousArea* is a spatial segment of land that has an impervious land cover and is adjacent to a *HRU*, *Vlei* or *RiparianZone* and flows directly onto this *HRU*, *Vlei* or *RiparianZone*.
- A *Vlei* is a spatial segment of land adjacent to part of the flow network, it receives excess flow from the flow network and may be modelled together with a dam.
- A *RiparianZone* is a spatial segment of land adjacent to part of the flow network, it receives baseflow from upslope *HRUs* and also excess flow from the flow network.
- A *River* is a spatial river reach and forms part of the flow network.
- A *RiverInflowNode* is a spatial node that flows into a *River*, a *RiverInflowNode* must exist for each *River*.
- A *Dam* is a spatial dam reach and forms part of the flow network.
- A *DamInflowNode* is a spatial node that flows into a *Dam*, a *DamInflowNode* must exist for each *Dam*.

The *ACRU* ModelConfiguration file has been populated with the necessary information for all main features of the *ACRU* model. Sample ModelData files have been created for *ACRU* and the *ACRU* model has been successfully run using these files.

#### 4.3.2 Components

In parallel with changes to the Component structure used in the *ACRUXml* version of the ModelConfiguration file several changes were made to the Component classes in the *ACRU*

model. The main spatial Component classes are shown in Figure 4.24 and the main spatial subcomponents of these Components are shown in Figure 4.25.

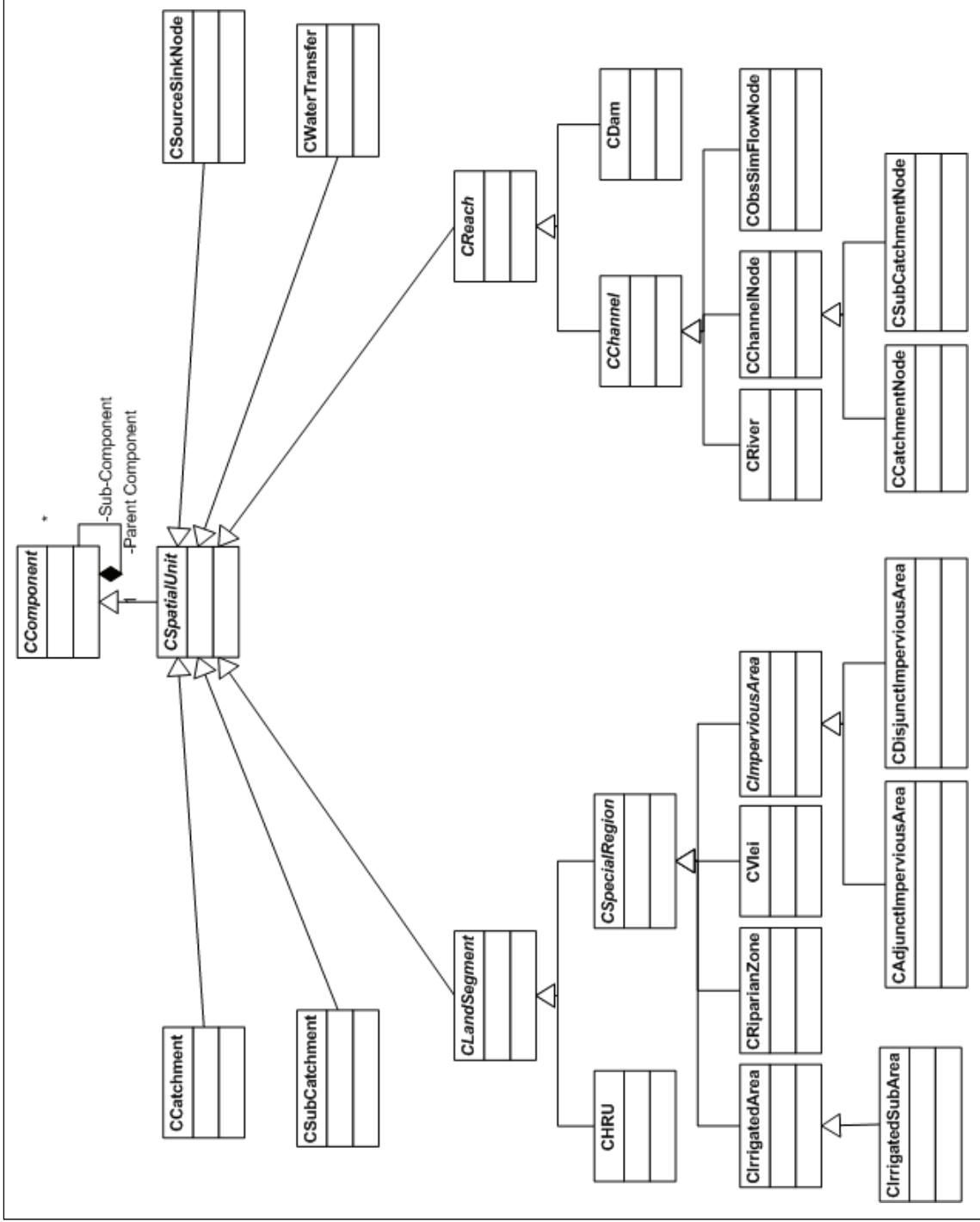


Figure 4.24 A UML Class Diagram showing the new main Component classes in the ACRUXm version of the model

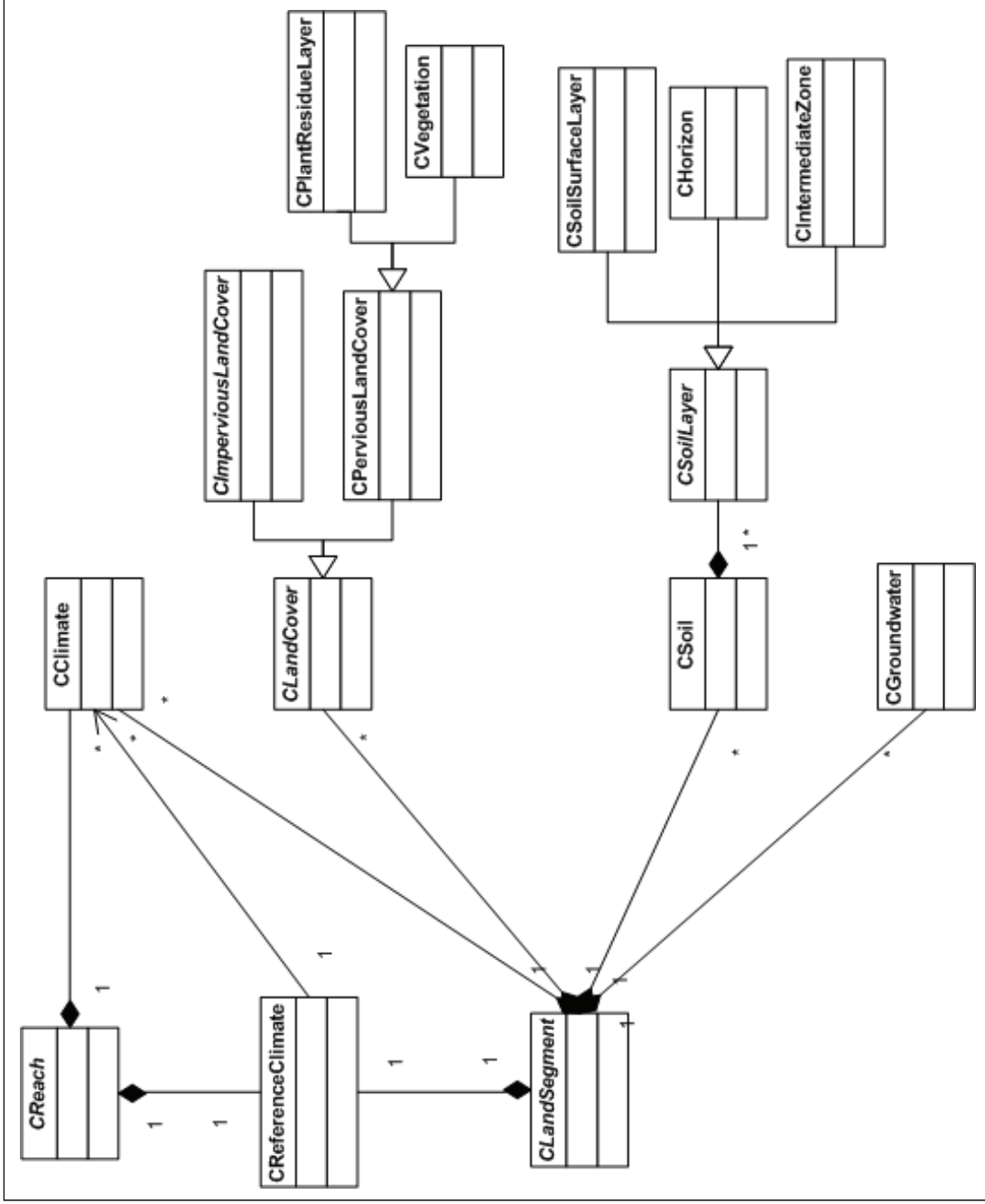


Figure 4.25 A UML Class Diagram showing the main subcomponents of the main spatial Component classes in the ACRUXm/ version of the model

In the *ACRU2000* version of the model the *CLandSegment* class was used to represent the *ACRU 3.00* version view of a subcatchment. The *CLandSegment* class acted both as a container for other optional spatial Components such as a river, a dam, an irrigated area and impervious areas, and at the same time also represented the remaining land portion of the subcatchment. This led to many complications and one of the goals in the *ACRUXml* version of the model was to change the way in which the spatial components of the hydrological system were represented and in particular to remove cases where one instance of *CLandSegment* was contained within another. It should be noted that changes to the Component structure would only result in minor coding changes to the Process classes and no change to the simulation results. With this goal in mind the following changes were made to the way in which the spatial Components were structured:

- The *CLandSegment* class was made abstract and now generically represents a portion of land which may be subclassed to represent specialised land units.
- The new *CHRU* class was introduced as a subclass of *CLandSegment* to represent a portion of land termed as a “hydrological response unit” (HRU), where a HRU is understood to be any non-specialised portion of land where, for modelling purposes, the soil and land cover are assumed to be homogeneous.
- The *CIrrigatedArea* class was modified so that it is no longer a subcomponent of *CLandSegment*, but rather a subcomponent of *CSubCatchment* as is the case for all other subclasses of *CLandSegment* which is now abstract. This required that the area of an irrigated area be constant for the duration of a simulation. The concept of the area of irrigation varying from month to month was a previous feature in *ACRU* that was seldom if ever used and can now be modelled by placing more than one irrigated area within a subcatchment.
- The *CIrrigationSystem* class was changed to extend the *CWaterTransfer* class so that the irrigation water supply path does not need to contain a *CWaterTransfer* class to simplify setting up water sources for irrigated areas.
- The new *CSubCatchment* class was introduced as a spatial container for other spatial entities including the flow network and portions of land such as HRU, irrigated areas and impervious areas. An instance of the *CSubCatchment* class should not contain other instances of the *CSubCatchment* class as it is intended to be the smallest spatial container representing a surface flow watershed. An instance of *CSubCatchment* can now contain more than one instance of each type of spatial unit.
- The new *CSubCatchmentNode* class was introduced to represent a node in the flow network at which flows out of a subcatchment can be evaluated.

- The *CCatchment* class was changed to extend *CSpatialUnit* instead of *CComponent* as this makes sense even though it is currently just serves as a container with no processes. The *CCatchment* class was modified so that an instance of *CCatchment* acts as a spatial container for instances of *CSubCatchment* and other instances of *CCatchment* forming sub-areas within the catchment represented. The main role of the *CCatchment* class is to act as a container enabling instances of the *CSubCatchment* to be grouped in a hierarchical manner. The code used to calculate the catchment area has been removed as this is now a model input and is used as a check on the total area of the spatial subcomponents within the catchment.
- The new *CCatchmentNode* class was introduced to represent a node in the flow network at which flows out of a catchment can be evaluated.
- In *CSpatialUnit* removed the *catchment* instance variable and associated methods as this is now included as part of the component-subcomponent hierarchy
- In *CSpatialUnit* removed the code that calculated the net area of the Component as with the new *CCatchment*, *CSubCatchment* and *CLandSegment* structure this complication has been removed.
- In *CReach* added new methods *getUpstreamSubCatchments* and *getUpstreamSpatialUnitsInSameSubCatchment* and *getTotalUpstreamArea*.
- The *CGully* and *CStream* classes were removed from the model as they were not being used.
- Removed the *CObservedFlowInput* and *CSimulatedFlowRemoved* classes as they were no longer required and removed all related code from the *CObsSimFlowNode* class.
- The *CSourceSinkNode* was added to the model to represent sources and sinks of water which do not form part of the flow network of a subcatchment, they are used to add water to and remove water from the hydrological system being modelled.
- The *CSpatialUnit* class and all its subclasses were moved from the *ACRU.Components* package to a new sub-package named *ACRU.Components.SpatialUnits*.

One of the Component related issues that the *ACRUXml* version of the ModelConfiguration file addressed was how to minimise the repetition of climate data stored through the introduction of the concept of reference climates and monitoring points. The context of the problem is that each spatial Component has its own set of climate related parameters and variables, but observed data may only be available at widespread monitoring points and would potentially be used by several spatial Components, but it would be wasteful to store



the same data values for each individual climate Component. This problem was partially addressed in the *ACRU2000* version of the model through the *CClimateStation* class, but a better solution was required and this has been achieved through the new *ReferenceClimate* and *MonitoringPoint* Component types in the *ACRUXml* version of the ModelConfiguration file and the new *CReferenceClimate* and *CMonitoringPoint* classes. The concept of a reference climate Component is that two or more spatial Components may refer to the same set of climate variables, for example the HRUs within a single subcatchment. The concept of a monitoring point Component is that the data for a single climate variable could be stored once and accessed by one or more reference climate Components. The *CClimateStation* class has been removed from the model. The *CClimateRelatedComponent* was also removed from the model as it complicated the Component inheritance structure and was not really necessary anymore. As part of the water tracking functionality within *ACRU* the *CClimate* class previously contained two subclasses *CRainfallStore* and *CEvaporationStore*. Although the *CRainfallStore* and *CEvaporationStore* classes made good sense from a strict coding conceptual point of view, they created additional complexity for model users so these classes were removed from the model and the code in the relevant Process classes was modified where necessary.

In the *ACRU2000* version of the model the *CVegetation* class was designed to have several subcomponent classes, namely *CLeafCanopy*, *CSeeds*, *CStems* and *CRoots* representing various components of the vegetation. While these subcomponents would make perfect sense for a detailed plant physiology model they served no real purpose in *ACRU* and the only one being used was the *CLeafCanopy* class. For the purpose of simplifying the model the *CLeafCanopy*, *CSeeds*, *CStems* and *CRoots* classes were removed from the model and the necessary changes were made to the model to move the parameters and variables associated with the *CLeafCanopy* class to the *CVegetation* class. The *CVegetation* class had also been subclassed to represent a large selection of vegetation types. These subclasses of the *CVegetation* class contained no specialised code and served no real purpose except to identify the vegetation type being modelled and prevent, for example, a maize yield model being run for a vegetation type that was not maize. All the subclasses of the *CVegetation* class have been removed from the model and if necessary the vegetation type can be specified using a parameter stored in a *DData* object.

In the *ACRU2000* version of the model the *CComponent* class and the *DData* class contained static variables called *COMPONENT\_REFERENCE* and *DATA\_REFERENCE* respectively. These static variables stored references of all the instances of *CComponent* and *DData* in a configuration of the *ACRU* to enable individual Component objects to be

accessed easily and to enable sets of similar Data objects to be queried and returned. In retrospect using static variables to do this was not the best solution from a computer programming point of view. The *COMPONENT\_REFERENCE* and *DATA\_REFERENCE* variables have both been removed from their respective classes. A Component reference and associated methods to query it has been implemented in a different manner as an instance variable in the *MModel* class. Recent code changes have removed the need for a Data reference. Minor changes were also made to the *CNode* class in which the two *getTypeOfNexts* methods were removed and replaced them with similar methods named *getNextsOfClass* and *getNextsOfSpecificClass*.

### 4.3.3 Internal data structure

It is typical when doing water resources planning that simulations will be run for long time periods so that sufficient data points are available for statistical analysis or to determine the long term influence of decisions made. A long simulation period may also be required to ensure that there is a suitable warm up period for the model so that state variables have time to stabilise. However, operation modelling requires short simulation periods to assist water managers in making short term decisions. Currently the *ACRU* model requires a suitable warm up period at the start of the time period being simulated so that certain state variables have time to stabilise. It would be an advantage if these state variables could be initialised with appropriate values at the start of the simulation to remove this need for a warm up period. In addition operational modelling requires that the *ACRU* model needs to be able to start from a known state, this is termed "hot-starting". The ability to hotstart is a feature that needed to be added to the *ACRU* model to enable it to be used for water resources operations modelling. The ability to hotstart *ACRU* is has been recognised as a critical feature for several years, not only by the developers of *ACRU* but also by specialist hydrological consultants. The use of forecast data, such as short term rainfall forecasts, is an important part of operational modelling and when used in this mode a model needs to be able to run simulations using forecast data and then roll back to observed or previously simulated states as updated forecasts come available. The reasons previously preventing hot-starting being implemented in *ACRU* were the lack of a suitable model input data structure for the storage of state variables, the internal data structure not meeting all the requirements for the storage and handling of state data, and that that the model included several internal state variables that could not be initialised from the model input files and were always set to zero at the start of each simulation.

Hydrological models such as *ACRU*, by means of parameters, variables and process algorithms, attempt to represent real hydrological systems which by nature are often complex. Not only are the process algorithms simple models of reality, but the model input parameters and variables are often also simplified. In many cases model input variables are provided as constant values, for example land use variables. In some cases providing land use variables as constants is reasonable, while in other cases especially for simulations over a long time period, it may be necessary to vary the values of the land use variables over time to reflect real land use changes. The *ACRU* 3.00 version of the *ACRU* model made provision for modelling certain model variables dynamically through the use of a dynamic input file. This feature was absent in the *ACRU2000* version of the model. One disadvantage of the dynamic input files in the *ACRU* 3.00 version was that working with these files was perceived by users to be difficult as these files were structured differently to the main model input files and data had to be entered into these files manually. The new model input data structures provided in the *ModelData* and *ModelConfiguration* schemas enable model parameters and variables in which data values can be entered as constant values or as time series of values. An advantage of the new model input data structure is that dynamic variables use the same data structure and model input file as other variables and corresponding changes need to be made to the model's internal data structure.

A UML Class Diagram showing the main Data classes used in the *ACRU2000* version of the model is shown in Figure 4.26. This data structure was strongly influenced by the main data types used in the *ACRU* 3.00 version of the model. Data classes, as opposed to simpler data types, were used to represent the model parameter and variables as they could do so much more than simply store data but also enable range checking, metadata storage and other functions. The main Data class was the abstract *DData* class which was extended by several abstract subclasses representing the various data types and structures required by the *ACRU* model. Only two main types of time related Data class were used, “daily” to represent time series of fixed interval daily data values and “monthly” to represent sets of 12 month-of-year data values which were generally disaggregated to create sets of day-of-year data values. The *DHydrographData* class was introduced to handle the sub-daily time series used when doing flow routing. The *DFluxRecord* class and its subclasses were created to meet the more complex data requirements of keeping track of the storage, fluxes, ownership and source or destination of matter or resources such as water, nitrogen or phosphorous within and between Components. The *DFluxRecord* class and its subclasses were in effect used to store the state of the different types of resources within a Component. The main classes shown in Figure 4.26 were in turn subclassed to create a data class for each input, output and state parameter or variable in the *ACRU* model. This was necessary as the class

names were used to identify the Data objects representing parameters and variables as at the time there was no other means to identify model parameters and variables, but this resulted in a large number of Data classes being created which in most cases did not include any additional functionality to that of their respective subclasses.

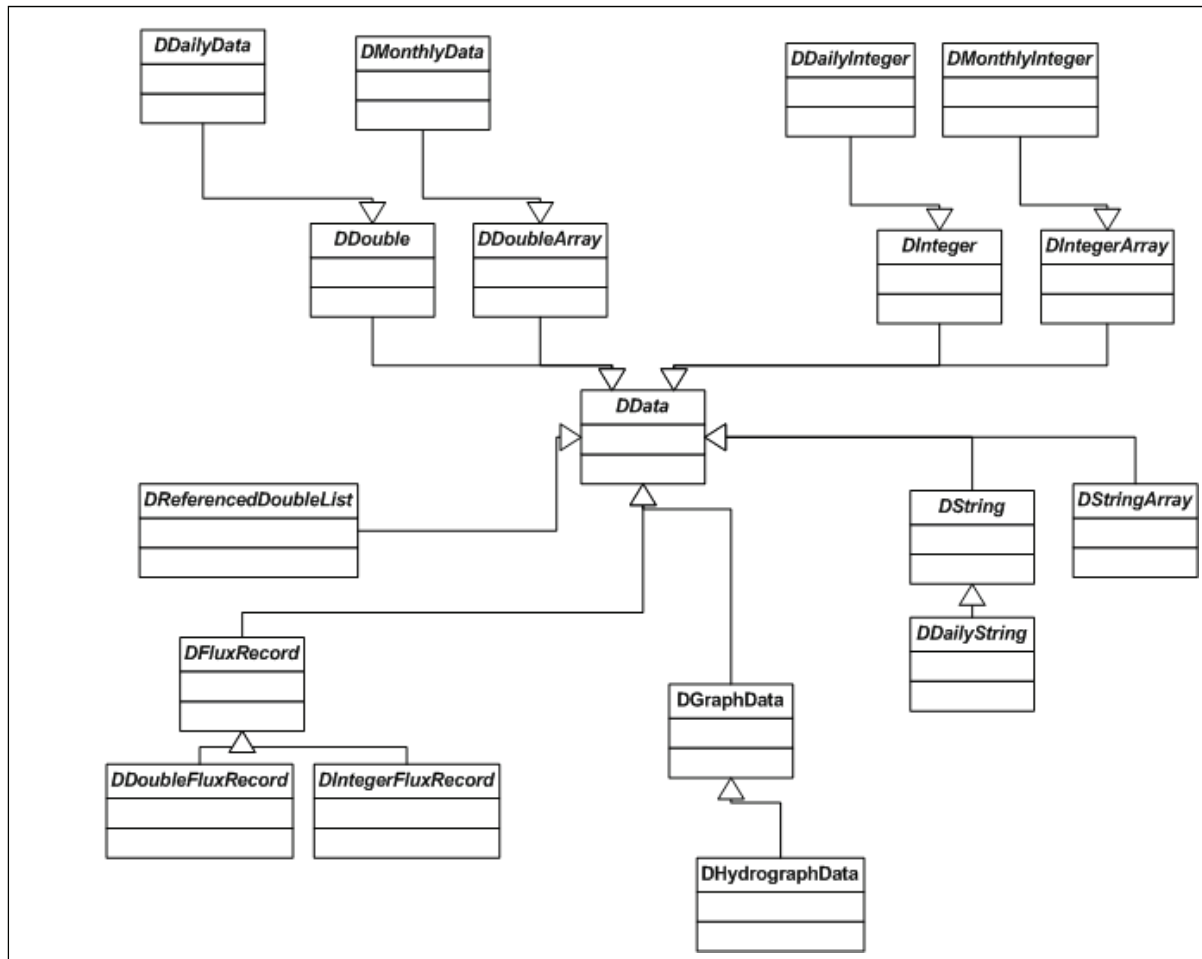


Figure 4.26 A UML Class Diagram showing the main Data classes used in the ACRU2000 version of the model

For the reasons discussed above it was necessary to completely restructure the internal data structure used by ACRU. The objectives of this restructuring were to (i) implement a better means of representing time series data to facilitate storage of state data required for hot-starting and enabling dynamic changes to certain model variables during a simulation, and (ii) remove the need for a class to be created for each parameter and variable in the model as Data objects can now be identified using the *ID*, *Name* and *Alias* attributes which are part of the *DataDef* element defined in the ModelConfiguration schema.

The UML Class Diagram in Figure 4.27 shows the structure of the restructured *Data* package within the *ACRU* model. The design of the new data structure is closely linked to the *Data* element in the *ModelData* schema and the associated *DataDef* element defined in the *ModelConfiguration* schema.

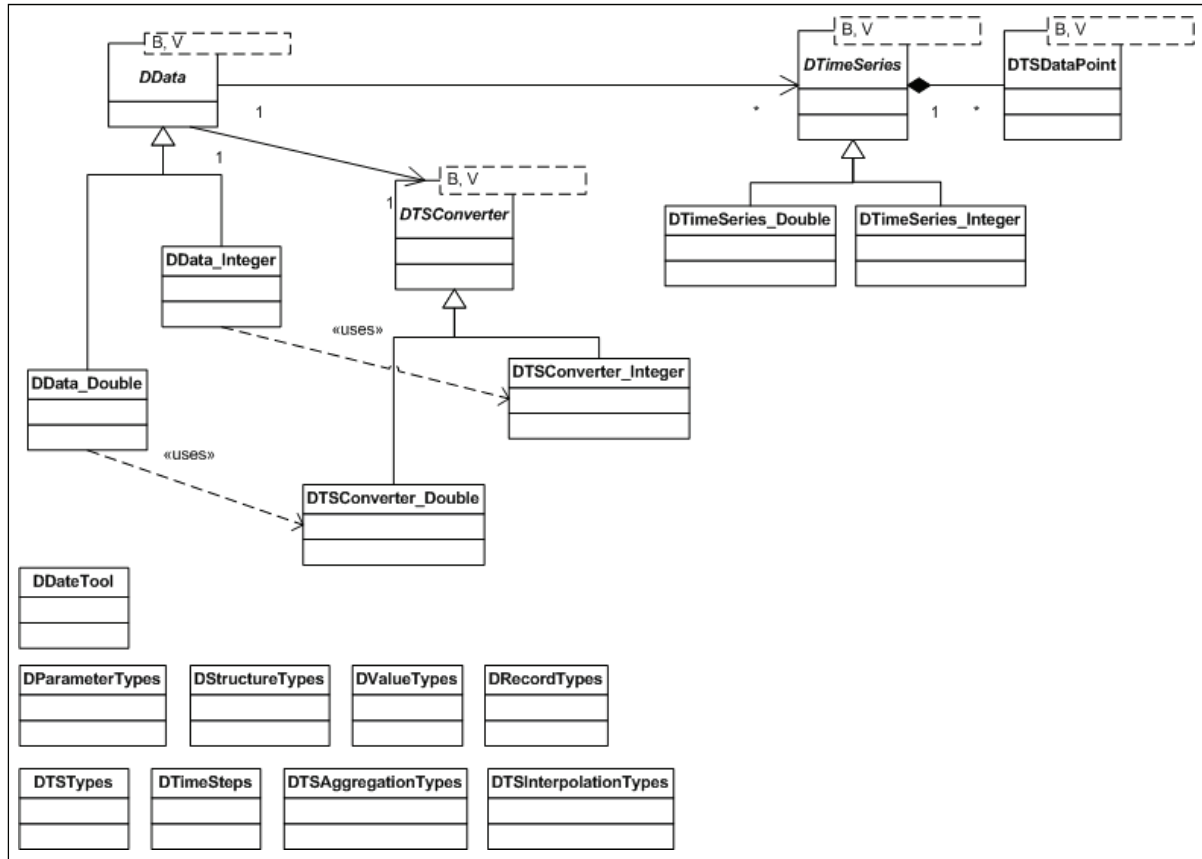


Figure 4.27 A UML Class Diagram showing the new main Data classes created for the *ACRUXml* version of the model

The *DData* class is still at the top of the Data hierarchy, but it has been completed recoded. The instance variables within the new *DData* class are shown and described in Table 4.2. The *id*, *name* and *alias* instance variables are identical to those in the *DataDef* element associated with the *Data* element used to create the instance of *DData*. The *parameterType*, *valueType*, *aggregationType* and *interpolationType* instance variables help describe the data values and how they should be processed. The *saveOption* instance variable indicates whether values in the instance of *DData* will be output or not as specified by the user. The *DData* class has been parameterised with a base data type represented by the generic declaration “*B*” and data value type represented by the generic declaration “*V*” to minimise code repetition in subclasses and ensure type safety at compile time. The purpose of the *baseType* instance variable is to store the Class of the base data type to assist in

casting values to the correct data type as the Java programming language does not provide a means to query the type of a generic parameter. Two categories of base data type are envisaged, “values” and “records”, where “values” are individual data values such as an individual integer or floating point value, and “records” are collections of data values such as an array of integer or floating point values. These “values” and “records” categories would also apply to time series, for example daily rainfall could be stored as a time series of floating point values, and a set of calibration constants that vary with time could be stored as a time series of records. For “value” base data types the base data type and the data value type would be the same.

One of the objectives in restructuring the *DData* class was to better represent variables that vary with time, either continuously or at irregular intervals. Some *ACRU* variables are termed “dynamic” in that they are usually considered as constant during a simulation, but could be set to change at intervals during the simulation, for example to simulate a land use change. This means that within the *DData* class it is necessary to make provision for a variable to be represented by either a constant value or a time series of values, as each variable can only be represented by one *DData* class. For constant parameters or variables the data value or values are stored in the *constantData* instance variable. For time series parameters or variables the data value or values are stored in an instance of the *DTimeSeries* class which is in turn stored in the *tsDatasets* instance variable. The *tsDatasets* instance variable is a Java Hashtable that can store a set of *DTimeSeries* instances, one for each time series type. Typically the *tsDatasets* instance variable would initially be populated with just one instance of *DTimeSeries*, and any aggregated or interpolated variations of the original time series could also be stored to save having to repeat the aggregation or interpolation process. It is envisaged that the main non-abstract Data classes shown in Figure 4.27 would be sufficient to represent all the variables required for the *ACRU* model, however, these classes could be subclassed if necessary to provide for any variable with specific requirements.

In order to better represent temporal data in *ACRU* the *DTimeSeries* and *DTSDatapoint* classes were created to provide a means of storing and transferring time series data in a data structure that is independent of, but used by the *DData* class and its subclasses. The *DTimeSeries* class is an abstract class that has been subclassed as shown in Figure 4.27 to provide concrete subclasses that include additional methods specific to the type of time series data stored. The *DTimeSeries* class contains an instance variable *tsTypeID* which stores the time series type, which is one of the time series types listed in Table 4.3. Each instance of the *DTimeSeries* class stores a chronologically ordered collection of data points

where each data point is represented by an instance of the *DTSDataPoint* class. Each instance of the *DTSDataPoint* class stores three pieces of information, a date/time, a data value or record and a data quality flag. The *DTimeSeries* and *DTSDataPoint* classes have also been parameterised with a base data type “*B*” and a data value type “*V*” the same as that described for the *DData* class.

Table 4.2 The instance variables belonging to the *DData* class and their descriptions

Variable	Type	Description
<i>id</i>	String	The ID of the instance of <i>DData</i> .
<i>name</i>	String	The name of the instance of <i>DData</i> .
<i>alias</i>	String	The alias of the instance of <i>DData</i> .
<i>parameterType</i>	int	The parameter type specifies whether the instance of <i>DData</i> represents a model input, output or state (input and output) variable or parameter (1 = INPUT, 2 = OUTPUT, 3 = STATE).
<i>valueType</i>	int	The value type specifies the data type of the data values stored in the instance of <i>DData</i> (1 = STRING, 2 = BOOLEAN, 3 = BYTE, 4 = SHORT, 5 = INT16, 6 = INTEGER, 7 = INT32, 8 = LONG, 9 = INT64, 10 = FLOAT, 11 = DOUBLE, 12 = DATETIME).
<i>aggregationType</i>	int	The aggregation type specifies how time series of values will be aggregated to a larger time step (0 = NONE, 1 = SUM, 2 = MEAN, 3 = MAXIMUM, 4 = MINIMUM).
<i>interpolationType</i>	int	The interpolation type specifies how breakpoint time series of values will be interpolated to estimate intermediate data values (0 = NONE, 1 = ISOLATED, 2 = STEP, 3 = STRAIGHTLINE, 4 = CUBICSPLINE).
<i>saveOption</i>	boolean	The save option indicates whether values in the instance of <i>DData</i> will be output or not as specified by the user.
<i>baseType</i>	Class<B>	The Class representing base type of the instance of <i>DData</i> , for example the base type for a variable represented by an individual double precision floating point value would be Class<Double> or for an array of String values would be Class<String[]>.
<i>constantData</i>	B	The data value to be stored by this instance of <i>DData</i> if it represents a constant parameter or variable.
<i>tsDatasets</i>	Hashtable<Integer, DTimeSeries<B,V>>	A Hashtable that may contain one or more time series of data values. More than one time series may be stored so that aggregated or interpolated time series may be stored to save them having to be recalculated.
<i>timeSeriesConverter</i>	DTSCConverter<B,V>	The instance of <i>DTSCConverter</i> to be used by the instance of <i>DData</i> to convert from one time series type to another.
<i>dataOwner</i>	IDDataOwner	The owner of this instance of <i>DData</i> , for example and instance of <i>MModel</i> or <i>CComponent</i> which both implement the <i>IDataOwner</i> interface.

Table 4.3 Time series types that apply to the *DTimeSeries* class and their descriptions

Time Series Type	Description
ANNUAL	Regular time series with data points at year intervals
MONTHLY	Regular time series with data points at month intervals
DAILY	Regular time series with data points at day intervals
HOURLY	Regular time series with data points at hour intervals
MINUTE	Regular time series with data points at minute intervals
SECOND	Regular time series with data points at second intervals
MILLISECOND	Regular time series with data points at millisecond intervals
BREAKPOINT_YEAR	Variable interval time series with data points at intervals of one or more whole years
BREAKPOINT_MONTH	Variable interval time series with data points at intervals of one or more whole months
BREAKPOINT_DAY	Variable interval time series with data points at intervals of one or more whole days
BREAKPOINT_HOUR	Variable interval time series with data points at intervals of one or more whole hours
BREAKPOINT_MINUTE	Variable interval time series with data points at intervals of one or more whole minutes
BREAKPOINT_SECOND	Variable interval time series with data points at intervals of one or more whole seconds
BREAKPOINT_MILLISECOND	Variable interval time series with data points at intervals of one or more whole milliseconds
MONTHOFYEAR	Set of 12 aggregated values, one for each month of the year
DAYOFMONTH	Set of 31 aggregated values, one for each day of the month
DAYOFYEAR	Set of 366 aggregated values, one for each day of the year, identified by day and month
DYNAMIC_MONTHOFYEAR	Variable interval time series with sets of month-of-year values at intervals of one or more whole years
DYNAMIC_DAYOFMONTH	Variable interval time series with sets of day-of-month at intervals of one or more whole months
DYNAMIC_DAYOFYEAR	Variable interval time series with sets of day-of-year values at intervals of one or more whole years

It is common when dealing with time series data to need to aggregate time series values to obtain an aggregated value at a courser time step, for example from daily to monthly or annual. It is also common when dealing with breakpoint time series to need to interpolate between two data points to obtain an estimated data value at specified point in time. For this purpose the *DTSCConverter* class and its subclasses were developed to enable conversion, that is aggregation or interpolation, between time series of different time series types. The *DTSCConverter* class contains one public method named *convert* and a set of protected methods each handling to the conversion from one specific time series type to another specific time series type, for example the method named *convertDailyToMonthly* to convert from a fixed interval daily time series to a fixed interval monthly time series. The *convert* method would be called from the code and, based on the input time series type and other conversion parameters, would in turn call the relevant conversion method. The



*DTSCConverter* class has also been parameterised with a base data type “*B*” and a data value type “*V*” the same as that described for the *DData* class. In the *ACRU* model individual Processes require input data at varying levels of detail, and part of the rationale of the *DTSCConverter* class is to take care of the conversion process internally to save the user having to do this, especially when linking models which operate at different time scales. The *DDateTool* class was created as a utility class to simplify use of the Java *Date* class by providing methods to deal with formatting dates and times as strings and returning the next or previous date or time relative to a specified data or time as the specified time step.

The *ModelData* and *ModelConfiguration* XML schemas used for *ACRU* model input allow *Data* elements to be specified for the *Model* element and not just for *Component* elements, so that general model parameters may be handled in a similar manner to the parameters and variables used to describe the *Components* of the hydrological system being modelled. This concept has been carried through to the *ACRUXml* version of the *ACRU* model. The *IDDataOwner* interface shown in Figure 4.23 has been created and is implemented by both the *MModel* and the *CComponent* class so that certain operations related to instances of the *DData* class and their owner or container class can be standardised and thereby simplify and reduce the code required for these shared operations.

As the *DData* objects belonging to a *CComponent* object are now identified by the *ID* attribute instead of the class name it was necessary to make changes to the *CComponent* class to put this into effect. Implementation of the new internal data structure of *ACRU* also required that changes be made to the code for classes in *ACRU* that deal with model input, model creation and model output.

#### **4.3.4 Resources**

In the *ACRU* model the physical entities making up the hydrological system being modelled are represented using discrete *Components* though in reality catchment attributes such as soil types and land cover are far from discrete. *ACRU* as a model is a simplification of reality and a modeller setting up *ACRU* will need to decide on a suitable level at which to discretise the system being modelled. The *ACRU* model is a hydrological model and thus its main purpose is to model the water resource within a given hydrological system, but it also includes modules to model other matter such as salinity, nitrogen and phosphorus which could also be generically described as resources. These resources are also in a sense a physical part of the system being modelled but are continuous within a hydrological system. This leads to the question regarding how these resources should be represented in the

ACRU model. Should they be represented as Components or Data? In addition, it should be considered that not only resource quantities need to be modelled but also ownership and location of portions of the resource. Water and other resources were modelled in the ACRU2000 version of the model using a special Data class *DFluxRecord* which was subclassed for each type of resource being modelled, for example *DWaterFluxRecord*, *DNitrateFluxRecord* and *DLabilePFluxRecord*, among many others. Each Component contained an instance of *DWaterFluxRecord* representing the portion of the water resource contained within the Component. These *DFluxRecord* classes worked well and enabled the model to keep track of how much of each resource was contained within each Component, whether the resource was unallocated or owned by one or more water owner Components and from which Component and to which Component a quantity of the resource had been received or lost.

This concept of resources has been taken a step further in this project by introducing the *RResource* class. The *RResource* class contains an internal variable *resourceTypeID* which enables the ID of the resource type being represented, for example "WATER", to be stored. As shown in Figure 4.23 each instance of *CComponent* may contain one or more instances of *RResource* where each instance of *RResource* would have a different ID representing a different resource type. The *RResource* class stores the quantity of water stored within the parent Component and which Components if any own a portion of the stored resource. It also stores a record of influxes and outfluxes of the resource, not only the quantities but also the source and destination Components. It also stores a record of the quantities of the resource owned by the parent Component but stored in other Components. The *RResource* class has been subclassed as *RResource\_Double* and *RResource\_Integer* to represent different data value types. The *RResource* class contains several methods enabling the resource to be transferred between Components or transformed to a different form, for example Nitrate to Ammonium, where Nitrate and Ammonium are modelled as different resources. The *RResource* class offers the following advantages over the *DFluxRecord* class: it better represents the concept of resources, it does not have to be subclassed for each different resource type making the model more extensible, it enables resource ownership information to be retained as state data for hotstarting and can be represented more easily in the ModelConfiguration schema than a complex specialised *DFluxRecord* Data class. The ModelConfiguration schema was extended to include the *ResourceTypes* and *ResourceType* elements within the *ModelConfiguration* element and to include the *ResourceDefinitions* and *ResourceDef* elements within the *ComponentType* element as shown in Figure 4.28. The *ResourceType* element is used to define each resource type that

can modelled. The *ResourceDef* element specifies the resource types that can exist within each *ComponentType* element.

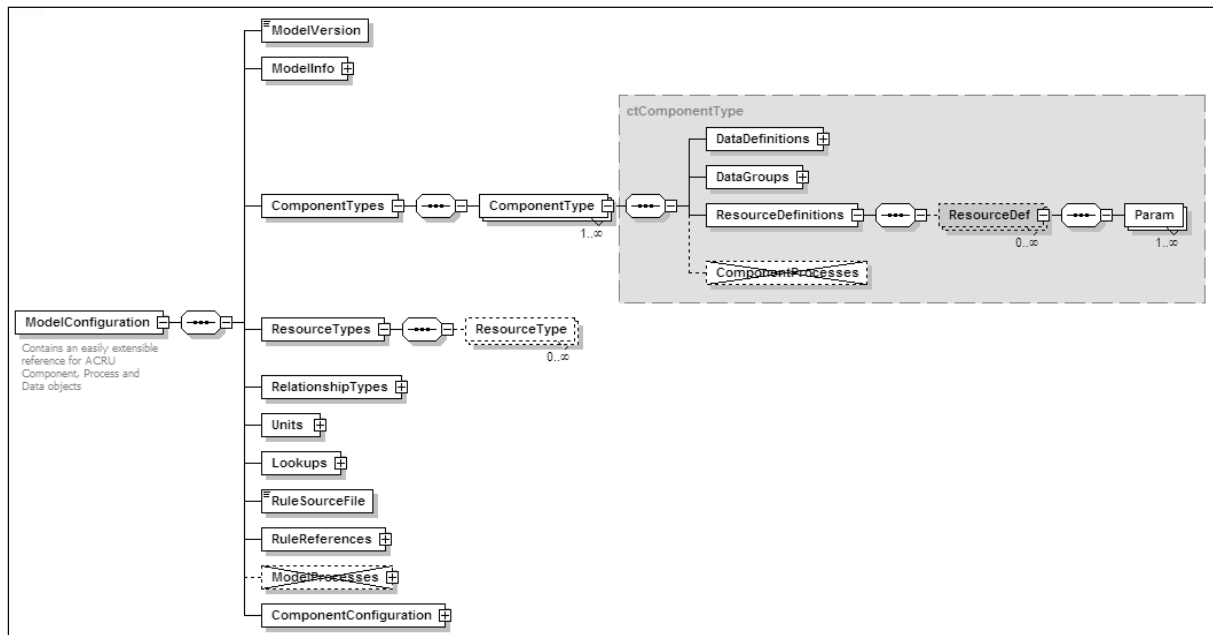


Figure 4.28 A diagram of the extended ModelConfiguration schema showing the new Resource related elements

### 4.3.5 Scenarios

The ability to be able to easily and efficiently configure and run a range of different scenarios is especially useful when modelling for water resources planning. The manner in which data for different modelled scenarios is handled within a ModelData file is explained in detail in Section 4.2.1. The implementation of this functionality within the *ACRU* model itself was relatively straight forward as only the *AAcruXmlModelInput* class within *ACRU* required a few changes to ensure that only the data values for the relevant scenario are read into the model. The *AAcruXmlModelOutput* class also required a few changes to ensure that when state data is saved back to a ModelData file it is saved within the correct data *Scenario* element. While the setting up of scenarios in a ModelData file may be complex, if they are configured correctly the *ACRU* model itself “sees” only one set of data values at a time and simulations will proceed as before.

### 4.3.6 State data and hotstarting

The necessity for *ACRU* to be able to store data and be hotstartable was discussed in the introduction to Section 4.3.3. The new XML model input data structures and the new model

internal data structures have been designed and developed to facilitate the storage of state data. The *AAcruXmlModelInput* and *AAcruXmlModelOutput* classes within *ACRU* have been modified to read state data in, to initialise state variables at the beginning of a simulation and to save state data back to these variables at user specified times during the simulation. Additional model input variables were created for the main internal state variables to enable initial states to be specified.

#### **4.3.7 Dynamic data variables**

As discussed in the introduction to Section 4.3.3, in some cases providing land use variables as constants is reasonable, while in other cases especially for simulations over a long time period, it may be necessary to vary the values of the land use variables over time to reflect real land use changes. The new model input data structures provided in the *ModelData* and *ModelConfiguration* schemas enable model variables in which data values can be entered as constant values or time series of values as explained in Section 4.2.2. An advantage of the new data structure is that dynamic variables use the same data structure and model input file as other variables and can thus be configured and edited using the same software utilities. This renewed and hopefully improved ability for *ACRU* to handle dynamically changing variable values is important for modelling land use changes over time and forms part of the model's ability to model real world complexity.

#### **4.3.8 Data readers and writers**

As part of further development of the *ACRU* model completed in WRC project K5/1870 a software library named *ModelDataAccess* was developed for the .Net platform. This software library contains a set of classes to read and write data from and to several data formats including SPATSIM-HDSF database, *ACRU* Single format, *ACRU* Composite format and *ACRU* CompositeY2K format, where each of these reader/writer classes implements a common interface named *IDataReaderWriter*. The *ACRU* model previously contained several classes to read and in some cases write to these same data formats. As part of this project a Java version of the *ModelDataAccess* library reader/writer classes has been created, where the *IDataReaderWriter* interface and the *DataReaderWriter\_AcruSingle*, *DataReaderWriter\_AcruComposite* and *DataReaderWriter\_AcruCompositeY2K* class have been duplicated and additional *DataReaderWriter\_AcruCSV* and *DataReaderWriter\_DBF\_AcruOutput* have been developed.

#### 4.4 Summary and Conclusions

A considerable amount of work has been done in this project to revise the initial design of the *ModelData* and *ModelConfiguration* schemas and this has resulted in a design that is not only more robust but is expected to provide the *ACRU* model with model input functionality necessary for both planning and operations modelling. The design of these schemas is expected to be stable from this point on and no substantial changes to the design are expected. Following from the revision of the design for the *ModelData* and *ModelConfiguration* schemas various changes were made to the *ACRU* model itself to be compatible with and make full use of model input and configuration files that uses these schemas and to make the model more suitable for use for water operations modelling. Restructuring the data structure used within the *ACRU* model was an bigger undertaking than initially anticipated largely due to the complexity of dynamic type variables, however, this restructuring was vital in enabling the *ACRU* model to handle time series better especially with regard to state and dynamic type data variables. The Component structure was also revised to simplify it and make it compatible with new concepts introduced to the *ACRU* model configuration file such as HRUs. The introduction of the concept of resources using the *RResource* class was also a step forward from a conceptual and model extensibility point of view. Further development of the *ACRU* model has taken place to implement new functionality such as scenarios, hotstarting and the storage of state data, dynamic variables and flexible spatial component configurations. A Java version of the *ModelDataAccess* was also created.

## 5 IMPLEMENTATION OF OPENMI FOR MODEL LINKING

DJ Clark and A Lutchminarain

The OpenMI interface specification standard was accepted as the most appropriate linking mechanism for use in the project. The OpenMI Association supports the standard and has released the OpenMI 1.4 version of the OpenMI Standard and associated OpenMI Software Development Kit (SDK) in 2005. OpenMI 1.4 was adopted and implemented in certain models (Gijsbers *et al.*, 2010). Gijsbers *et al.* (2010) state the OpenMI Association core group realized that through their own implementation experiences and feedback there were certain aspects of the standard that needed to be improved. The result of these improvements was OpenMI 2.0. The OpenMI 2.0 version of the OpenMI Standard was recently released to provide additional user requirements not met by the OpenMI 1.4 version. The OpenMI 2.0 version of the SDK for OpenMI 2.0 has not been officially released yet though the working code has been made available.

OpenMI 1.4 is restricted to only one type of model, which is a model that progresses in time during the simulation. OpenMI 2.0 introduces a base interface, which is a very generic interface that can be extended to support different types of models, including models that progress in time (OpenMI, 2012).

OpenMI 1.4 uses a link object to connect two models. This link object contains references to the source and target models, the output and the input exchange items, and data operations (OpenMI, 2012). An example showing the link object used to link components in OpenMI 1.4 is shown in Figure 5.1.

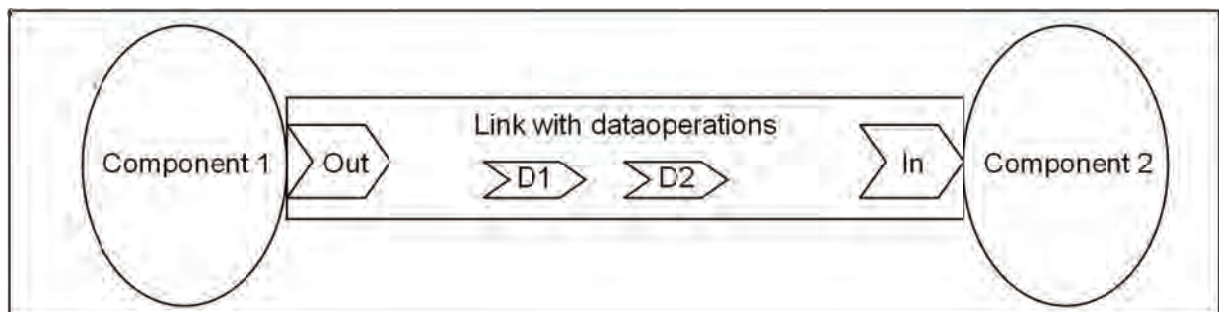


Figure 5.1 Illustration depicting a link between components in OpenMI 1.4 (OpenMI, 2012)

According to Gijbbers *et al.*(2010) the *ILink* interface used for linking models in OpenMI 1.4, was not intuitive as there was no clear ownership between models. *ILink* was also not an efficient solution, as it was memory intensive, and with larger study areas, this memory consumption would become excessive (Gijbbers *et al.*, 2010).

In the OpenMI 2.0 Standard the link object has been replaced and there is now a direct reference between input exchange items and output exchange items. This direct reference is more intuitive as there is a direct link between the output exchange item and the input exchange item, clearly indicating ownership. The data operations have been replaced by adapted outputs, to ensure the data from the output exchange item is in the correct form for the input exchange item. One or more adaptors can be applied to the data between the output exchange item and the input exchange item (OpenMI, 2012). An example of linking models in OpenMI 2.0 is shown in Figure 5.2; a single output exchange item is linked to three input exchange items, with data adaptors applied to each link. The direct relationship between input and output exchange items in OpenMI 2.0 consumes less memory. Other improvements include increased flexibility of data definitions and improved requests for values and control flow. Further details on these improvements can be found in OpenMI (2012).

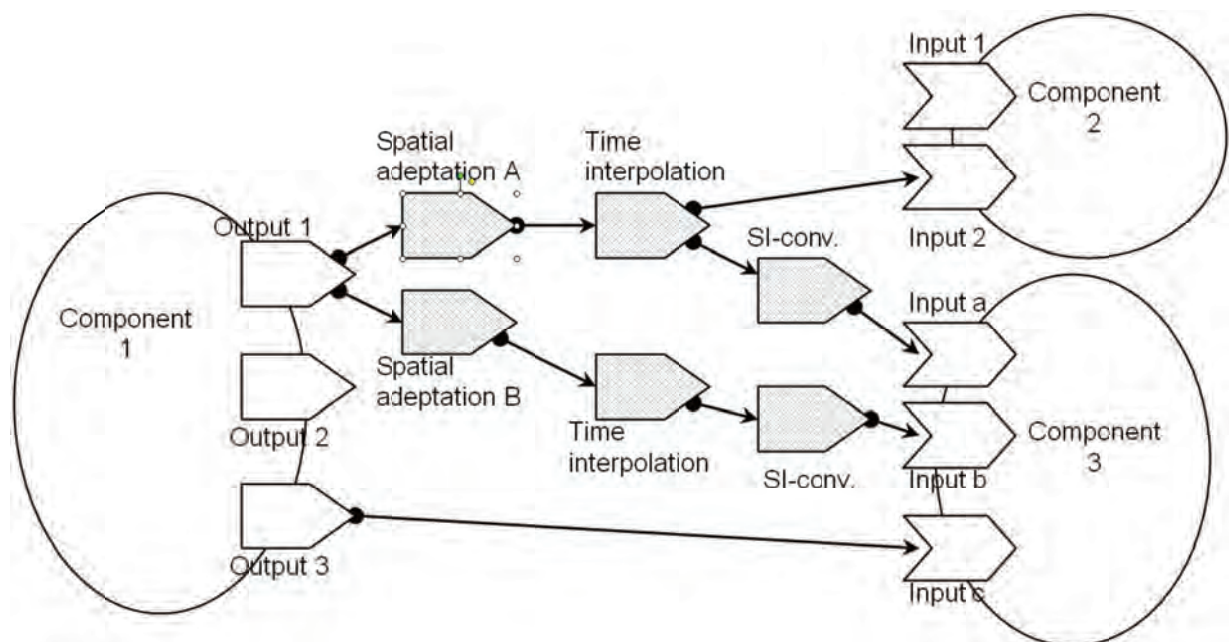


Figure 5.2 An example of linking components in OpenMI 2.0 (OpenMI, 2012)

Initially it was decided that the OpenMI 2.0 version should be used in the project as it was the more advanced version. The working code of the OpenMI 2.0 SDK was used and according to OATC (2012), working code is not tested as extensively as released code but should be fairly stable to use due to testing carried out during the development phase.

Though the wrapper itself was successfully created, several problems were encountered while trying to run it using the Configuration Editor graphical user interface provided for OpenMI 2.0. Some of these problems in the working code for OpenMI 2.0 were fixed enabling the wrapper code to be run. The implementation of the wrapper using the OpenMI 2.0 version of the standard and problems and fixes are described in Section 5.1.2. Further testing revealed additional problems in the working code of the OpenMI 2.0 SDK and a decision was made to rather implement the wrappers for *ACRU* and MIKE BASIN in the stable OpenMI 1.4 version.

There are two main approaches to making a model OpenMI compliant, which is through the use of, either an OpenMI compliant wrapper or direct implementation in a model's source code. A model that is made OpenMI compliant by either of these approaches is referred to as a *LinkableComponent*.

In this project both *ACRU* and MIKE BASIN are existing models. Blind *et al.* (2005) recommended that the wrapping approach should be used for existing model. One of the advantages of using the wrapping approach is that it minimises the amount of changes made to the model. Another advantage of using the wrapping approach is that the OpenMI specific code can be separated from the model's engine implementation (Blind *et al.*, 2005). This allows the model's engine to still be used in its existing standalone application.

To meet OpenMI compliance a model would need to implement the *ILinkableComponent* interface of the OpenMI standard. The OpenMI Association realised that most time stepping models have common functionality, and so provided the *LinkableEngine* abstract class and the *IEngine* interface within the OpenMI 1.4 SDK, to aid in the creation of OpenMI 1.4 compliant wrappers. The OpenMI Association recommends creating two classes to form an OpenMI 1.4 compliant wrapper for a model as explained in Blind *et al.* (2005) and shown in Figure 5.3.

The first wrapper class interacts with the model engine of the model being wrapped, by implementing the methods of the *IEngine* interface. These methods interact with the model engine and provide access to the model engine to any class that uses the first wrapper class. In Figure 5.3 the *MyEngineWrapper* is an example of the first wrapper class that implements the *IEngine* interface and interacts with the example model engine *RiverModel.dll*. The methods that need to be implemented by the first wrapper class are shown in Table 5.1 The populated model component referred to in Table 5.1 is the model engine populated with data for a particular study area.



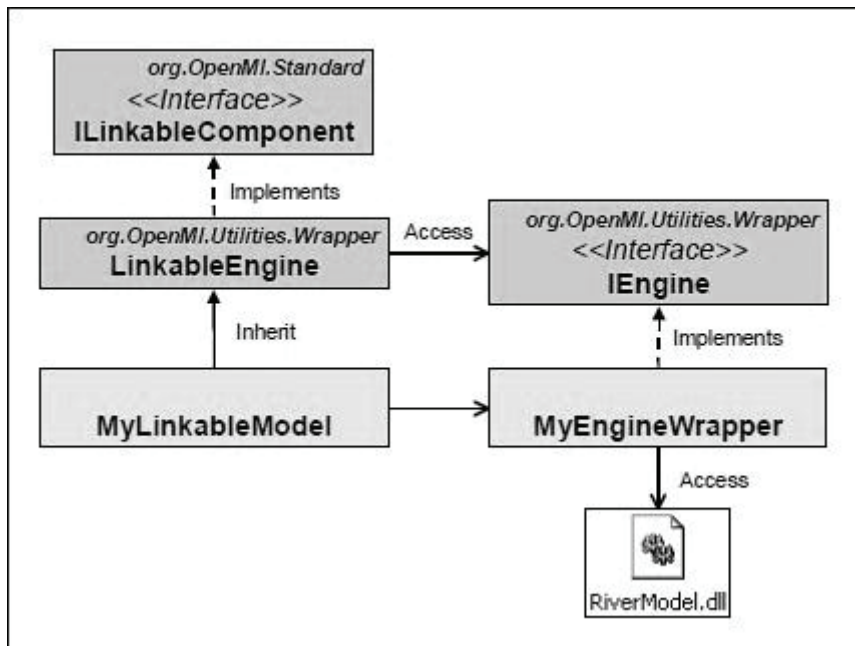


Figure 5.3 An example of an OpenMI compliant wrapper (Blind *et al.*, 2005)

The second wrapper class communicates with other OpenMI compliant models and provide access to the model engine through the first wrapper class. It inherits the *LinkableEngine* abstract class, which has already implemented most of the methods for the *ILinkableComponent* interface. The only method that needs to be implemented by the second wrapper class, is the *SetEngineApiAccess()* method, which provides access to the model engine. In Figure 5.3 the *MyLinkableModel* class is an example of the second wrapper class that inherits the *LinkableEngine* abstract class, and uses the *MyEngineWrapper* class to provide access to the example model engine *RiverModel.dll* in its implementation of the *SetEngineApiAccess()* method.

According to the OATC (2010) and Blind *et al.* (2005) each model that is set up for a study or research area must have an associated xml (OMI) file containing information about the model, and its capabilities and also indicates its availability to be linked to other OpenMI compliant models. An example of an OMI file is shown in Figure 5.4. This OMI file is used by the OpenMI Configuration Editor to identify, instantiate and configure a *LinkableComponent*. The Configuration Editor can be used to create and run a composition of linked *LinkableComponents*. Both OpenMI 1.4 and OpenMI 2.0 provide a Configuration Editor within their respective SDK's. An example showing *LinkableComponents* loaded and connected in the OpenMI 1.4 Configuration Editor is shown in Figure 5.5.

Table 5.1 The *IEngine* interface methods

Method Name	Method Description
GetModelID()	Returns the ModelID of the populated model component.
GetModelDescription()	Returns a description of the populated model component.
GetComponentID()	Returns the name of the non-populated component. This is typically the name of your model engine.
GetComponentDescription()	Returns a description of the non-populated component. This typically gives a description of the model engine.
GetTimeHorizon()	Returns the time horizon for the populated model component, which is typically the same as the simulation period.
GetInputExchangeItemCount()	Returns a count of the input exchange items for the populated model component.
GetOutputExchangeItemCount()	Returns a count of the output exchange items for the populated model component.
GetOutputExchangeItem()	Returns index based output exchange item from the populated model component.
GetInputExchangeItem()	Returns index based input exchange item from the populated model component.
Initialize()	The first method called to initialize the model and populate the model engine with data, to create a populated model component.
PerformTimeStep()	This method will execute the model engine for one time step of the simulation period.
GetCurrentTime()	Returns the current time of the model engine.
GetInputTime()	Returns the time for which the next input is needed for a specific Quantity and ElementSet combination
GetEarliestNeededTime()	This method returns the earliest needed time. The period from the earliest needed time till the current time is the buffer period. Generally for most time stepping model engines this time will be the time for the previous time step.
SetValues()	This method sets the values in a model engine.
GetValues()	This method gets the values in a model engine.
GetMissingValueDefinition()	Gets the missing value, which is used in place of a value that cannot be returned.
Finish()	This method is invoked after all calculations are carried out. Typically de-allocation of memory and closing of any files implemented in this method.
Dispose()	This method will be invoked after the Finish method has been invoked to dispose of any objects.

```

<?xml version="1.0"?>
<LinkableComponent xmlns="http://www.openmi.org"
Type="UKZN.OpenMI.MIKE_BASIN.MIKEBASINLinkableComponent" Assembly="UKZN.OpenMI.MIKE_BASIN.dll">
  <Arguments>
    <Argument Key="Id" Readonly="true" Value="MIKE BASIN OPENMI MODEL - 1" />
    <Argument Key="Caption" Readonly="true" Value="MIKE BASIN OPENMI MODEL - 1" />
    <Argument Key="SimulationDescription" Readonly="true" Value="MIKE BASIN OPENMI MODEL" />
    <Argument Key="workingDirectory" Readonly="true" Value="D:\Implementation\MIKE
BASIN\MIKE_BASIN_MODEL\Fraction_Catch" />
    <Argument Key="MSaccessFileName" Readonly="true" Value="Catch.mdb" />
  </Arguments>
</LinkableComponent>

```

Figure 5.4 Example of an OMI file

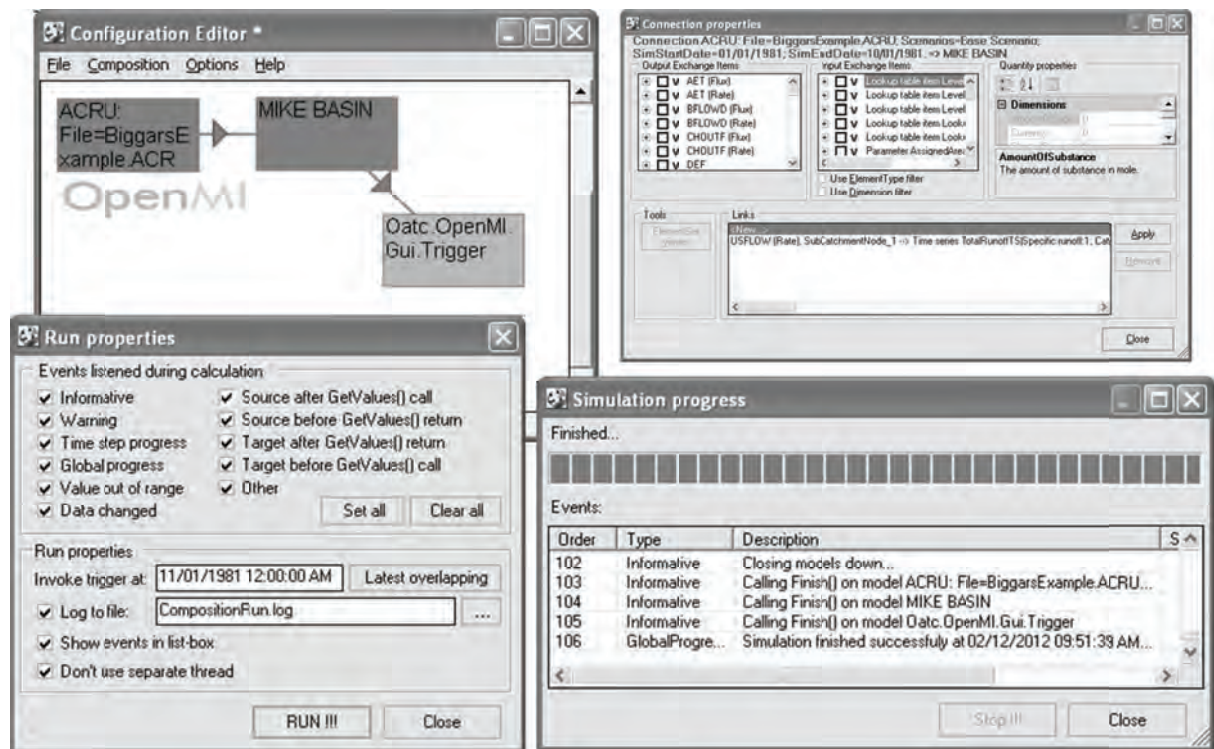


Figure 5.5 Example showing *LinkableComponents* loaded and connected in the OpenMI 1.4 Configuration Editor

This chapter describes the process of implementing OpenMI wrappers for the ACRU and MIKE BASIN models.

### 5.1 Development of OpenMI Wrapper for MIKE BASIN

As explained in the model evaluation in Section 3.2.2, MIKE BASIN is not OpenMI compliant but does provide access to its model engine. Wrapper code will be written around the MIKE BASIN model engine, which will interact with the model engine and provide the required functionality for OpenMI compliance. The wrapper code controls how a MIKE BASIN model will run and will enable a MIKE BASIN model to link to other OpenMI compliant models. The

OpenMI compliant wrapper together with the MIKE BASIN model engine will be referred to as the MIKE BASIN *LinkableComponent*.

Initially it was decided that the OpenMI 2.0 version should be used in the project as it was the more advanced version but due to several problems encountered in the working code, a decision was made to rather implement the wrapper in the stable OpenMI 1.4 version. For both versions of OpenMI, the Microsoft .NET C# programming language was used in the creation of the OpenMI compliant wrapper. The Microsoft .NET versions of the OpenMI standards and software development kits were used. The reason for using the Microsoft .NET platform was because MIKE BASIN's access to its model engine had been created using this platform. The following sections detail the creation of the wrappers using the .NET OpenMI 1.4 SDK and the .NET OpenMI 2.0 SDK working code.

### 5.1.1 OpenMI 1.4 wrapper

The OpenMI 1.4 compliant wrapper was developed for MIKE BASIN by creating two classes. The first wrapper class named *MBEngineWrapper*, which inherits the *IEngine* abstract class overriding the abstract methods to access specific behaviour of the MIKE BASIN model engine. The second class named *MBLinkableComponent*, inherits the *LinkableEngine* abstract class, which uses the first class to provide access to the MIKE BASIN model engine to other OpenMI 1.4 compliant models. These two classes form the wrapper for the MIKE BASIN model, enabling MIKE BASIN to communicate with other OpenMI 1.4 compliant models. The methods of the MIKE BASIN engine that were used to implement the methods of the *IEngine* abstract class in the *MBEngineWrapper* class are detailed in Table 5.2.

Table 5.2 Implementation of the *IEngine* interface methods in the *MBEngineWrapper* class

Method Name	Description of Method Implementation
<i>GetModelID()</i>	Retrieved a model ID value from the OMI file.
<i>GetModelDescription()</i>	Returned the <i>SimulationDescription</i> property of the MIKE BASIN model engine.
<i>GetComponentID()</i>	Hard-coded "MIKE BASIN" as the component ID.
<i>GetComponentDescription()</i>	Hard-coded a description of the MIKE BASIN model engine.
<i>GetTimeHorizon()</i>	Returns the time span between the <i>SimulationStart</i> and <i>SimulationEnd</i> properties of the MIKE BASIN model engine.
<i>GetInputExchangeItemCount()</i>	Returns a count of the ArrayList of input exchange items called <i>_inputExchangeItems</i> defined in the <i>MBEngineWrapper</i> class.

Table 5.2 (continued) Implementation of the *IEngine* interface methods in the *MEngineWrapper* class

Method Name	Description of Method Implementation
<i>GetOutputExchangeItemCount()</i>	Returns a count of the ArrayList of output exchange items called <i>_outputExchangeItems</i> defined in the <i>MEngineWrapper</i> class.
<i>GetInputExchangeItem()</i>	Returns index based input exchange item from the <i>_inputExchangeItems</i> defined in the <i>MEngineWrapper</i> class.
<i>GetOutputExchangeItem()</i>	Returns index based output exchange item from the <i>_outputExchangeItems</i> defined in the <i>MEngineWrapper</i> class.
<i>Initialize()</i>	The <i>SimulationDescription</i> property of the MIKE BASIN engine is assigned. The <i>Initialize</i> method of the MIKE BASIN engine is used to initialize and populate the MIKE BASIN model. The <i>SimulationStart</i> property of the MIKE BASIN model engine is used to assign the properties <i>_simulationStartTime</i> and <i>_dtCurrentDateTime</i> of the <i>MEngineWrapper</i> class. The <i>SimulationEnd</i> property of the MIKE BASIN model engine used to assign the property <i>_simulationEndTime</i> of the <i>MEngineWrapper</i> class. The MIKE BASIN model engine methods <i>GetModelObject()</i> , <i>GetExtendedInfo()</i> , <i>GetInputSpecs()</i> , and <i>GetResultSpecs()</i> are used to create the input and output exchange items a MIKE BASIN model requires and provides.
<i>PerformTimeStep()</i>	The MIKE BASIN model engine uses the <i>SimulateTimeStep()</i> and <i>AdvanceTimeStep()</i> methods to execute the MIKE BASIN model engine for one time step of the simulation period.
<i>GetCurrentTime()</i>	Returns the <i>_dtCurrentDateTime</i> property of the <i>MEngineWrapper</i> class.
<i>GetInputTime()</i>	Returns the <i>_dtCurrentDateTime</i> property of the <i>MEngineWrapper</i> class advanced by the MIKE BASIN model engine <i>TimeStep</i> property in seconds.
<i>GetEarliestNeededTime()</i>	Returns the <i>_dtCurrentDateTime</i> property of the <i>MEngineWrapper</i> class, as this is the time step that has already been executed.
<i>SetValues()</i>	The <i>GetModelObject()</i> , <i>GetInputSpecs()</i> , <i>FindInputIndex()</i> and <i>SetInput()</i> methods of the MIKE BASIN model engine are used to set the values in the MIKE BASIN model.
<i>GetValues()</i>	The <i>GetModelObject()</i> and <i>GetCurrentResult()</i> methods of the MIKE BASIN model engine are used to get the values from the MIKE BASIN model.
<i>GetMissingValueDefinition()</i>	This is a hard-coded value.
<i>Finish()</i>	The <i>FinishSimulation()</i> method of the MIKE BASIN model engine is invoked.
<i>Dispose()</i>	The <i>Dispose()</i> method of the MIKE BASIN model engine is invoked.

A simple OpenMI 1.4 compliant test application, named Test, was also created for use in testing the MIKE BASIN wrapper. The Configuration Editor graphical user interface, provided by the OpenMI Association for setting up linked model runs, was used to link and run the Test *LinkableComponent* and the MIKE BASIN *LinkableComponent*. The Configuration Editor with the linked composition loaded is shown in Figure 5.6 . The results of the model runs demonstrated that the development of the OpenMI 1.4 compliant wrapper for MIKE BASIN had been successful.

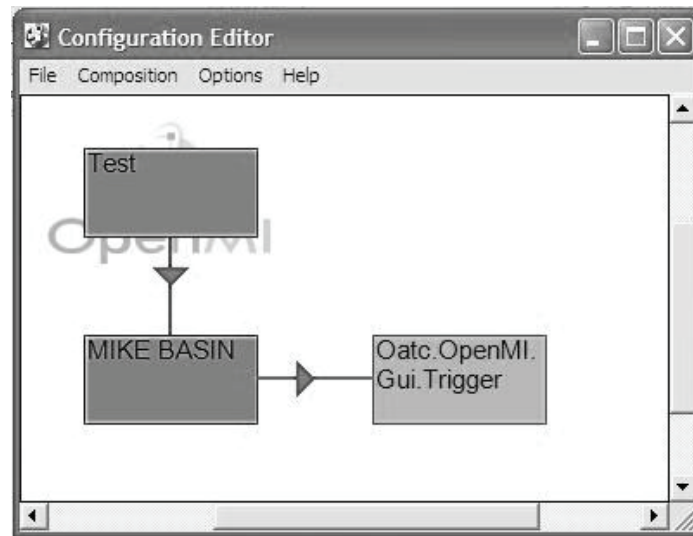


Figure 5.6 The OpenMI 1.4 SDK configuration editor

It is important to note that no code changes had to be made to the OpenMI 1.4 SDK during the implementation. A problem was encountered with regards to the MIKE BASIN *LinkableComponent* writing its output data to a file. After some debugging of the code of the Configuration Editor contained in the OpenMI 1.4 SDK, it was found that running the linked composition using threads was the cause of the problem. This was solved by selecting the checkbox option *Don't use separate thread* in the *Run properties* dialog box of the Configuration Editor as show in Figure 5.7.

A test was conducted to verify if a data operation could be applied to an output exchange item of the OpenMI compliant wrapper. A simple class named *MultiplyByFactor*, inherited from the *IDataOperation* class, was created which multiplied the output values passed to it by a factor. The *MultiplyByFactor* data operation was added to the OpenMI compliant wrapper solution and successfully applied to output exchange items.

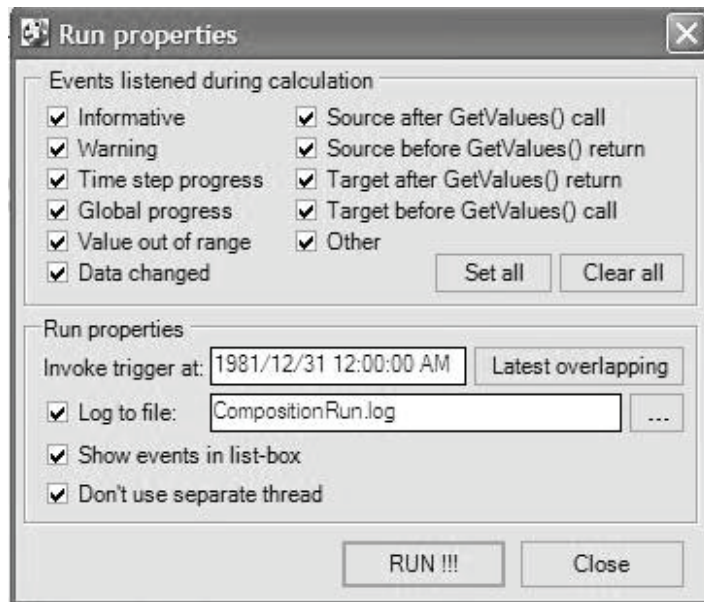


Figure 5.7 The *Run properties* dialog box of the configuration editor

A .NET profiler was used to evaluate the performance and memory usage of the MIKE BASIN *LinkableComponent* against the MIKE BASIN engine being run standalone. In both cases the same test model configuration was used. The OpenMI 1.4 Configuration Editor was used to run the MIKE BASIN *LinkableComponent*. A test application had been written to run the MIKE BASIN engine standalone. The results of this evaluation showed there is a slight increase in memory usage and decrease in performance of the MIKE BASIN *LinkableComponent* compared to the MIKE BASIN engine being run standalone. This can be expected as there are additional data being stored and operations performed with the MIKE BASIN *LinkableComponent* compared to the MIKE BASIN engine being run standalone.

An evaluation of the performance and memory usage of the OpenMI 1.4 MIKE BASIN *LinkableComponent* showed that there is a slight increase in memory usage and decrease in performance of the MIKE BASIN *LinkableComponent* compared to the MIKE BASIN engine being run standalone. This was expected as there are additional data being stored and operations performed with the MIKE BASIN *LinkableComponent*. These slight increases in memory usage and decrease in performance would be multiplied for larger model configurations and compositions of *LinkableComponents*. The flexibility offered by the OpenMI interface specification standard to link to other OpenMI compliant models results in a performance and memory usage penalty. The OpenMI Association has released OpenMI 2.0 interface specification standard that has attempted to decrease memory consumption and improve the performance of *LinkableComponents* in OpenMI but they have not officially released a stable version of the OpenMI 2.0 SDK. Further testing of the OpenMI 1.4 MIKE

BASIN *LinkableComponent* will need to be conducted to determine if the benefits of running models linked in parallel using OpenMI outweighs the performance and memory usage penalties experienced.

### 5.1.2 OpenMI 2.0 wrapper

Initially it was decided that the OpenMI 2.0 version should be used in the project as it was the more advanced version. The wrapper was successfully created but several problems were encountered while trying to run it using the Configuration Editor graphical user interface provided for OpenMI 2.0. Some of these problems in the working code for OpenMI 2.0 were fixed enabling the wrapper code to be run. The implementation of the wrapper using the OpenMI 2.0 version of the standard and problems and fixes are described in this section

According to OATC (2010) a model needs to implement the *OpenMI.Standard2.IBaseLinkableComponent* interface to be OpenMI 2.0 compliant. The *OpenMI.Standard2.IBaseLinkableComponent* interface is the base interface and the *ITimeSpaceComponent* is an extension of this base interface for models that progress in time. A single class needed to be created for the OpenMI 2.0 compliant wrapper which inherited the abstract class called the *LinkableEngine*, which is a default implementation of the *TimeSpaceComponent* interface. The OpenMI 2.0 compliant wrapper developed for MIKE BASIN inherits the *LinkableEngine* abstract class, and overrides the abstract methods to implement specific behaviour of the MIKE BASIN model engine. The *LinkableEngine* abstract class with its abstract methods in italics is shown in Figure 5.8. The method names not shown in italics in Figure 5.8 indicate functionality that is already implemented, which the OpenMI 2.0 compliant wrapper does not have to override.

The *LinkableEngine* abstract class has fewer methods that need to be implemented compared the *IEngine* abstract class of OpenMI 1.4. These methods that needed to be implemented by the OpenMI 2.0 compliant wrapper are shown in Table 5.3

The single class that was created for the OpenMI 2.0 compliant wrapper was called *MIKEBASINLinkableComponent*. It inherits the *LinkableEngine* abstract class. The methods of the MIKE BASIN engine that were used to implement the methods of the *LinkableEngine* abstract class in the *MIKEBASINLinkableComponent* class are detailed in Table 5.4



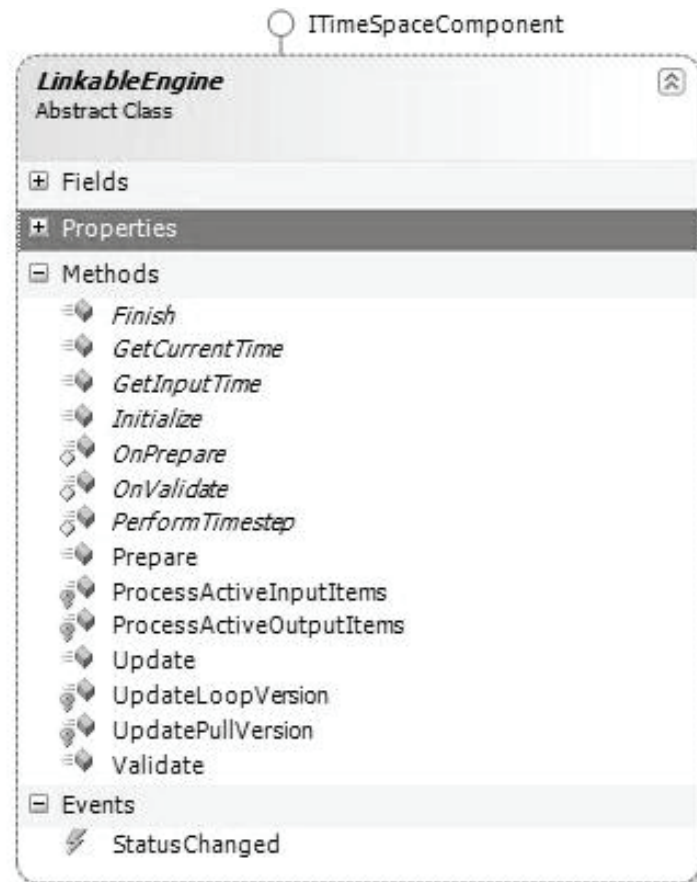


Figure 5.8 *LinkableEngine* abstract class showing methods

Table 5.3 The *LinkableEngine* abstract class methods

Method Name	Method Description
Initialize()	The first method called to initialize the model and populate the model engine with data, to create a populated model component.
PerformTimeStep()	This method will execute the model engine for one time step of the simulation period.
GetCurrentTime()	Returns the current time of the model engine.
GetInputTime()	Returns the time for which the next input is needed.
OnPrepare()	This method carries out any additional preparation operations required before running the model.
OnValidate()	This method carries out any validation operations required.
Finish()	This method is invoked after all calculations are carried out. Typically de-allocation of memory and closing of any files implemented in this method.

Table 5.4 Implementation of the *LinkableEngine* abstract class methods

Method Name	Description of Method Implementation
<i>Initialize()</i>	The <i>SimulationDescription</i> property of the MIKE BASIN engine is assigned. The <i>Initialize</i> method of the MIKE BASIN is used to initialize and populate the MIKE BASIN model. The <i>SimulationStart</i> property of the MIKE BASIN model engine used to assign the properties <i>_simulationStartTime</i> , <i>_dtCurrentDateTime</i> of the <i>MIKEBASINLinkableComponent</i> class. The <i>SimulationEnd</i> property of the MIKE BASIN model engine used to assign the property <i>_simulationEndTime</i> of the <i>MIKEBASINLinkableComponent</i> class. The MIKE BASIN model engine methods <i>GetModelObject()</i> , <i>GetExtendedInfo()</i> , <i>GetInputSpecs()</i> , and <i>GetResultSpecs()</i> are used to create the input and output exchange items a MIKE BASIN model requires and provides.
<i>PerformTimeStep()</i>	The MIKE BASIN model engine uses the <i>SimulateTimeStep()</i> and <i>AdvanceTimeStep()</i> methods to execute the MIKE BASIN model engine for one time step of the simulation period.
<i>GetCurrentTime()</i>	Returns the <i>_dtCurrentDateTime</i> property of the <i>MIKEBASINLinkableComponent</i> class.
<i>GetInputTime()</i>	Returns the <i>_dtCurrentDateTime</i> property of the <i>MIKEBASINLinkableComponent</i> class advanced by the MIKE BASIN model engine <i>TimeStep</i> property in seconds.
<i>OnPrepare()</i>	This method was implemented but had no functionality associated with it.
<i>OnValidate()</i>	This method was implemented but had no functionality associated with it.
<i>Finish()</i>	The <i>FinishSimulation()</i> method of the MIKE BASIN model engine is invoked.

The input items need to implement the *EngineInputItem* abstract class and the output items need to implement the *EngineOutputItem* abstract class for OpenMI 2.0. There are a number of classes that exist that implement the *EngineInputItem* and *EngineOutputItem* which form part of the *Oatc.OpenMI.Wrappers.EngineWrapper* dll. The difference between the classes is the approach taken to get and set the values of the model engine. The following is a list and brief description of these classes:

- *EngineDInputItem* and *EngineDOutputItem*: Use a delegate, which is pointer to a method that does the set and get of the values in the model engine. This pointer can also point to code that does the set and get of the values in a model engine.
- *EngineIInputItem* and *EngineIOutputItem*: Uses an object implementing the *IValueSetter* or *IValueGetter* interface. These interfaces have a *SetValues* and *GetValues* method, which the object must override and provide the code to set and get the values from the engine.
- *EngineEInputItem* and *EngineEOutputItem*: These classes mimic the get and set functionality of OpenMI 1.4. It uses the *LinkableGetSetEngine* abstract class which moves the setting and getting of values from the exchange item to the wrapper or engine. The methods *SetEngineValues* and *GetEngineValues* of the *LinkableGetSetEngine* abstract class need to be overridden and code added to set

and get the values from the model engine. This approach, according to OATC (2010), is not efficient as it causes bottle necks.

The *EngineDInputItem* and *EngineDOutputItem* classes were used in the OpenMI 2.0 wrapper for MIKE BASIN to define the input and output items as objects to get and set model engine values.

A simple OpenMI 2.0 compliant test application, named Test, was also created for use in testing the OpenMI 2.0 MIKE BASIN wrapper. During the process of linking and running the OpenMI 2.0 MIKE BASIN *LinkableComponent* and the OpenMI 2.0 Test *LinkableComponent*, problems arose with the OpenMI 2.0 Configuration Editor. These problems prevented the loading of the OpenMI 2.0 *LinkableComponents*, the creation of links between the OpenMI 2.0 *LinkableComponents* and the execution of the linked composition of OpenMI 2.0 *LinkableComponents*. These problems and the fixes are detailed below.

Initially the OpenMI 2.0 MIKE BASIN *LinkableComponent* did not load into the OpenMI 2.0 Configuration Editor. The documentation and examples provided by the OpenMI Association did not mention that the *Caption* string variable of the *LinkableEngine* interface needs to be set in the *Initialize* method to load a *LinkableComponent*. This was discovered through debugging the source code of the OpenMI 2.0 Configuration Editor. Once the *Caption* string variable of the *LinkableEngine* interface was assigned a value, the MIKE BASIN *LinkableComponent* loaded successfully in the OpenMI 2.0 Configuration Editor.

It is important to note that *TortoiseSVN* client software, used to access the OpenMI 2.0 source code repository, uses a *Revision* number for each file to keep track of the version of files. If a change is made to a file and committed to the server working code, the file receives a new incremented *Revision* number. This ensures the file being worked on a client machine is the latest updated code. The *Revision* number of files changed to solve the main problems or issues will be provided below. It is also important to note that the *root* folder described in the rest of this document refers to the folder on the development machine where the working code has been downloaded using the *TortoiseSVN* client software.

The first problem experienced was that a link could not be correctly added between any two *LinkableComponents* as the OpenMI 2.0 Configuration Editor tried to add the link twice. The problem was traced to the *Output* class of the *Output.cs* file contained in the *Oatc.OpenMI.Sdk.Backbone* namespace. The *Output.cs* file forms part of the

*Oatc.OpenMI.Sdk* project and can be found in the *root\src\csharp\Oatc.OpenMI\Sdk\Backbone\* directory. The *Revision* number for the file was 1640. There was no update of the *Output.cs* file at the time of writing of this document. To rectify the problem code was commented out from the *AddConsumer(IBaseInput consumer)* method of the *Output.cs* file to prevent the OpenMI 2.0 Configuration Editor from adding the link twice, and new code was added.

The second problem encountered was that the linked *LinkableComponents* could not run within the OpenMI 2.0 Configuration Editor. The OpenMI 2.0 Configuration Editor was using threads to run the linked *LinkableComponents*. The threaded approach could not use a method defined for an interface within the MIKE BASIN *LinkableComponent* across different threads, which prevented the linked *LinkableComponents* from running. An attempt was made to change the code of the MIKE BASIN *LinkableComponent* to get the linked *LinkableComponents* to run using threads, without changing the code of the OpenMI 2.0 Configuration Editor, but this was not successful. By studying the code of the OpenMI 2.0 Configuration Editor a method called *Run* was found to run the linked *LinkableComponents* without using threads. The *Run* method can be found in the *CompositionRun* class, falling under the *Oatc.OpenMI.Gui.Core* namespace. The *CompositionRun* class is from the *Oatc.OpenMI.Gui.Core* project. The *CompositionRun.cs* file can be found in the *root\src\csharp\Oatc.OpenMI\Gui\Core* directory. The code change was made to the *Run* class of the *Run.cs* file of the *Oatc.OpenMI.Gui.ConfigurationEditor* namespace. The *Run.cs* file forms part of the *Oatc.OpenMI.Gui.ConfigurationEditor* project and can be found in the *root\src\csharp\Oatc.OpenMI\Gui\ConfigurationEditor\* directory. There was no update of the *Run.cs* file at the time of writing of this document.

The third problem encountered was that the linked *LinkableComponents* would not stop running. The linked *LinkableComponents* ran beyond the simulation end date and time. It was discovered through debugging of the *Oatc.OpenMI.Gui.ConfigurationEditor* project working code, that the *DoRun* method contained in the *CompositionRun* class of the *Oatc.OpenMI.Gui.Core* namespace did not check for a status *Done*, which indicates a *LinkableComponent* has completed its run. There was no update of the *CompositionRun.cs* file at the time of writing of this document

Some of these problems discussed were solved by changing the code of the OpenMI 2.0 environment but were done without exploring the consequences these changes would have on the rest of the functionality within the OpenMI 2.0 environment. The changes could possibly affect the results produced. Problems continued to arise during the creation of the

OpenMI compliant wrapper and it was decided to use the OpenMI 1.4 SDK, as it is a released version that has been tested extensively to eliminate any bugs and would require no changes to its code. The OpenMI 2.0 compliant wrapper was successfully created for MIKE BASIN, but due the problems experienced with OpenMI 2.0 Configuration Editor, further implementation, testing and use did not take place.

## **5.2 Development of OpenMI Wrapper for *ACRU***

The *ACRU* model is not OpenMI compliant, but refinements to the model in this project, described in Chapter 4, made it suitable to be made OpenMI compliant, either by direct implementation of the OpenMI Standard or by means of a wrapper. The *ACRU* model is written in the Java programming language and the necessity for *ACRU* to be made both Java and .Net OpenMI compliant was an important criteria.

The OpenMI Standard is duplicated in Java and .Net versions and each of these is supported by a corresponding OpenMI Software Development Kit (SDK) containing a default implementation of the OpenMI Standard interfaces and other helper classes. The OpenMI 2.0 version of the OpenMI Standard was recently released to provide additional user requirements not met by the OpenMI 1.4 version. It was decided that in this project the *ACRU* model would be made OpenMI 1.4 compliant for the following reasons: (i) the OpenMI 2.0 SDK for Java is still under development and has not been released, (ii) the problems experienced with the OpenMI 2.0 SDK for .Net when creating and using an OpenMI 2.0 wrapper for MIKE BASIN indicate that the supporting tools for the OpenMI 2.0 version may not be mature, and (iii) all the models registered on the OpenMI Association website as being OpenMI compliant are currently only OpenMI 1.4 compliant.

It was decided that *ACRU* should be made OpenMI compliant by means of a wrapper instead of direct implementation of the OpenMI Standard for the following reasons: (i) the wrapper option is easier and takes advantage of functionality already coded into the classes provided in the SDK's *nl.alterra.openmi.sdk.wrapper* package, and (ii) wrapping would enable both OpenMI 1.4 and OpenMI 2.0 compliant versions of *ACRU* to be provided at some point in the future without changing the *ACRU* code and without the two versions potentially conflicting with each other.

## 5.2.1 OpenMI 1.4 Java wrapper

An OpenMI 1.4 compliant Java wrapper was created for the *ACRU* model through two classes, the *ACRU.OpenMI.AcruEngineWrapper* class which implements the *nl.alterra.openmi.sdk.wrapper.IEngine* interface and the *ACRU.OpenMI.AcruLinkableComponent* class which extends the *nl.alterra.openmi.sdk.wrapper.LinkableEngine* class.

The methods in the *ACRU* engine that were used to implement the methods of the *IEngine* abstract class in the *AcruEngineWrapper* class are detailed in Table 5.5

Table 5.5 Implementation of the *IEngine* interface methods in the *AcruEngineWrapper* class

Method Name	Description of Method Implementation
<i>getModelID()</i>	Calls the <i>MAcruXml.getModelID()</i> method which returns the model run ID if it exists
<i>getModelDescription()</i>	Calls the <i>MAcruXml.getModelDescription()</i> method which returns the model run description if it exists or a string containing the model input file name, the scenario set selected for the run and the simulation start and end dates.
<i>getComponentID()</i>	Returns "ACRU" as the component ID.
<i>getComponentDescription()</i>	Returns "ACRU daily physical conceptual agrohydrological model" as the component ID.
<i>getTimeHorizon()</i>	Returns a time span starting with the date/time returned by the <i>MAcruXML.getSimulationStartDate()</i> method and ending with the date/time returned by the <i>MAcruXML.getSimulationEndDate()</i> method
<i>getInputExchangeItemCount()</i>	Returns the size of the <i>inputExchangeItems</i> ArrayList of input exchange items defined in the <i>AcruEngineWrapper</i> class.
<i>getOutputExchangeItemCount()</i>	Returns the size of the <i>outputExchangeItems</i> ArrayList of output exchange items defined in the <i>AcruEngineWrapper</i> class.
<i>getInputExchangeItem(int exchangeItemIndex)</i>	Returns the input exchange item at the specified index in the <i>inputExchangeItems</i> ArrayList defined in the <i>AcruEngineWrapper</i> class.
<i>getOutputExchangeItem(int exchangeItemIndex)</i>	Returns the output exchange item at the specified index in the <i>outputExchangeItems</i> ArrayList defined in the <i>AcruEngineWrapper</i> class.

Table 5.5 (continued) Implementation of the *IEngine* interface methods in the *AcruEngineWrapper* class

Method Name	Description of Method Implementation
<i>initialize(HashMap properties)</i>	Uses the model initialisation arguments specified in <i>properties</i> to generate an array of model initialisation arguments required by the <i>ACRU</i> model engine, which is used to create a new instance of <i>MAcruXml</i> . The <i>MAcruXml.initialiseModel()</i> method is then called to initialise the new model. The new model is then queried to find all instances of <i>CComponent</i> and all the instances of <i>DData</i> belonging to each <i>CComponent</i> . For each instance of <i>DData</i> an input exchange item and/or output exchange item is created depending on whether the instance of <i>DData</i> is an input, output or state variable.
<i>performTimeStep()</i>	The <i>MAcruXml.runTimeStep()</i> method is called to execute the <i>ACRU</i> model engine for one time step.
<i>getCurrentTime()</i>	Calls the <i>MAcruXml.getCurrentDate()</i> method which returns the current date of the simulation.
<i>getInputTime(String QuantityID, String ElementSetID)</i>	Returns the date time of the next day after the date/time returned by the <i>MAcruXml.getCurrentDate()</i> method.
<i>getEarliestNeededTime()</i>	Returns the date time of the previous day before the date/time returned by the <i>MAcruXml.getCurrentDate()</i> method.
<i>setValues(String QuantityID, String ElementSetID, IValueSet values)</i>	Uses the <i>ElementSetID</i> parameter to find the relevant instance of <i>CComponent</i> and the <i>QuantityID</i> parameter to find the relevant instance of <i>DData</i> . If the instance of <i>DData</i> is a state variable then the <i>DData.setCurrentState(...)</i> method is called, otherwise the <i>DData.setData(...)</i> method is called to set the value specified in the values parameter.
<i>getValues(String QuantityID, String ElementSetID)</i>	Uses the <i>ElementSetID</i> parameter to find the relevant instance of <i>CComponent</i> and the <i>QuantityID</i> parameter to find the relevant instance of <i>DData</i> . If the instance of <i>DData</i> is a state variable then the <i>DData.getCurrentState()</i> method is called, otherwise the <i>DData.setData(Date dateTime)</i> method is called to return a value for the end of the previous simulation day.
<i>GetMissingValueDefinition()</i>	Returns a hard-coded value of "-999.9".
<i>Finish()</i>	Calls the <i>MAcruXml.finaliseModel()</i> method
<i>Dispose()</i>	Disposes of the instance of <i>MAcruXml</i> by setting it to null.

One potential problem that had to be overcome was the manner in which units of measure are specified for variables representing fluxes in *ACRU*. For example, in *ACRU* the units of measure associated with streamflow are cubic metres, and not cubic metres per day. This makes sense in *ACRU* as it is a daily model and so all fluxes are implicitly per day. This also means that the units of measure do not need to change when a flux is temporally aggregated, for example, to annual streamflow. However, a model such as MIKE BASIN can be run for different time step lengths and fluxes may be specified to have units of

measure which include the time period, for example, cubic metres per second. This was resolved by further developing the OpenMI wrapper to provide duplicate input and output exchange items for each flux variable, one as a quantity and one as a rate.

To test the OpenMI wrapper for *ACRU*, an OpenMI 1.4 compliant Java wrapper was created for the *DataReaderWriter\_AcruCSV* class which is part of the *ModelDataAccess* package developed as part of the *ACRU* modelling system. The *DataReaderWriter\_AcruCSV* class reads and writes time series data in a comma separated value (CSV) file designed to provide time series input data to the *ACRU* model. The development of this wrapper not only provided a means of testing the OpenMI wrapper developed for the *ACRU* model, but also demonstrated the versatility of the OpenMI standard and the benefit of OpenMI compliance in making models and modelling tool interoperable. Tests were then run using a simple test catchment for the following scenarios: (i) CSV file as input to *ACRU*, (ii) *ACRU* output to CSV file, and (iii) one *ACRU* model flowing into another *ACRU* model. As the OpenMI Association does not provide a Java version of the Configuration Editor graphical user interface shown in Figure 5.5, some custom code was written using classes provided in the OpenMI 1.4 SDK to enable an OpenMI configuration to be configured and run. These tests were all successfully completed, providing confidence that the wrapper was working as expected. The development of an OpenMI wrapper for *ACRU* was surprisingly easy where this was partly due to the wrapping tools provided by the OpenMI Association and partly due to the changes made to the *ACRU* model reported in Chapter 4.

### **5.2.2 OpenMI 1.4 .Net wrapper**

The *ACRU* model is written in the Java programming language and has been made OpenMI 1.4 Java compliant. However, all the OpenMI compliant models listed on the OpenMI Association website are OpenMI 1.4 .Net compliant, which emphasised the need to develop an OpenMI 1.4 .Net compliant wrapper for *ACRU*.

There were two possible approaches to overcoming the compatibility problem between the Java and .Net platforms, (i) to use a Java-.Net bridge that makes use of the Java Native Interface (JNI) for Java, and (ii) to compile the *ACRU* Java code into a .Net assembly. The second approach was selected as this approach was expected to offer better model run speeds, though with the disadvantage of having to recreate the .Net assembly every time a change is made to the *ACRU* model. The IKVM.NET (<http://www.ikvm.net/>) tool was used to statically compile the *ACRU* model code written in Java to a .Net assembly named *ACRU.dll*.



The *ACRU.OpenMI.AcruEngineWrapper* and *ACRU.OpenMI.AcruLinkableComponent* classes were then duplicated in .Net using the C# programming language to create an OpenMI 1.4 .Net wrapper for *ACRU*.

A similar procedure to that used for *ACRU* was used to create a .Net assembly for the *ModelDataAccess* package and duplicate the Java wrapper for the *DataReaderWriter\_AcruCSV* class in .Net. Tests were then run for a simple test catchment, using the Configuration Editor shown in Figure 5.5, for the following scenarios: (i) CSV file as input to *ACRU*, (ii) *ACRU* output to CSV file, (iii) one *ACRU* model flowing into another *ACRU* model, and (iv) *ACRU* model flowing into MIKE BASIN model. These tests were all successfully completed.

### **5.3 Results and Recommendation**

An OpenMI 1.4 compliant wrapper for MIKE BASIN was successfully implemented. Tests were carried that verified the OpenMI 1.4 compliant wrapper worked as expected. Data operations applied to output from a *LinkableComponent* were also successfully implemented and tested.

Though the initial implementation of the OpenMI 2.0 compliant wrapper for MIKE BASIN was successful, problems with the OpenMI 2.0 Configuration Editor prevented further development and application of this wrapper. If the OpenMI Association releases a stable version of OpenMI 2.0 SDK it would be recommended to use the OpenMI 2.0 compliant wrapper for MIKE BASIN, as the OpenMI 2.0 Standard is an improvement to the OpenMI 1.4 Standard.

An OpenMI 1.4 Java and OpenMI 1.4 .Net compliant wrappers were also successfully implemented for the *ACRU* model. Tests indicated that the wrappers were working as expected and the *ACRU* and MIKE BASIN models were successfully linked using OpenMI for a simple test catchment.

## 6 USE CASES FOR THE LINKED MODELS

DJ Clark and SLC Thornton-Dibb

Modellers often try to simplify complex systems by breaking them up into more manageable components, but sometimes by doing this, important interrelationships between the modelled components are lost. Integrated water resource management requires a more holistic assessment of entire water resource systems, including surface water, groundwater, economics and social impacts. This requires integrated water resource modelling. One approach to integrated modelling would be to develop a big model to do everything, but this approach has high development and maintenance costs and does not take advantages of using existing models for the various components of a water resources systems. A more practical approach is to integrate several existing models, saving costs and enabling different combinations of models to be integrated for each unique modelling exercise. There are two main reasons for integrating models: (i) to gain functionality not available in an individual model and (ii) to model feedbacks between the system components represented by the individual models to better represent the system being modelled. If there are no feedbacks between the systems being represented by each individual model, then the models can be integrated using simple series links, otherwise a model linking mechanism such as OpenMI selected for this project can be used to link models in parallel. Parallel linking involves running one model for a single time step, then using the output from this model as input to another model which is run for a single time step, and this process is repeated for individual time steps until the end of the simulation time period. The models may have different time steps and the links between them may be uni-directional or bi-directional.

The *ACRU* model has the following limitations which necessitate linking it to another model such as MIKE BASIN:

- Flow routing, the *ACRU* 3.00 model included flow routing but this was slow and difficult to apply, and the flow routing has not yet been included in the *ACRUXml* version.
- Multiple water users can use one source, but a single user cannot have multiple water sources.
- *ACRU* does not explicitly include water users other than irrigation, though lumped dam and river in-transfer and out-transfer quantities can be specified to represent other water users.

- Water allocation is by means of a relatively simple priority system, and at this point there is no provision for other allocation methods, such as Fractional Water Allocation and Capacity Sharing (FWACS).
- No GIS user interface is available to configure and view river/dam flow networks.
- No optimization of flow networks is possible.
- The simple ground water model does not handle complex groundwater interactions.
- The *ACRU\_NP* water quality module only deals with the terrestrial part of the water system and does not route water quality constituents through the river network.

The MIKE BASIN requires the following inputs which could be provided by the *ACRU* model:

- Streamflow simulated by the *ACRU* daily timestep agrohydrological model, which has been extensively applied in South Africa, and for which default input data sets are available at a quinary catchment scale for the whole of South Africa.
- Irrigation water requirement and return flow quantities, though these can also be modelled in MIKE BASIN.
- Groundwater recharge quantities.
- Salinity, nitrogen and phosphorus loads leaving the terrestrial hydrological system.

Prior to this project, the *ACRU* and MIKE BASIN models have been integrated by means of simple series links, where *ACRU* generates a streamflow time series  $n$  years in length which are then translated and used as input to MIKE BASIN, as shown in the use case presented in Figure 6.1, or using custom code as shown in the use case presented in Figure 6.2. This approach works well if there are no feedbacks between the terrestrial hydrological system being modelled by *ACRU* and the river network system being modelled by MIKE BASIN. However, a common example of a feedback between these two systems is irrigation, where the irrigation requirement is calculated using the hydrological model, this requirement is provided by the river network model, and then applied to the irrigated fields in the hydrological model, which will calculate return flows which then need to be fed back to the river network model. This particular feedback problem was recognized, and resulted in the developers of MIKE BASIN recently including an irrigation module into the MIKE BASIN model, so that the feedback was dealt with within the model.

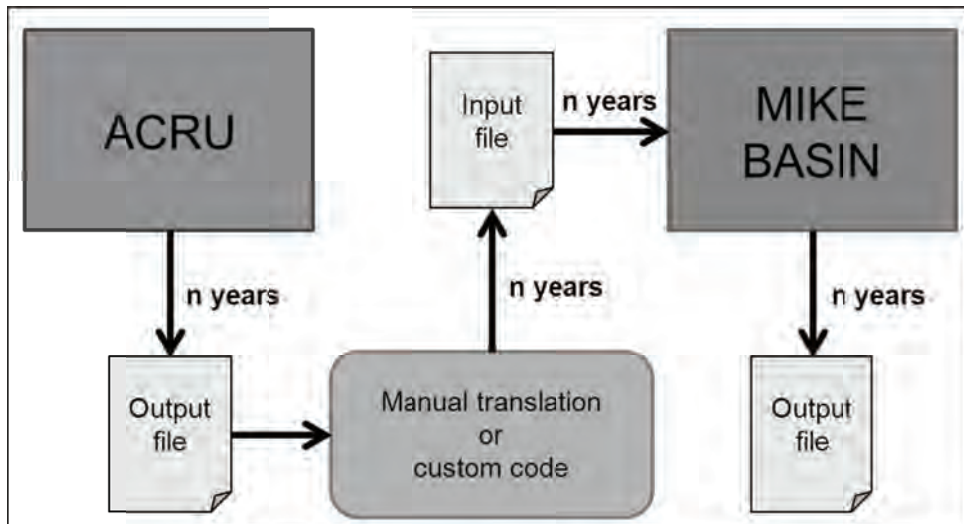


Figure 6.1 *Status quo* simple series link by file translation

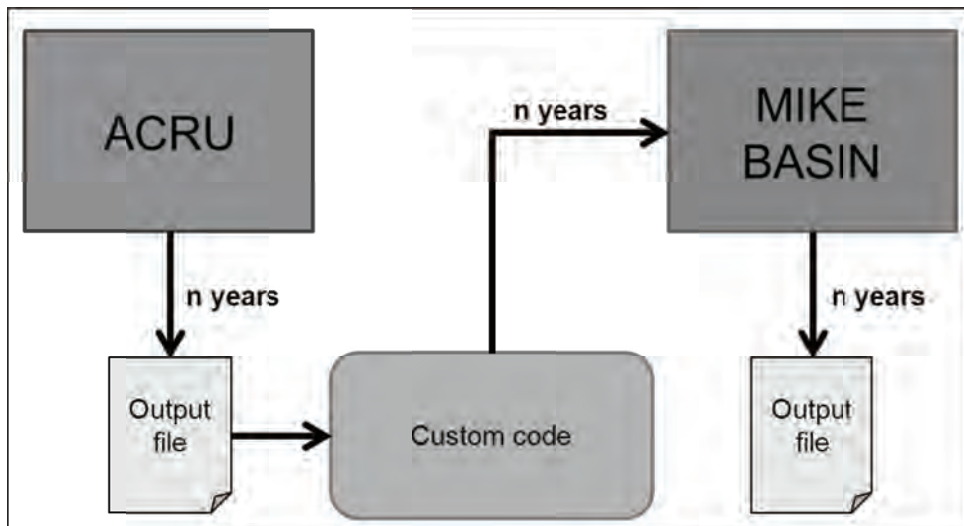


Figure 6.2 *Status quo* simple series link by custom code

The advantages of using OpenMI are that it:

- Saves manual translation between model output and input file formats.
- Saves developing custom code to link models, any OpenMI compliant model and can be linked to any other OpenMI compliant model.
- Enable chains of models, e.g. one-to-many and many-to-one, which would be difficult in custom code which is usually one-to-one.
- Enables linking models with different spatial and temporal resolutions.
- Enables feedbacks to be modelled.

The general concepts of linking two models such as *ACRU* and MIKE BASIN using OpenMI is shown in Figure 6.3. In simple terms, when a simulation is triggered, one model, for example MIKE BASIN, will request an input value for the first time step from the other model, for example *ACRU*. *ACRU* will then run for the first time step and provide the required input value to MIKE BASIN which will then also run for one time step. This sequence is then repeated for successive time steps until the end of the simulation period.

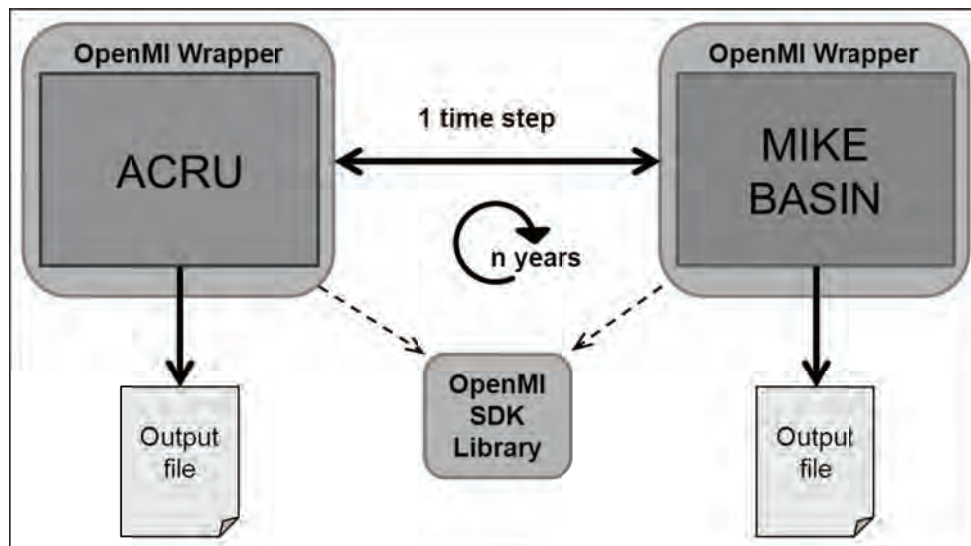


Figure 6.3 Linking *ACRU* and MIKE BASIN using OpenMI

Having successfully created OpenMI compliant wrappers for both *ACRU* and MIKE BASIN as described in Chapter 5, the next step was to define a set of use cases to demonstrate how these two models could be linked using OpenMI for a range of modelling scenarios. For each use case a diagram is provided to help illustrate how the models are linked, and an example containing a key explaining these diagrams is shown in Figure 6.4. In each use case, *ACRU* component-variable to MIKE BASIN component-variable pairs are specified to assist future users in setting up model links. In some use cases there may be more than one way to configure the model links and it is up to the user to decide which is best for their particular application depending on how the individual models have been set up. Though OpenMI makes linking models easier, users will still require a thorough conceptual understanding of both models in order to link them correctly. In these use cases it is assumed that both the *ACRU* and MIKE BASIN models are being run at a time step of one day.

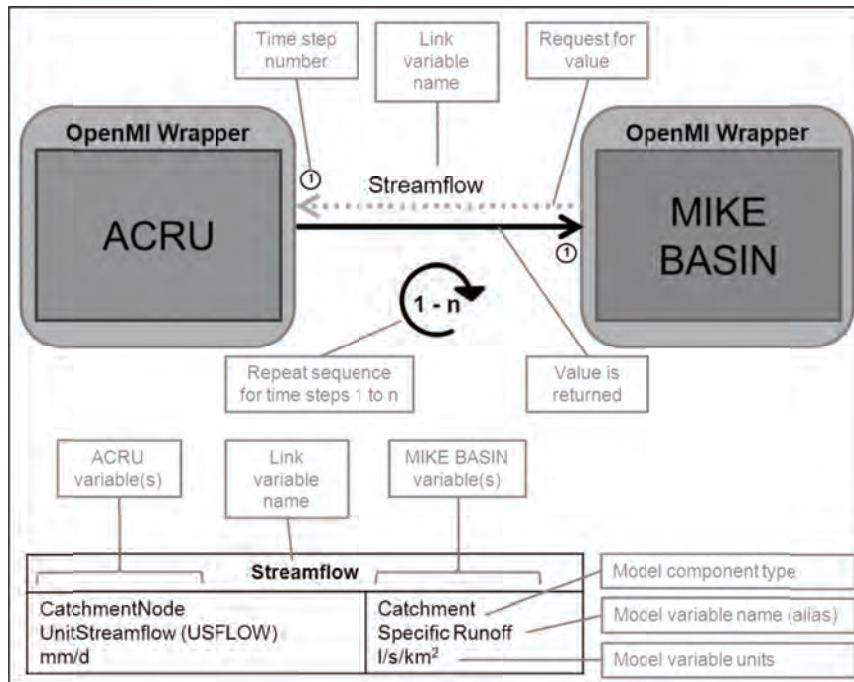


Figure 6.4 Key to the use case diagrams used in this chapter

## 6.1 Streamflow Links

The simplest use case is a uni-directional link where *ACRU* provides streamflow values to MIKE BASIN as shown in Figure 6.5. In this use case, streamflow includes both surface runoff and baseflow and is represented as a normalised depth of flow for a specified time period. At the start of the first time step MIKE BASIN requests streamflow values from *ACRU*, *ACRU* will then run for the first time step and return the streamflow values to MIKE BASIN which will then run for the first time step. This sequence of events will be repeated for the second time step and all subsequent time steps until the end of the simulation. This can already be done with series linking but is now much easier as it requires neither manual translation of streamflow data between the models nor custom code. The results from the linked models should be identical to obtained from a simple series link. For this use case, irrigation would still need to be done completely within MIKE BASIN as only streamflow values are exchanged between the models. Care should be taken when using this use case and modelling riparian zones or wetlands in *ACRU*, the streamflow values passed to MIKE BASIN must be downstream of the wetland or riparian zone being modelled as these have processes which model feedbacks within the flow network modelled in *ACRU*.

The MIKE BASIN model usually receives streamflow inputs to the *Specific Runoff* variable of a Catchment component by means of a DFS0 file. In this case these values are overwritten by streamflow values calculated in *ACRU*. The units of measure for the values exchanged

between the models for an individual link do not need to be the same as long as they have the same dimension, in this case length per unit time, as OpenMI takes care of the unit conversion. It is recommended that the *ACRU UnitRunoff* variable be used as the source of streamflow. If the *ACRU UnitStreamflow* variable for *CatchmentNode* or *SubCatchmentNode* is used then the user needs to make sure that flows are not being accumulated in *ACRU*'s own internal river network.

This use case enables the provision of additional modelling functionality some of which would not be available from the *ACRU* model on its own.

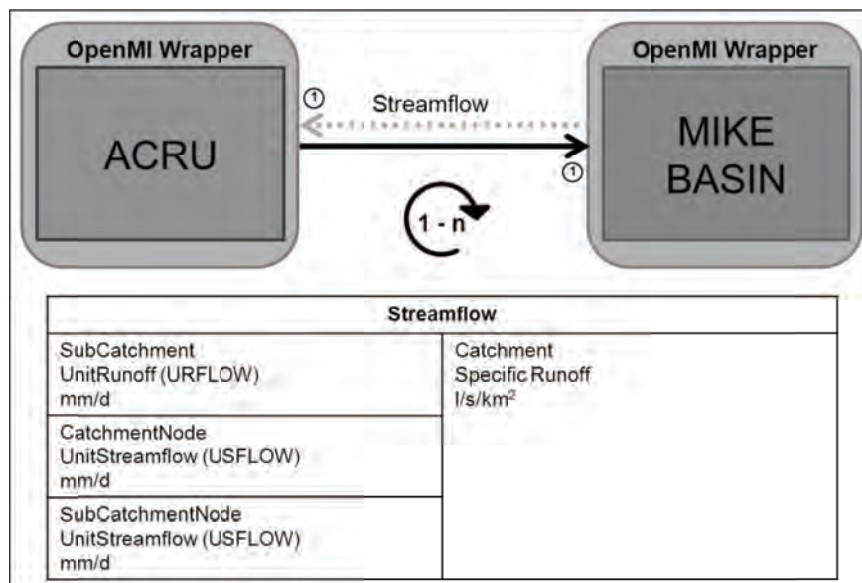


Figure 6.5 Streamflow use case

### 6.1.1 Water users and return flows

As stated in the introduction to this chapter *ACRU* does not explicitly enable individual non-irrigation water users such as industrial and municipal water users to be represented, although lumped dam and river in-transfer and out-transfer quantities can be specified. The *ACRU* model permits one or more *DamPumpInNode*, *DamDraftNode*, *DomesticAbstractionNode*, and *ExternalWaterSourceNode* to be added to each *SubCatchment* component, though in-transfers in the sense of return flows are not linked to the out-transfers. In *MIKE BASIN* Water User components can easily be added using the GIS graphical user interface to represent individual or lumped water users. For each Water User component the user may specify the water source, a time series of requested water quantities, a return flow destination and a time series of return flow fractions.

### **6.1.2 Multiple water sources for a user**

The *ACRU* model allows multiple water users to request water from a single water source, but does not allow a water user to request water from multiple water sources due to its relatively simple water allocation rules. However, MIKE BASIN does permit multiple water sources to be assigned to an individual water user and these water sources may be rivers, reservoirs or groundwater.

### **6.1.3 Water allocation options**

The *ACRU* model has a relatively simple priority based water allocation system. In *ACRU* a water user sends a water request to its single water source specifying the water supply path, water ownership and a request priority. Near the end of each time step all the water requests collected at a water source for the time step are evaluated and available water is allocated. If there are several requests with the same priority then this group of requests will either be met in full or, if there is insufficient water, this this group will share the available water in proportion to their requested quantity. The MIKE BASIN model provides the option to use either priority or FWACS allocation. The facility to track water ownership in *ACRU* does enable water in rivers and reservoirs to be reserved for a water owner, and would thus enable FWACS allocation to be added. However, water allocation and river network modelling are not the main focus of *ACRU*, and this an example of integration enabling the use of each model for its strengths.

### **6.1.4 Flow routing**

Flow routing is an important feature for any modelling system that is to be used for water resource operations modelling and management, as for short term operational decisions, with critical durations of the order of days or even hours, the timing of flows is critical in determining availability. Flow routing is another example of where integrating the models enables MIKE BASIN to provide functionality in an area that is not a main focus of *ACRU*. The *ACRU* 3.00 and *ACRU2000* versions included flow routing, but this was slow and difficult to apply. Flow routing has not yet been included in the *ACRUXml* version, though it is anticipated that the changes to the component configuration and the internal data structures described in Section 4.3, will facilitate easier implementation of flow routing. The GIS user interface for MIKE BASIN lends itself to setting up flow networks and the input parameters required for flow routing.



### 6.1.5 Scenarios

The ability to set up and easily run different scenarios for water resources planning and operations was one of the requirements noted at the beginning of the project. Changes made to the *ACRU* model input files and the model itself, as explained in Chapter 4, have made it easy to configure and run scenarios in *ACRU*. The MIKE BASIN model does not offer similar functionality for scenarios, beyond enabling users to save model outputs for scenarios in sets named by the user and enabling these to be analysed. If the configuration of the river network modelled by MIKE BASIN does not change with time, then the integrated models can be used to evaluate different land use scenarios and their effect on availability of water at different locations in the river network.

## 6.2 Streamflow and Groundwater Links

Another relatively simple use case includes a uni-directional link where *ACRU* provides separate surface runoff (termed “quickflow” in *ACRU*) and groundwater recharge values to MIKE BASIN as shown in Figure 6.6 and Figure 6.7. In this use case, MIKE BASIN will do the groundwater modelling, which will require that this option be turned on in MIKE BASIN and the necessary groundwater modelling variables be specified. In the simple streamflow use case described in Section 6.1, streamflow included both surface runoff and baseflow. In this use case, *ACRU* will supply separate surface runoff and groundwater recharge inputs to MIKE BASIN at each time step.

At the start of the first time step MIKE BASIN requests surface runoff and groundwater recharge values from *ACRU*. *ACRU* will then run for the first time step and return the surface runoff and groundwater recharge values to MIKE BASIN which will then run for the first time step. This sequence of events will be repeated for the second time step and all subsequent time steps until the end of the simulation. This can already be done with series linking but is now much easier as it requires neither manual translation of streamflow data between the models nor custom code. The results from the linked models should be identical to obtained from a simple series link. For this use case irrigation would still need to be done completely within MIKE BASIN.

The MIKE BASIN model usually receives surface runoff inputs to the *Specific Runoff* variable and groundwater recharge inputs to the *Specific Recharge* variable of a Catchment component by means of DFS0 files. In this case the values for these variables are

overwritten by surface runoff and groundwater recharge values calculated in *ACRU*. It is important to note that *ACRU* does not include an output of groundwater recharge as a normalised depth of flow for a specified time period at a catchment or subcatchment level, however there are two ways that this use case can be achieved in spite of this. The first and simplest method shown in Figure 6.6 is to use the *SubCatchment UnitBaseflow* output variable and set the coefficient of baseflow response *CoefBaseflowResp* (*COFRU*) to one for all instances of *HRU*, *RiparianZone* and *Wetland*. With this first method it is important to bear in mind that streamflow values calculated in *ACRU* will be affected by the baseflow workaround. In *ACRU* the coefficient of baseflow response for irrigated areas is set to 0.02 and can't be adjusted, but this is not important for this use case as irrigated areas are modelled in MIKE BASIN. The second method shown in Figure 6.7 requires creating a *Catchment* in MIKE BASIN for every *HRU*, *RiparianZone*, *Wetland*, *IrrigatedArea* and *AdjunctImperviousArea* in *ACRU*, enabling the groundwater recharge variables *SaturatedFlow* (*SUR*) for these components to be linked to the *Specific Recharge* variables in the corresponding MIKE BASIN *Catchments*. For this second method the *UnitQuickflow* (*UQFLOW*) variable for each *HRU*, *RiparianZone*, *Wetland*, *IrrigatedArea* and *AdjunctImperviousArea* instead of the *UnitQuickflow* (*UQFLOW*) variable for each *SubCatchment* as a whole. For this second method to work the *UnsaturatedRedistributionOption* (*IUNSAT*) variable for the *HRU/Soil*, *RiparianZone/Soil*, *Wetland/Soil* components would have to be turned off.

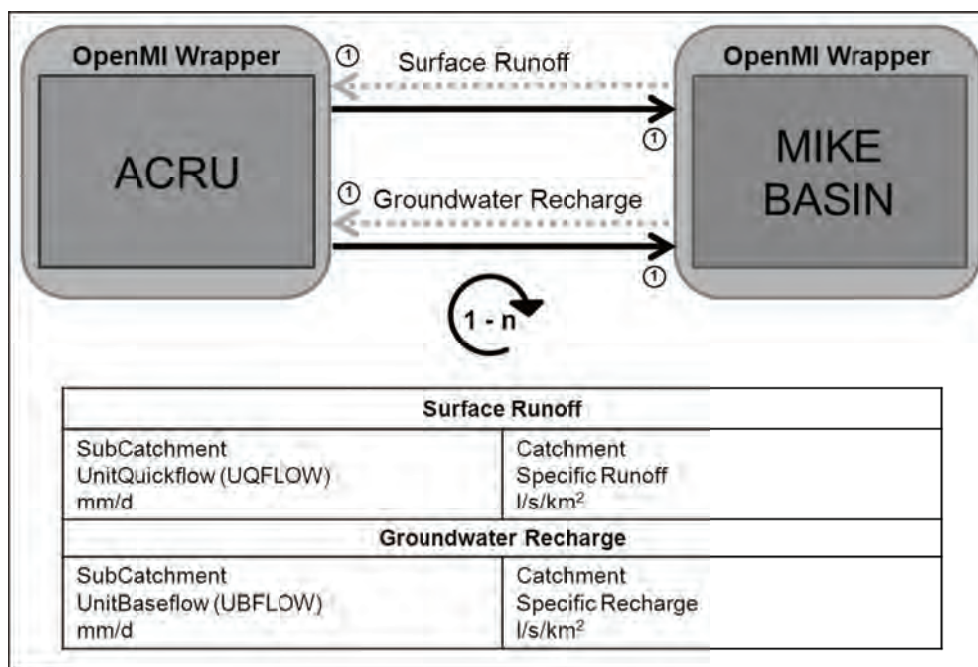


Figure 6.6 Surface runoff and groundwater recharge use case using the first method

This use case enables the provision of the same additional modelling functionality discussed in Section 6.1 for the simple streamflow use case. In addition this use case enables the water users to use groundwater as a water source and for river-groundwater seepage-recharge processes to be modelled, neither of which are currently possible in *ACRU*.

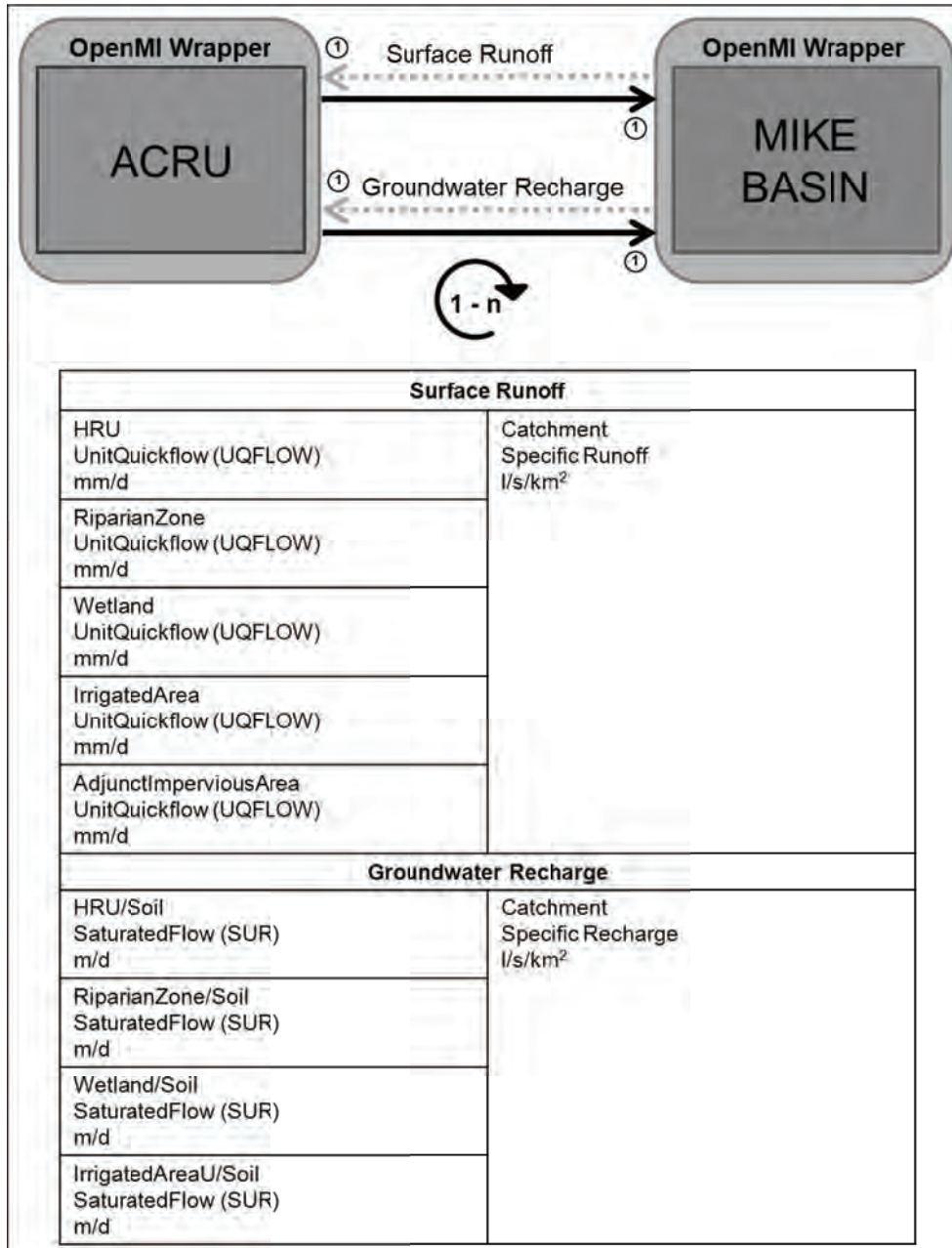


Figure 6.7 Surface runoff and groundwater recharge use case using the second method

### 6.3 Irrigation Requirement and Supply Links

A more complicated use case includes a bi-directional link where *ACRU* provides streamflow and irrigation requirement values to MIKE BASIN, and MIKE BASIN provides an irrigation supply value to *ACRU*, as shown in Figure 6.8. The purpose of this use case is to demonstrate modelling feedbacks using a bi-directional link. In this use case, *ACRU* will do the irrigation modelling and will send a request for irrigation water to MIKE BASIN, which will evaluate this request along with those of other water users and attempt to provide the requested irrigation water, which will be applied to the irrigated area in *ACRU*. Both *ACRU* and MIKE BASIN can model irrigation, but for the purpose of this use case it is assumed that MIKE BASIN does not model irrigation. For this use case the *ACRU* model should be set up such that each IrrigatedArea has an ExternalWaterSourceNode associated with it as the supplier of water to the IrrigationSystem using a WaterSourceDest relationship. In MIKE BASIN the irrigated area should be represented as Water User extracting water from the relevant River Node, Catchment Node or Reservoir.

At the start of the first time step MIKE BASIN requests streamflow and irrigation requirement values from *ACRU*. *ACRU* will then prepare to run for the first time step by requesting an irrigation supply amount for the first time step. MIKE BASIN has not yet run for the first time step as it is still waiting for a reply to its request to *ACRU*, but attempts to return a value to *ACRU* which in this case is zero. *ACRU* will then run for the first time step without applying any irrigation and return the streamflow and irrigation requirement values to MIKE BASIN which will then run for the first time step. This sequence of events will be repeated for the second time step, but this time MIKE BASIN will be able to provide an irrigation supply amount calculated in the first time step, which will be applied in *ACRU* when it runs for its second time step. When *ACRU* runs its second time step any return flows that may occur will be included in the streamflow values that are provided to MIKE BASIN for its second time step. This sequence of events is repeated for all subsequent time steps until the end of the simulation. This scenario where irrigation is applied and return flows are generated for the time step after the time step in which the irrigation supply is calculated is a common scenario that exists internally with *ACRU* when it runs as a standalone model and probably also occurs in MIKE BASIN. This scenario occurs due to the feedback loop, but is still a good modelling solution as feedbacks where irrigation return flows enter the river network upstream of the source can still be represented.

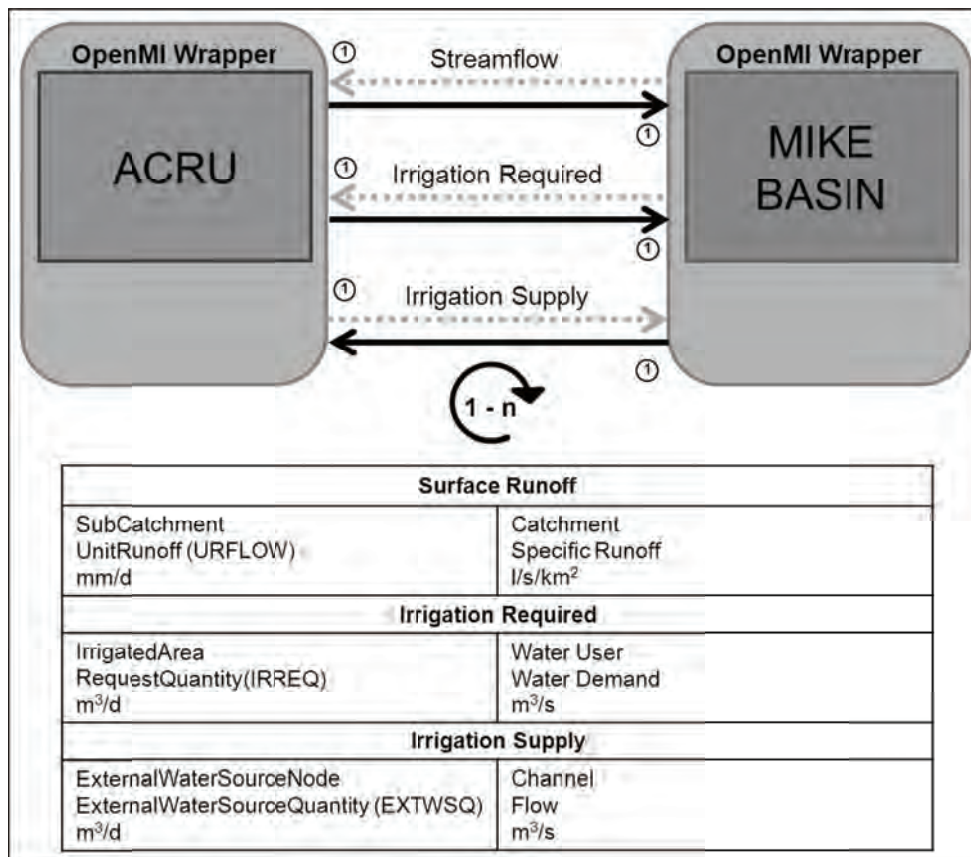


Figure 6.8 Streamflow, irrigation requirement and supply use case

#### 6.4 Water Quality Links

As an extension to the groundwater use case presented in Section 6.2, this use case includes a uni-directional link where *ACRU* provides separate surface runoff (termed quickflow in *ACRU*) and groundwater recharge values to *MIKE BASIN*, and in addition provides water quality loads (such as ammonium) as shown in Figure 6.9. In this use case surface runoff and groundwater recharge water quantity and ammonium loads are modelled in *ACRU* and then provided to *MIKE BASIN* which models flows and water quality in the river network. For this use case it would be necessary to set up *MIKE BASIN* as described for the second method in Section 6.2 which requires creating a *Catchment* in *MIKE BASIN* for every *HRU*, *RiparianZone*, *Wetland*, *IrrigatedArea* and *AdjunctImperviousArea* in *ACRU*. For this use case irrigation would still need to be done completely within *MIKE BASIN*. Though the *ACRUSalinity* and *ACRU\_NP* water quality module have yet to be updated for use in the *ACRUXml* version, this use case serves to demonstrate how water quality could be modelled by the integrated models once these modules have been updated.

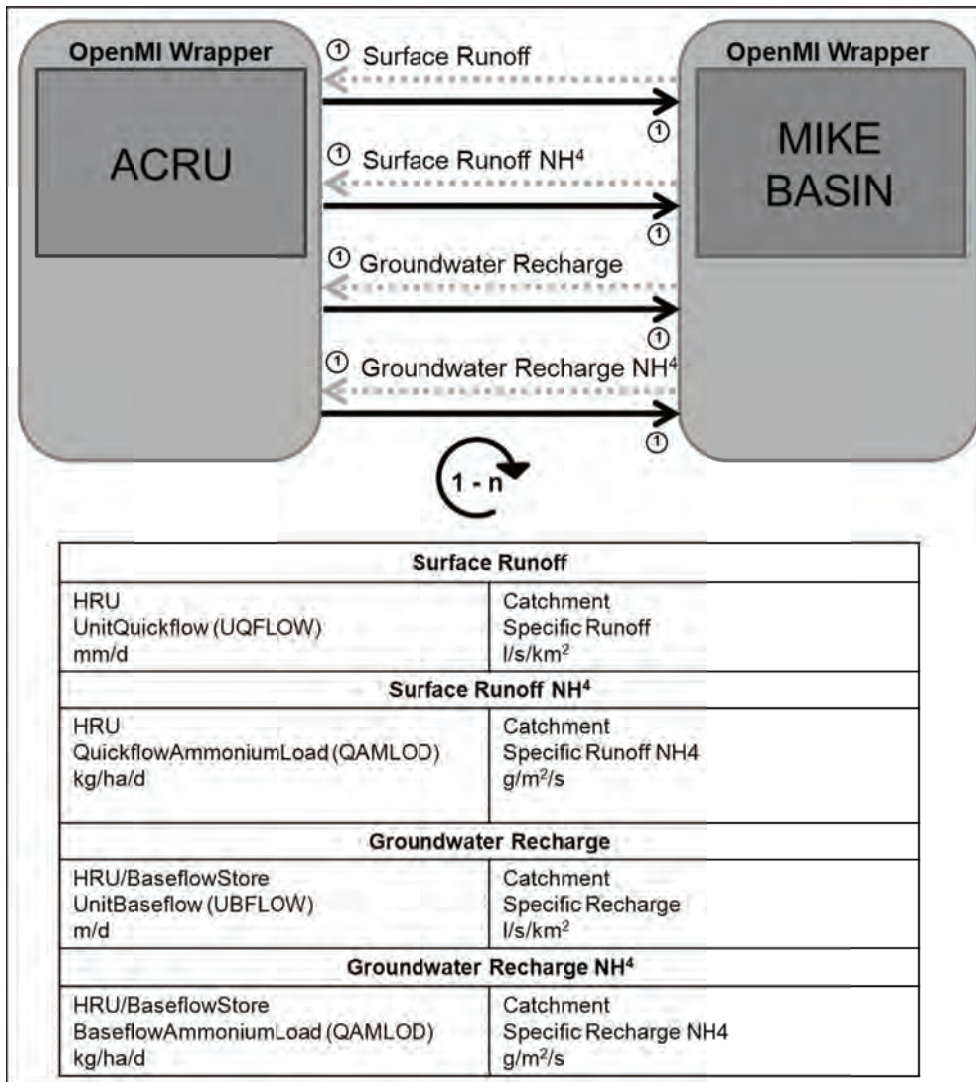


Figure 6.9 Water quality use case

## 6.5 Operations Modelling

The simple streamflow use case described in Section 6.1 is an example of how the integrated *ACRU* and MIKE BASIN models could be used in a planning context, where long historical or stochastic time series of model inputs such as rainfall and temperature are used as input to *ACRU* which generate simulated streamflows as input to MIKE BASIN which simulates water allocations and flows resulting in an estimate of catchment yield.

The integrated *ACRU* and MIKE BASIN models could be used in an operational context as demonstrated in the use case shown in Figure 6.10. Operational modelling requires that observed data be available to update critical state variables, such as reservoir storage, at regular intervals. If for example, seven day forecasts of rainfall and temperature are

available, these can be provided as input to the *ACRU* model. The integrated models can then be run for a range of reservoir release scenarios for seven daily time steps. A water manager can then make short term decisions regarding releases from reservoirs. The next day when a new seven day forecast is made available, the observed rainfall and temperature values for the previous week and forecast rainfall and temperature for the next seven days can be provided as input to the *ACRU* model, and both models are then for 8 daily time steps using the same starting state as before.

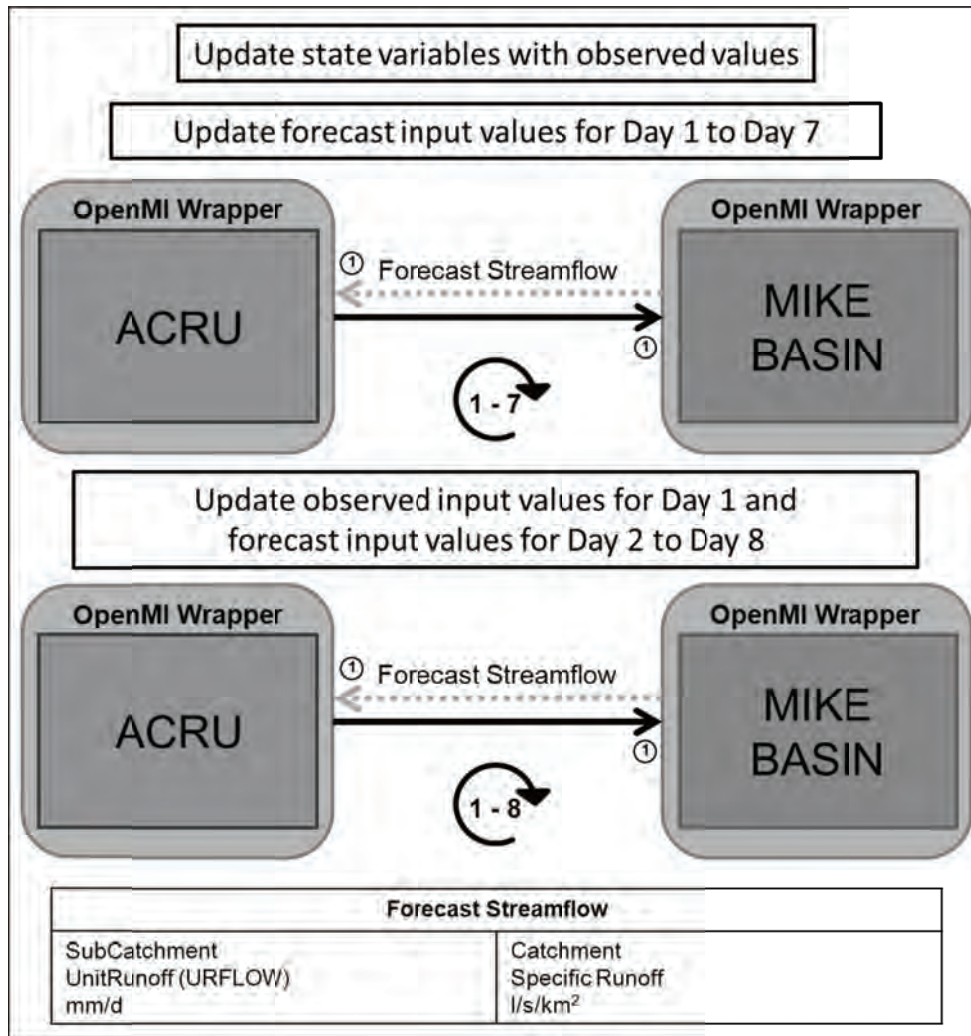


Figure 6.10 Operations modelling use case

## 7 CASE STUDY

### SLC Thornton-Dibb, DJ Clark and JC Smithers

The Kaap River is a tributary of the Crocodile River and is located within the Inkomati Water Management Area (WMA), situated in the north eastern part of South Africa as illustrated in Figure 7.1. The catchment area of the Kaap River is 1640 km<sup>2</sup> and the altitude varies from 1871 m, at Tafelkop in the west at the head of the Suidkaap tributary, down to 336 m in the north east at the confluence with the Crocodile River. There are two inter-catchment transfers into the Kaap River Catchment. The first is the Lomati transfer in which water is pumped from the Lomati Dam to the Umjindi Local Municipality (Barberton). The second is the Shiyalongubu Transfer that pumps water from the Shiyalongubu Dam for use by the Louws Creek Irrigation Board. The Lomati and Shiyalongubu Dams are located in the Quaternaries X14A and X14B respectively, within the Komati River Catchment to the south.

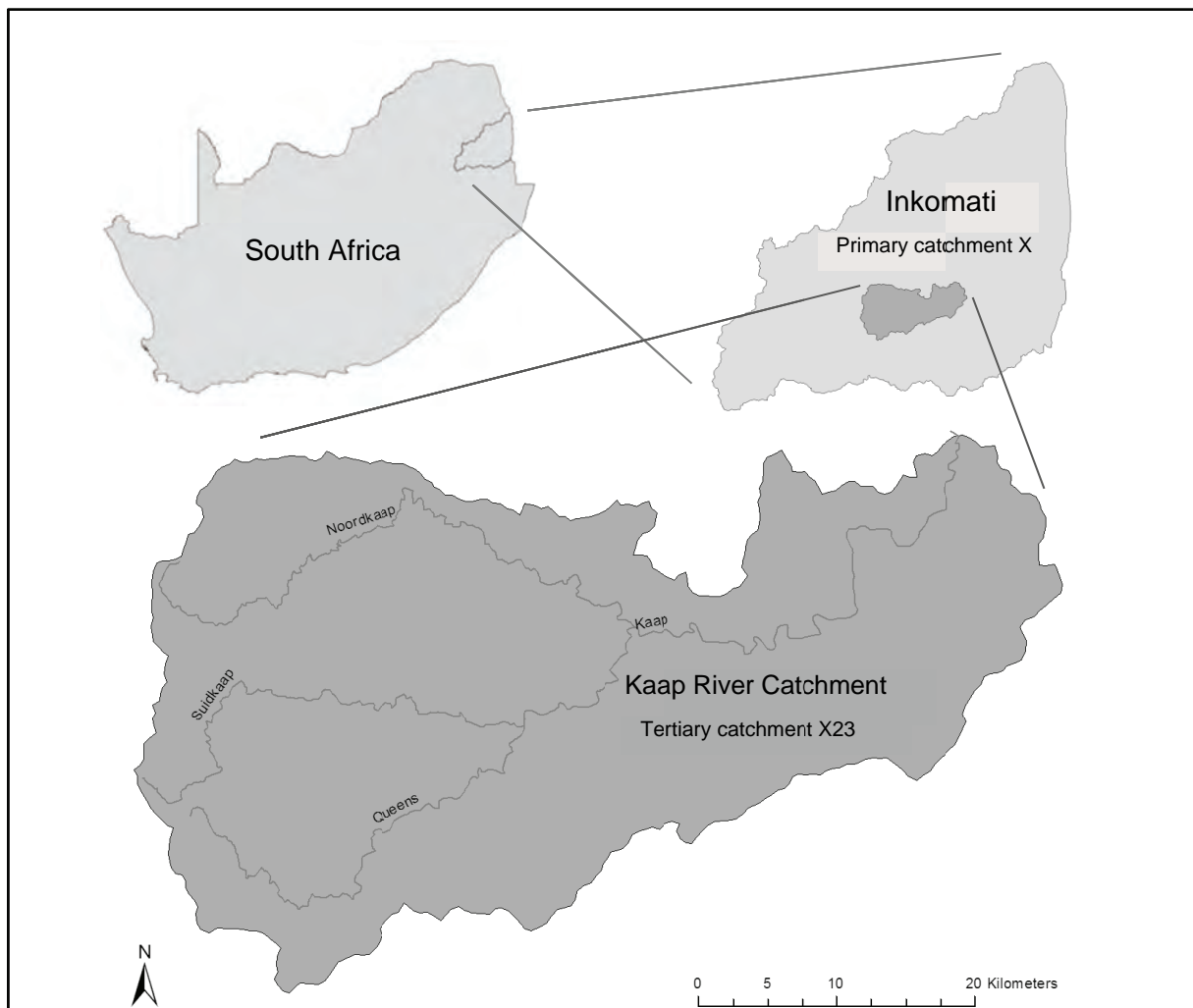


Figure 7.1 The Kaap River Catchment is represented by the tertiary catchment X23 which is located in the primary catchment X, namely the Inkomati



The Kaap River Catchment was selected as a test case as it has various competing water users including domestic, mining and irrigation, which primarily abstract water from the variable streamflow. The Kaap River Catchment forms a subcatchment of the Crocodile River Catchment, which flows from South Africa into Mozambique. The Kruger National Park's southern border is formed by the Crocodile River before it enters Mozambique and thus there is the Reserve Flow requirement required by the park as well as the international flow requirement at the Mozambique border that need to be considered. In order to meet these flow requirements the subcatchments of the Crocodile need to be managed holistically. This study was aimed at creating a better understanding of the Kaap River Catchment within this larger system. In order to achieve this, the *ACRU* and *MIKE BASIN* models were configured for the Kaap River Catchment.

### **7.1 MIKE BASIN Configuration**

The variation in altitude, represented by a 20 m Digital Elevation Model (DEM) obtained from Inkomati Catchment Management Agency (ICMA), is illustrated in Figure 7.2.

The delineation of the subcatchments was based on the “quinary” shapefile of the Inkomati catchment, received from DHI (Frezghi, 2012b) and originating from a report by Mallory and Beater commissioned by DWAF (DWAF, 2009). This “quinary” shapefile consisted of 20 subcatchments. Two of these subcatchments were further sub-divided based on the locations of the proposed Mountain View Dam (Haumann, 2008) and the Concession Creek Dam (Theron, 2006). The DEM was utilised to delineate these new subcatchments as well as the rivers within the *MIKE BASIN* software. The subcatchments, rivers, major dams, primary abstractions and transfers, based on a *MIKE BASIN* configuration received from (Frezghi, 2012a), were all added to the *MIKE BASIN* configuration as illustrated Figure 7.3. A dam representing the lumped dams within SubCatchment\_11 and the irrigation demand for the same catchment were added for testing model linking.

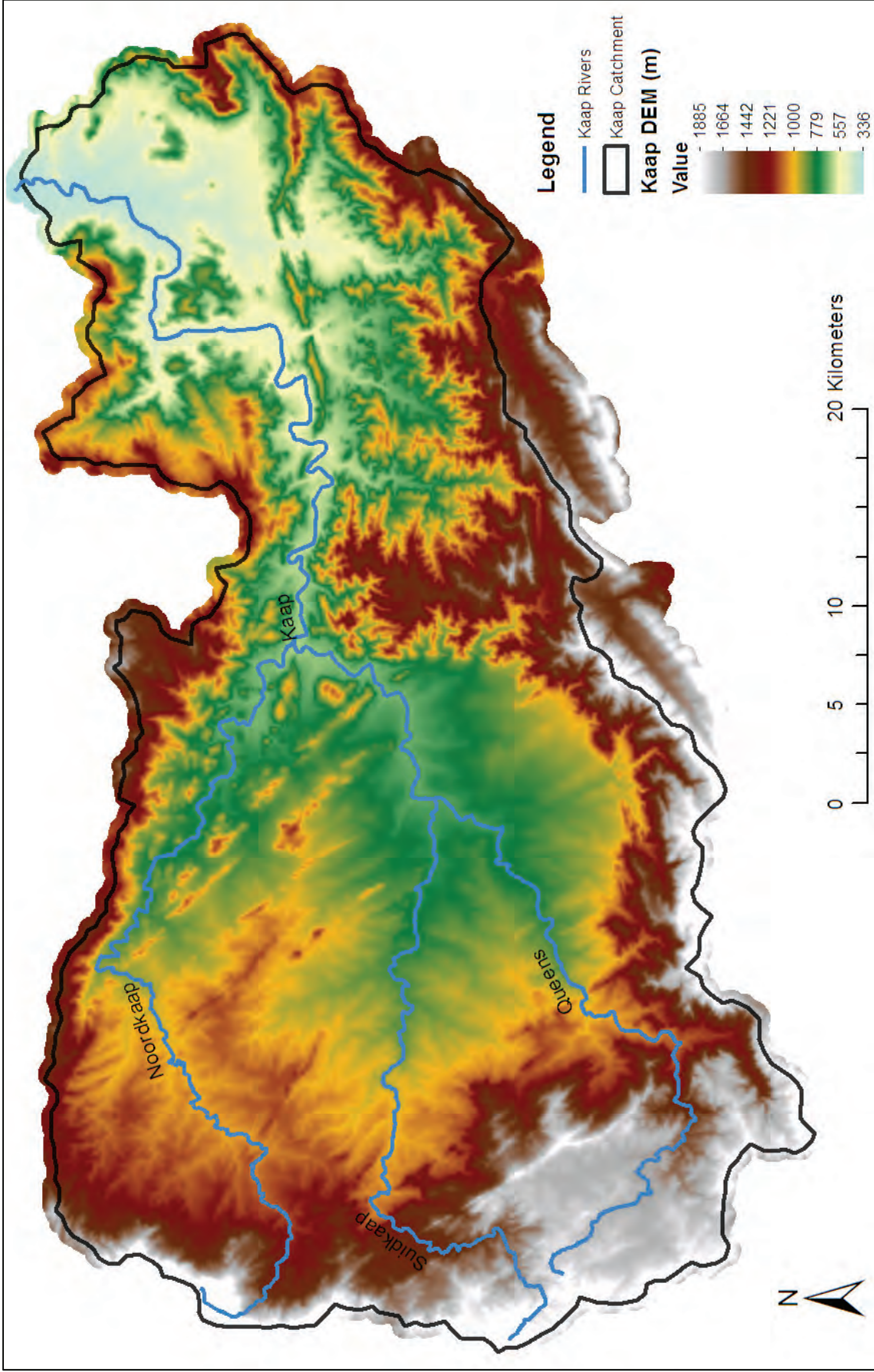


Figure 7.2 Altitude variation in the Kaap River Catchment

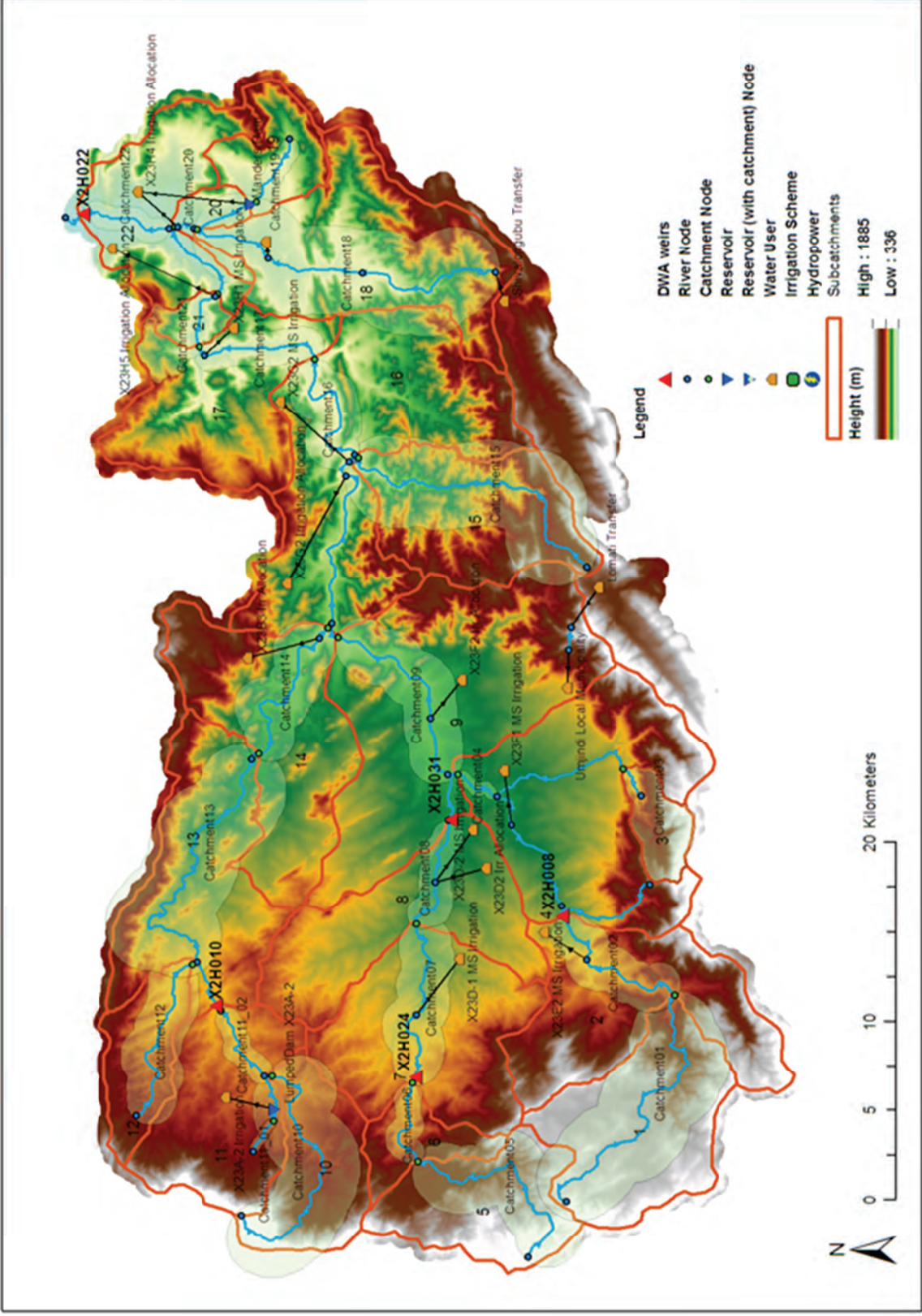


Figure 7.3 MIKE BASIN configuration for the Kaap River Catchment

## 7.2 ACRU Configuration

The ACRU4 version of the ACRU model described in Clark *et al.* (2009) was used for this case study. The ACRU model was configured for the Kaap River Catchment for subcatchments corresponding to those used in the MIKE BASIN configuration. Details of the climate, soils, land-use and other model inputs are described in the following sections.

### 7.2.1 Subcatchment and HRU configuration

The 22 subcatchments generated for the MIKE BASIN configuration were utilised to configure the ACRU model. These subcatchments, shown in Figure 7.4, were networked as shown in Figure 7.5. The areas of the quaternary catchments and their subcatchments are shown in Table 7.1.

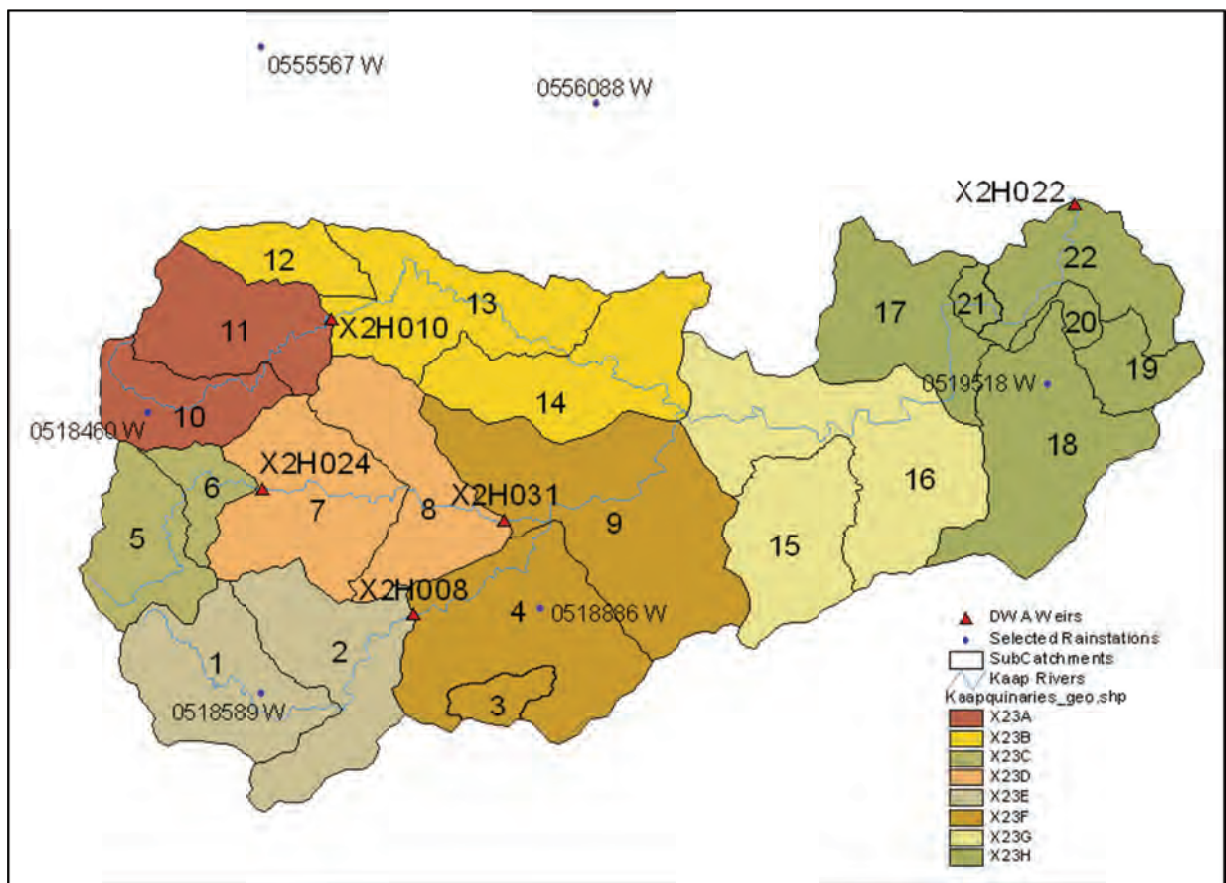


Figure 7.4 Location of quaternary catchments, subcatchments, flow gauging weirs and selected rain gauges

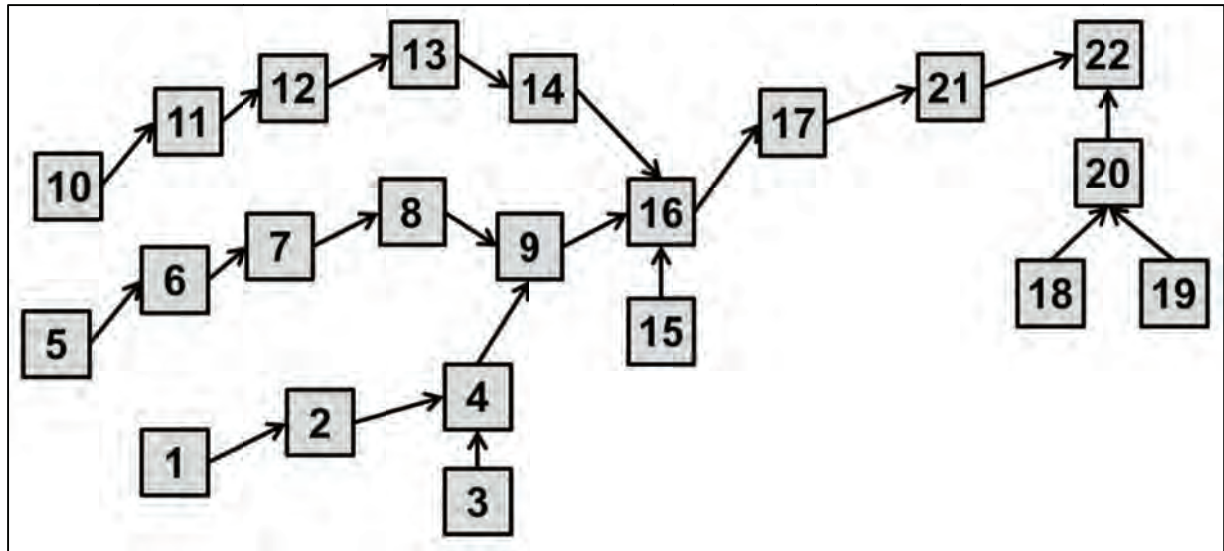


Figure 7.5 Subcatchment flow network for the Kaap River Catchment

Table 7.1 Quaternary catchment and subcatchment areas

Quaternary Catchment		Subcatchment	
ID	Area (km <sup>2</sup> )	ID	Area (km <sup>2</sup> )
X23A	126.813	10	51.637
		11	75.176
X23B	229.133	12	33.922
		13	97.288
		14	97.923
X23C	81.290	5	58.358
		6	22.932
X23D	181.871	7	98.422
		8	83.449
X23E	180.396	1	86.723
		2	93.673
X23F	309.588	3	15.614
		4	127.012
		9	166.962
X23G	225.098	15	75.889
		16	149.209
X23H	306.063	17	81.313
		18	110.171
		19	30.041
		20	11.017
		21	9.173
		22	64.348
<b>Total area of Tertiary Catchment X23 (km<sup>2</sup>)</b>		<b>1640.252</b>	

The *ACRU* model was configured for the Kaap River Catchment by firstly extracting the quaternary catchments within Tertiary Catchment X23, from the configuration used by Schulze *et al.* (2004). The *ACRU* input variables and parameters from these quaternary catchments were then used as generalised input into each of the new subcatchments.

The subcatchment coverage was used as input into the *ACRU* Grid Extractor (Lynch and Kiker, 2001), which is an ArcView extension used to extract the soils and land-type information required by the *ACRU* model. The subcatchments were divided into Hydrological Response Units (HRUs) based on a union of the land-use of the Kaap River Catchment which was obtained from the National Land Cover for the year 2000 (NLC, 2005) shown in Figure 7.9. The land-use polygons with the same specified land cover, contained within a subcatchment, were grouped together to form single representative HRUs. The representative HRUs contained within the subcatchments are listed in Appendix A.

### **7.2.2 Climate data**

The main driver input for the *ACRU* model is rainfall. Representative rain gauges were selected as driver stations for each of the subcatchments and extracted from the daily rainfall database (Lynch, 2004) into *ACRU* CompositeY2K format files. An *ACRU* Reference Climate Component was assigned to each of the subcatchments and the corresponding *ACRU* CompositeY2K linked to it. The catchment rainfall correction factors for each of the subcatchments used in this study were calculated from the gridded Mean Annual Precipitation (MAP) and updated in the configuration files. The MAP represented by a degree grid (Schulze *et al.*, 2008) is illustrated in Figure 7.6. Daily maximum and minimum temperatures were generated for the subcatchments, based on their centroids, from the gridded temperature database (Schulze *et al.*, 2010). Daily Penman-Monteith reference evaporation values were generated for the same points, and corresponding adjustment factors were derived to convert these Penman-Monteith reference evaporation values to A-pan equivalents, so that A-pan derived crop factors could be used for the vegetation, as was done by Schulze *et al.* (2010).

The variation in altitude, MAP and percentage reliability for the rain gauges in or near the Kaap River Catchment are shown in Table 7.2, and plots of accumulated rainfall are shown in Figure 7.7

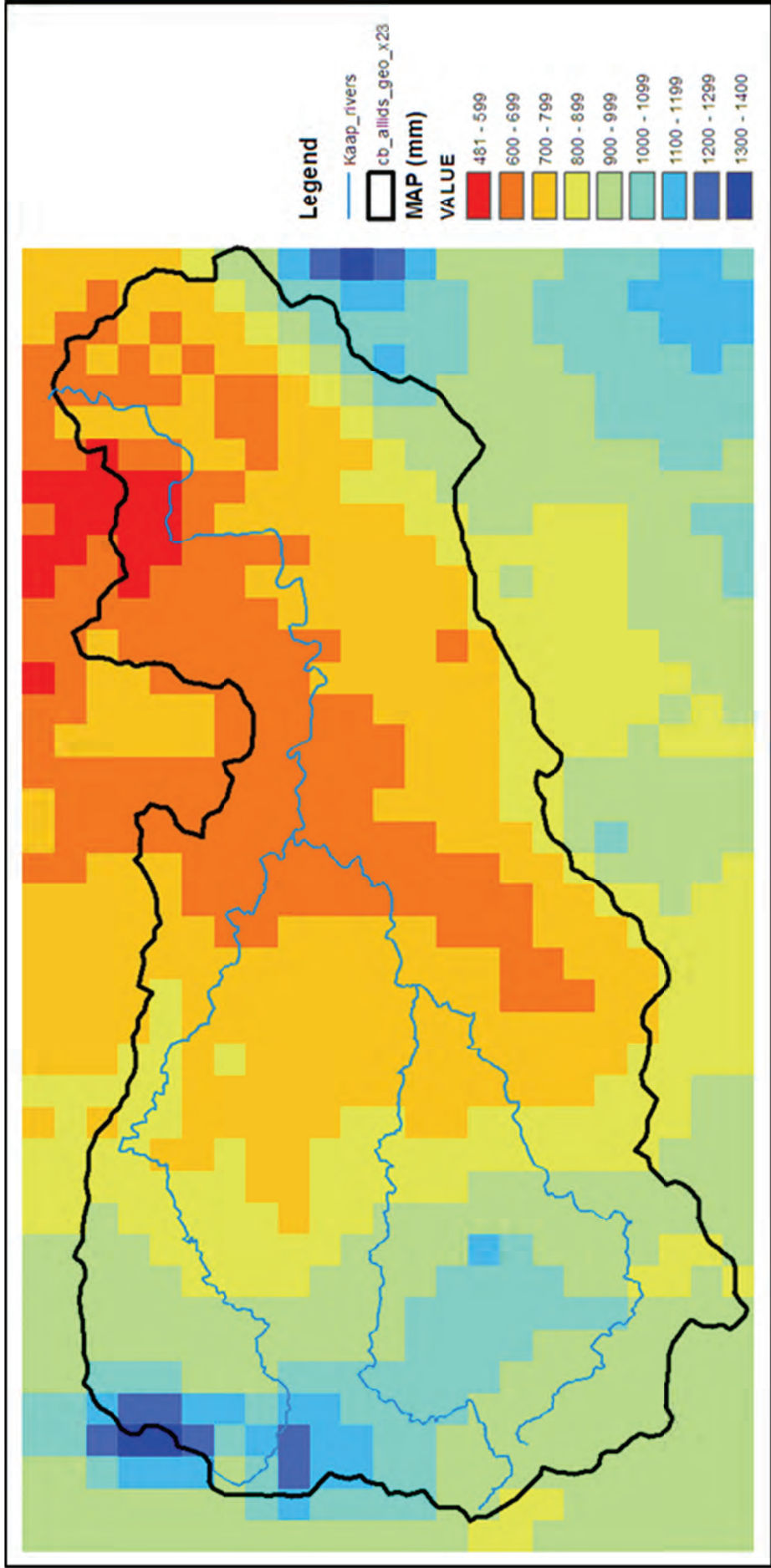


Figure 7.6 Distribution of MAP in the Kaap River Catchment (after Schulze *et al.*, 2008)

Table 7.2 Characteristics of rain gauges in or near the Kaap River Catchment

Rain Gauge ID	Altitude (m)	MAP (mm)	Reliability (%)
0518589W	1397	1036	97.4
0518460W	1079	1286	84.6
0518518W	994	957	17.2
0555567W	715	832	97.9
0518886W	703	681	80.7

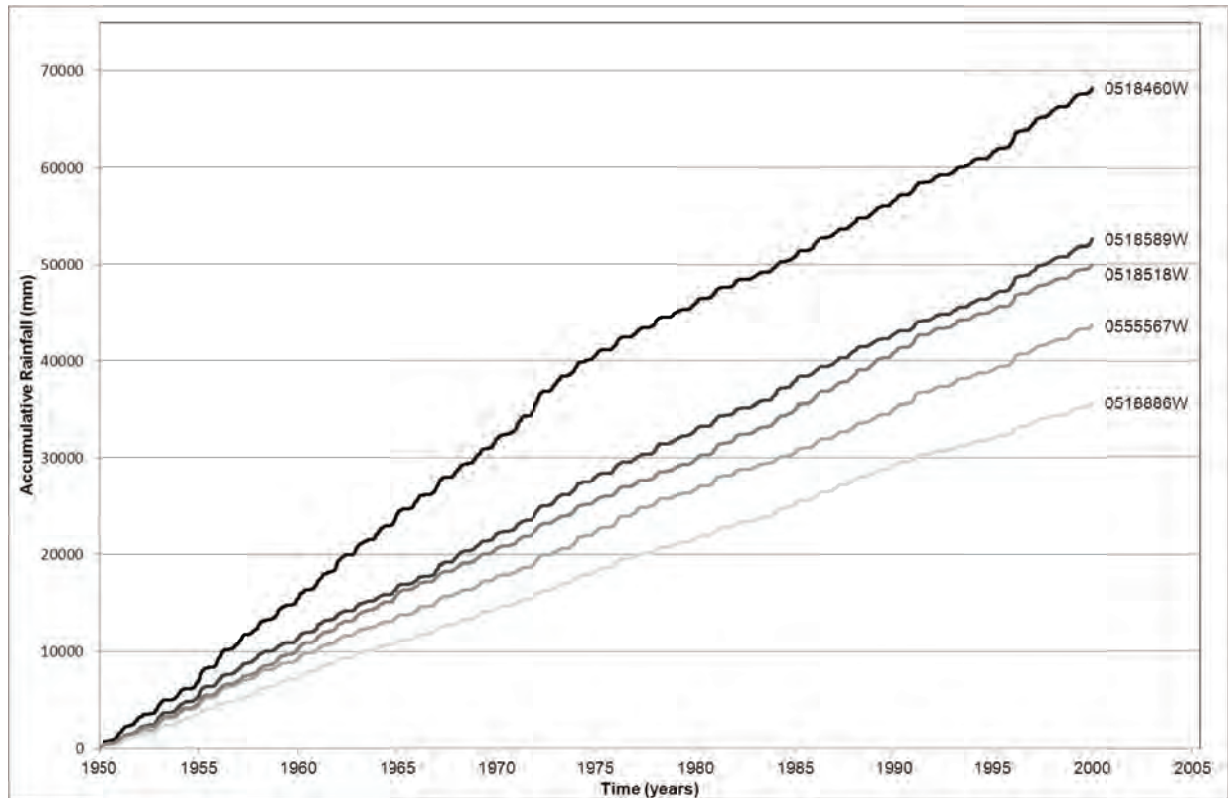


Figure 7.7 Rain gauges in or near the Kaap River Catchment

### 7.2.3 Soils and land-use

The Land Type and Year 2000 Land-use (Schulze *et al.*, 2008) were used as sources for the hydrological soils information, as shown by the broad soil type categories in Figure 7.8 and the Land-use in Figure 7.9. The catchment is predominantly covered by the land-covers described as “Thicket, Bushland, Bush Clumps, High Fynbos” and “Unimproved (natural) Grassland”. The forestry is concentrated in the higher altitudes and is largely in the western part of the catchment, where the soils are broadly of type Ab and Ac. The ACRU Grid extractor was used to extract soils parameters per subcatchment. This included Wilting Point (WP), porosity (PO), and Field Capacity (FC) for the A-Horizon and B-horizon; and depth and drainage response fractions for A and B Horizons (DEPAHO, DEPBHO, ABRESP and BFRESP respectively). Soil water evaporation and transpiration were modelled



separately, which required the *ACRU* soil texture class to be input, which was back calculated from the soils parameters (Schulze *et al.*, 1995a). Other land-uses include cultivated land, both irrigated and dry land, found chiefly in the lower altitudes, in addition to urban and mining areas.

#### **7.2.4 Dams and irrigated areas**

The polygons with descriptions containing irrigated areas, from the Land-use, were also simplified by combining them into a single representative irrigated HRU and adding these as specialised irrigation type HRUs within each subcatchment. Similarly, areas indicated as water bodies by the land-cover in each subcatchment, were aggregated into a single virtual dam and added as a Dam component in the relevant subcatchment.

The database of registered dams was downloaded from the Department of Water Affairs (DWA) Dams Safety website (DWA, 2012a). The database includes a Google Earth file that facilitated identifying dams within the Kaap River Catchment. However, this database is only accurate to the nearest second of a degree and therefore some of the locations in the file did not correspond with the dams identified using Google Earth. All the dams located in the Kaap River Catchment within this database were then correlated with the water bodies indicated by the land type coverage and the dams easily identifiable within Google Earth. Five unidentified dams were located within SubCatchment\_10 and Subcatchment\_11 and these small dams were lumped as a single virtual dam in the *ACRU* configuration.

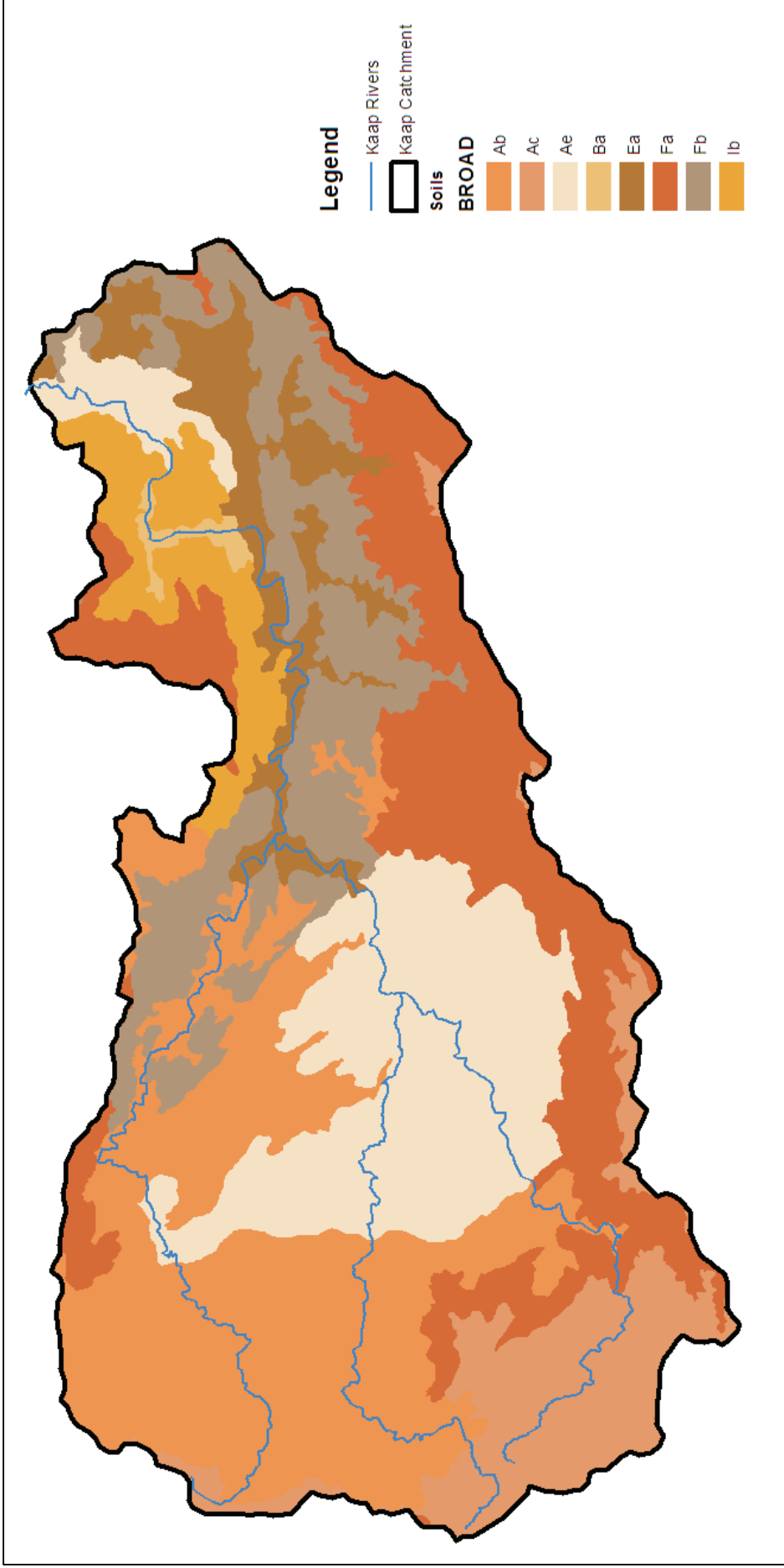


Figure 7.8 Broad soil classification within the Kaap River Catchment (after ISCW, 2005)

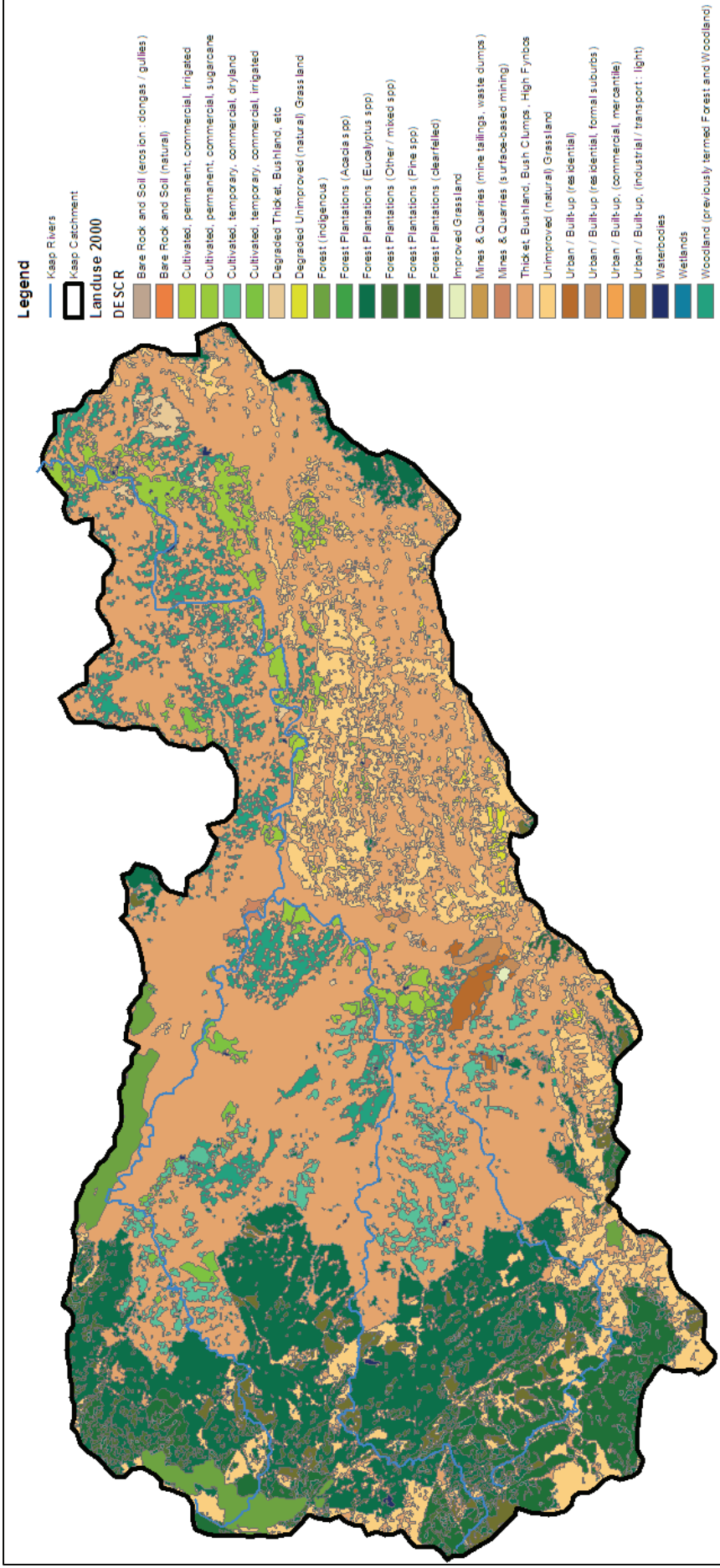


Figure 7.9 Land-use of the Kaap River Catchment (after NLC, 2005)

### 7.3 Verification Studies on Quaternary Subcatchments X23A, X23C and X23E

Schulze (1995) recommends that verification studies should ideally not be undertaken in operational catchments, as detailed input data are often not available and due to complexities arising from anthropogenic land and water use. This statement becomes clear in this chapter as the attempt to verify the simulations in Subcatchments X23A, X23C and X23E proved to be a challenge.

The HRUs within the subcatchments were configured as indicated in Figure 7.10. Even though there are multiple HRUs, only three are indicated in Figure 7.10 for the purpose of simplification.

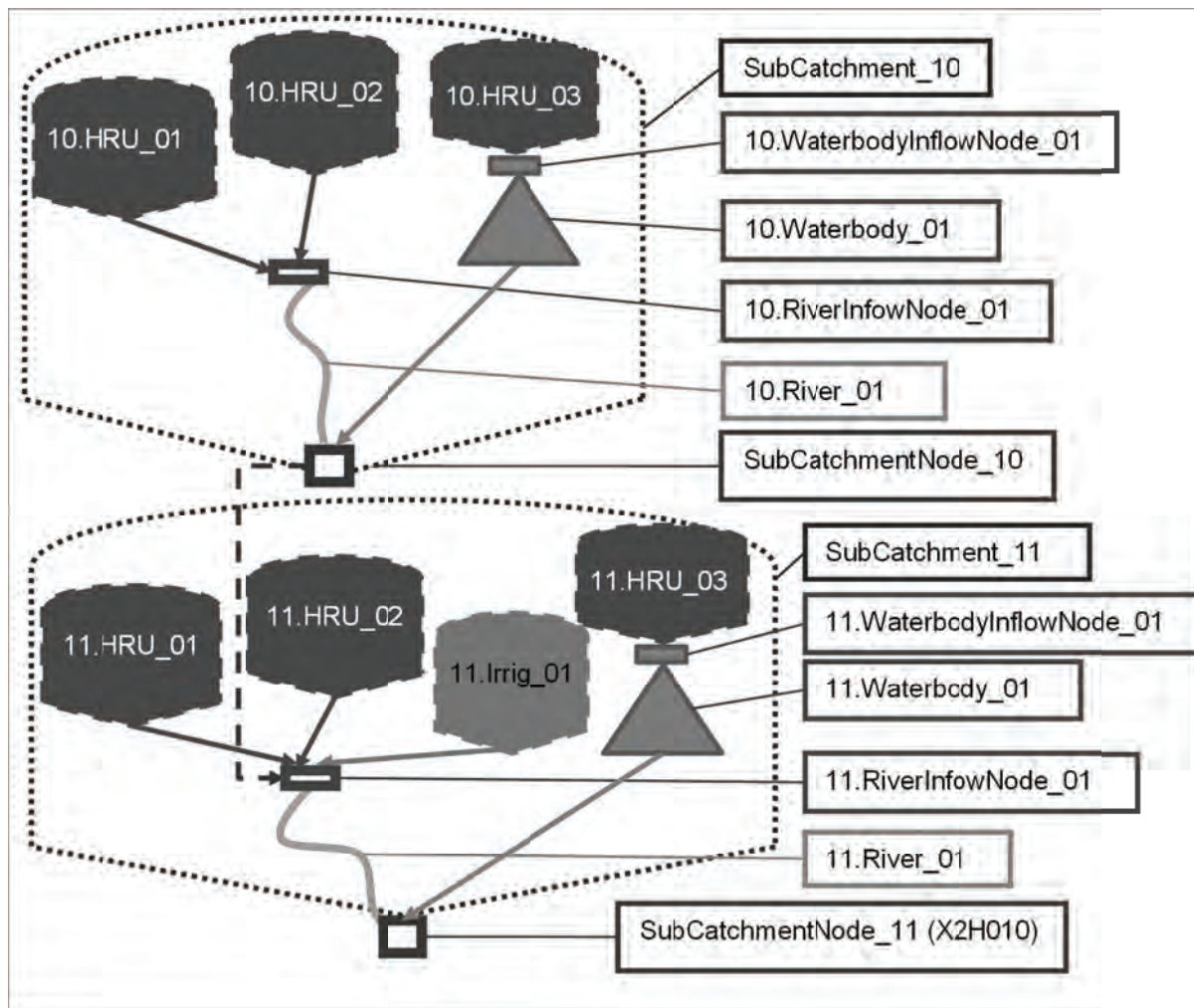


Figure 7.10 Example of configuration of HRUs within subcatchments

The Quaternary Catchment X23A is at the head waters of the Noordkaap River, with flows from the catchment recorded by the DWA Weir X2H010, shown in Figure 7.11. There is a

flow record from 1948 up to 2012 available for this weir from the DWA website (DWA, 2012). In the configuration used for this study the Quaternary Catchment X23A was divided into two subcatchments, SubCatchment\_10 and SubCatchment\_11.



Figure 7.11 Flow gauging weir X2H010 (DWA, 2012b)

From a comparison of the simulated and observed streamflow at SubCatchmentNode\_11 (Weir X2H010), it was apparent that there were a number of high simulated and observed flows that did not correlate, as indicated in Figure 7.12. On closer examination it was also noted that observed streamflow did not correspond to the rainfall events at these sites. The correlation coefficient ( $R^2$ ) of 0.44 also indicated a poor correlation between simulated and observed values. This could be as a result of the rain gauge selected for the subcatchment and the associated missing data that had been infilled, or errors in the observed flow data.

The accumulated plots of rainfall, observed and simulated unit flow depth are summarised in Figure 7.13. The accumulative plot of the rainfall, as shown in Figure 7.13, indicates a change in slope at approximately 1974 for the selected rain gauge, 0518460 W. The observed flow also indicates a change in slope in the mid-1960s. The observed and simulated accumulative totals appeared relatively similar, with a slight under simulation. As shown in Figure 7.13, there appears to be some under simulation occurring throughout the simulation period. However, the large events occurring in the years 1954, 1956, 1972 and 1996 appeared to have been over simulated. Furthermore, the weir may have not captured the full magnitude of these events as it is designed primarily to measure low flows.

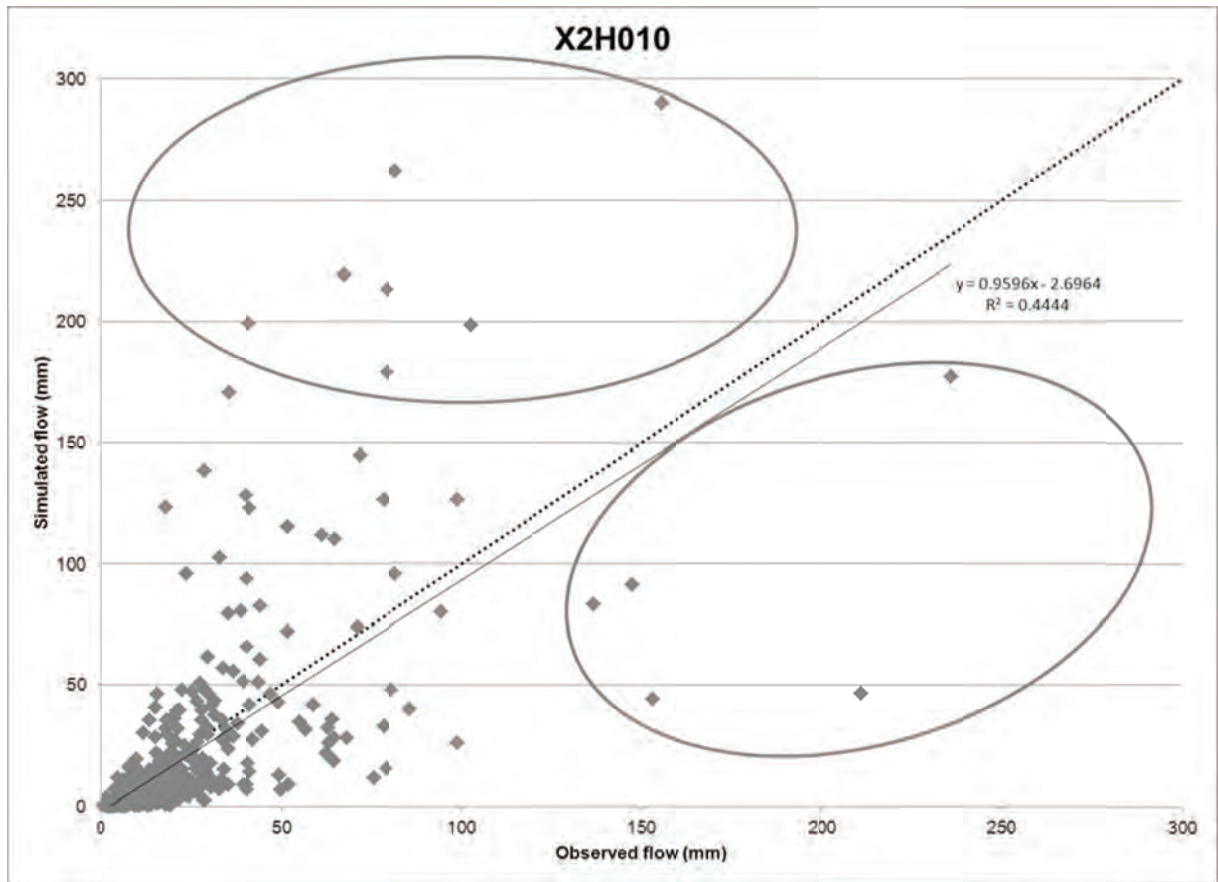


Figure 7.12 Simulated vs. observed monthly streamflow for Weir X2H010

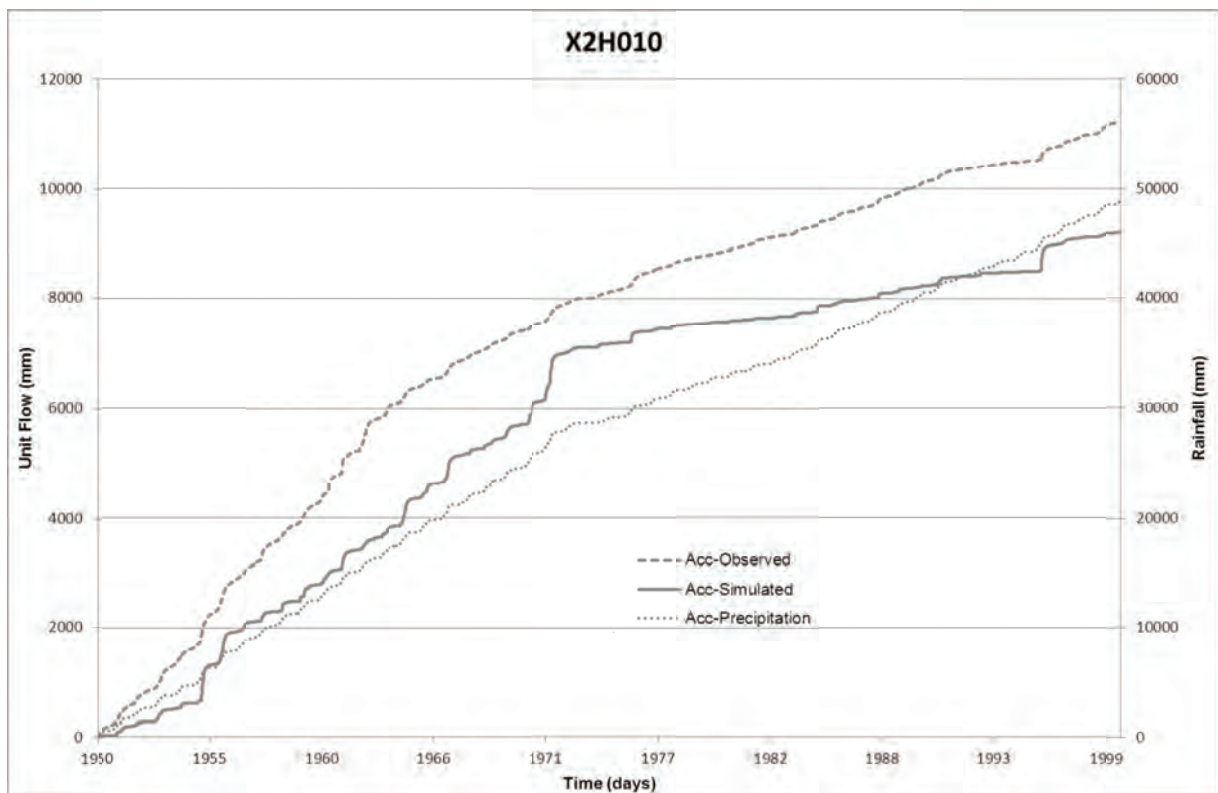


Figure 7.13 Accumulative rainfall, observed and simulated streamflow for Weir X2H010

The shift in the gradient of the accumulative observed flow could be as a result of land use changes with the catchment or abstractions. The increase in forest plantations over time is one possible explanation, as there is a large amount of forestry, approximately 60% of the catchment, as indicated by the land type coverage summarised in Table 7.3 and Table 7.4. An increase in abstractions is another possible explanation. The majority of the commercial forestry was planted before 1979 as indicated in Figure 7.14.

Table 7.3 Summary of land use in SubCatchment\_10

Description	Area (km <sup>2</sup> )	Area (%)
Cultivated, temporary, commercial, dryland	0.038	0.07
Forest (indigenous)	8.948	17.33
Forest Plantations	31.112	60.25
Thicket, Bushland, Bush Clumps, High Fynbos	2.770	5.36
Unimproved (natural) Grassland	8.729	16.90
Waterbodies	0.022	0.04
Wetlands	0.018	0.03
	51.637	100.00

Table 7.4 Summary of land use in SubCatchment\_11

Description	Area (km <sup>2</sup> )	Area (%)
Cultivated, temporary, commercial, dryland	4.918	6.54
Cultivated, temporary, commercial, irrigated	0.231	0.31
Forest (indigenous)	5.540	7.37
Forest Plantations	44.956	59.80
Thicket, Bushland, Bush Clumps, High Fynbos	16.210	21.56
Unimproved (natural) Grassland	3.185	4.24
Urban / Built-up, (industrial / transport : light)	0.097	0.13
Waterbodies	0.020	0.03
Wetlands	0.020	0.03
	75.177	100.00

The daily mean observed discharge at Weir X2H010, shown in Figure 7.15, clearly indicates a marked difference in the flow patterns before and after the mid-1960s. This is possibly due to a climatic shift or the introduction of commercial forestry. However, increased water abstractions and errors in measurement of streamflow could also result in differences between observed and simulated values.

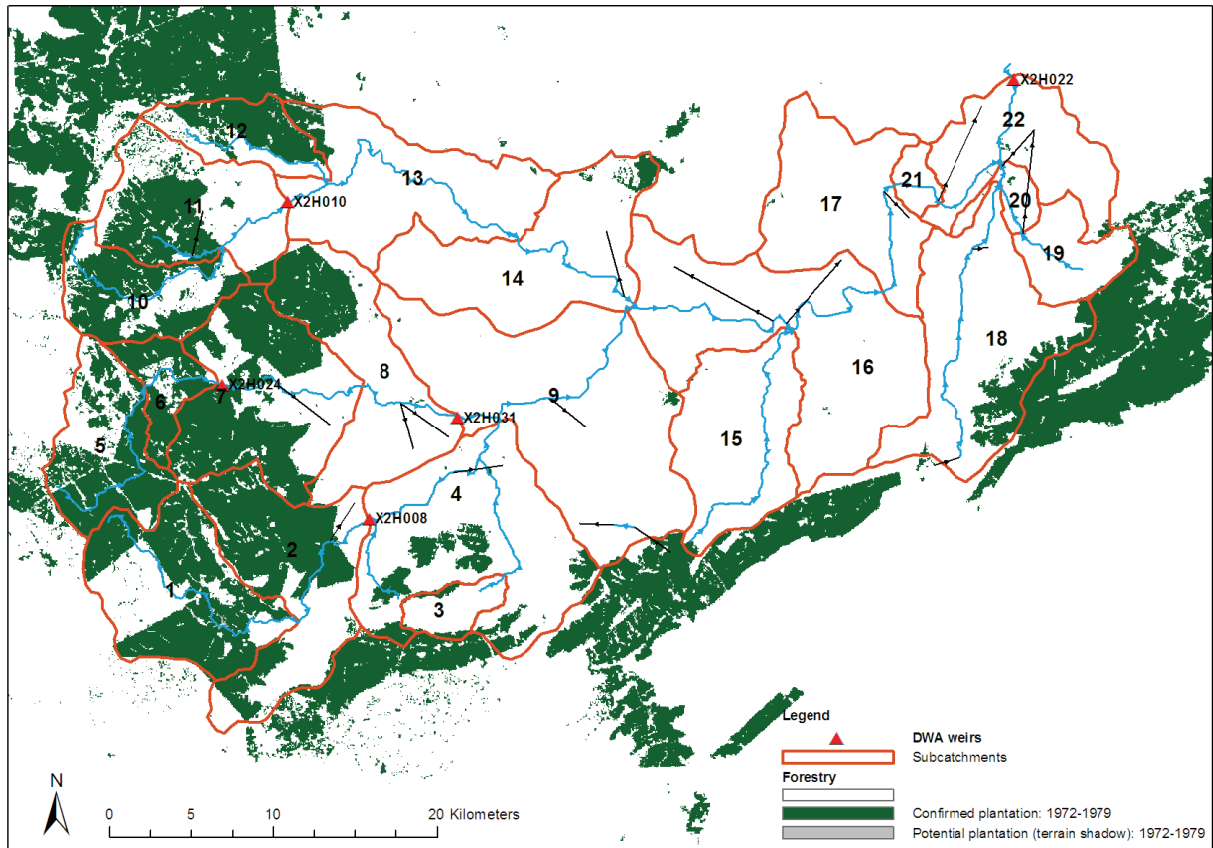


Figure 7.14 Plantation coverage (After Jackson, 2012)

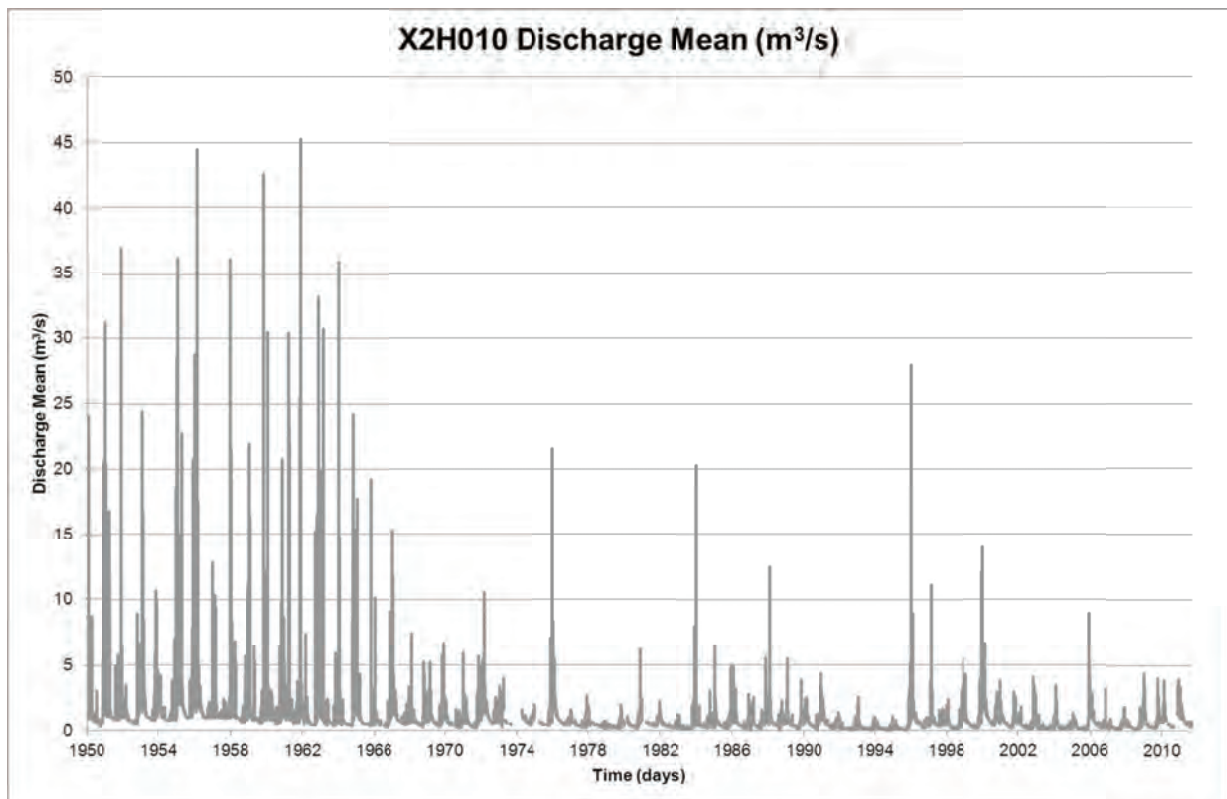


Figure 7.15 Daily mean observed discharge for X2H010



In a streamflow analysis for Weir X2H010 by Jewitt *et al.* (1999), a decrease in the observed low flows was indicated in the 7-day minimum flow plot as shown in Figure 7.16. Further investigation found that the weir had a change in ratings in July 1966, indicated by the dotted line added in Figure 7.16. The ratings are indicated in Figure 7.17.

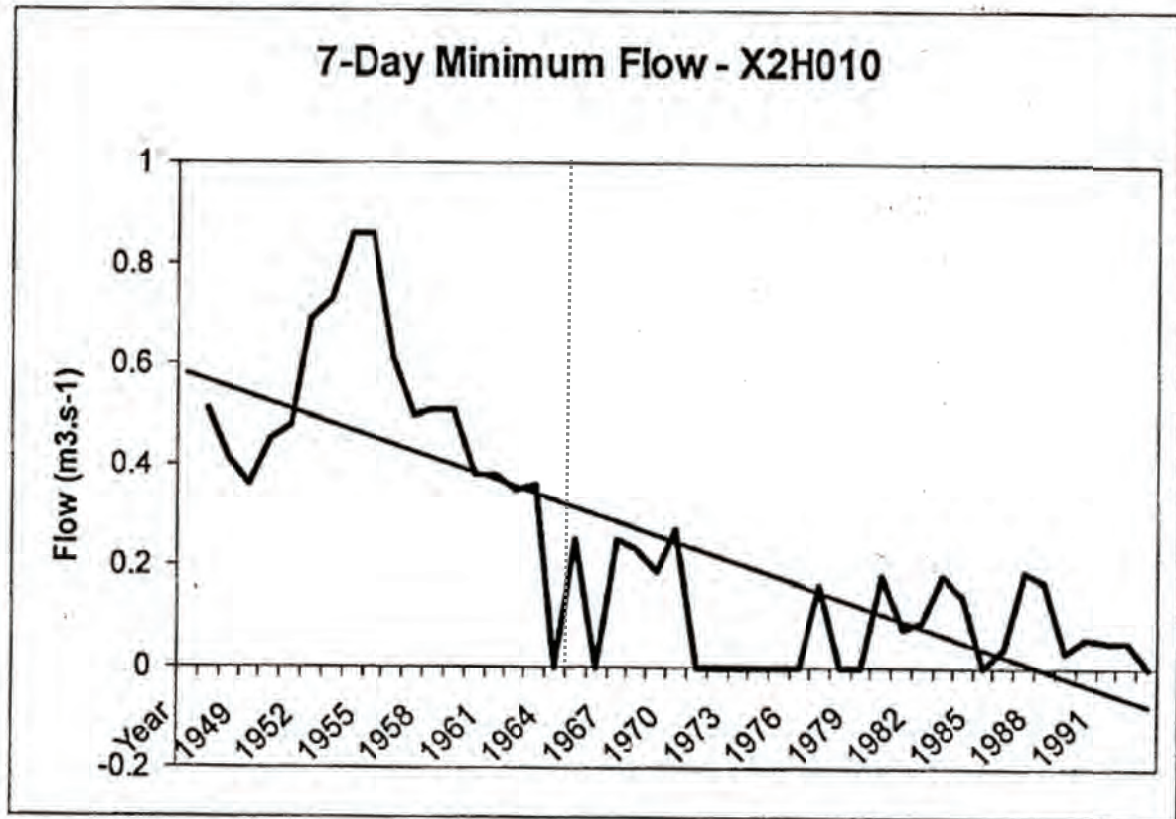


Figure 7.16 7-Day Minimum flow for Weir X2H010 (after Jewitt *et al.*, 1999)

The point at which the change in rating occurs is indicated on a double mass plot of the accumulative observed streamflow plotted against the accumulative rainfall in Figure 7.18. There are no flows in the record that exceed the rating capacity for the ratings for period (1966-2012), therefore the flattening out of the rating for the second period (1996-2012) does not have any effect on the current historical dataset. The second rating table does indicate that it has a higher accuracy in the low flows over the first period's ratings. It needs to be investigated if this change in ratings was accompanied by the design and construction of a new weir.

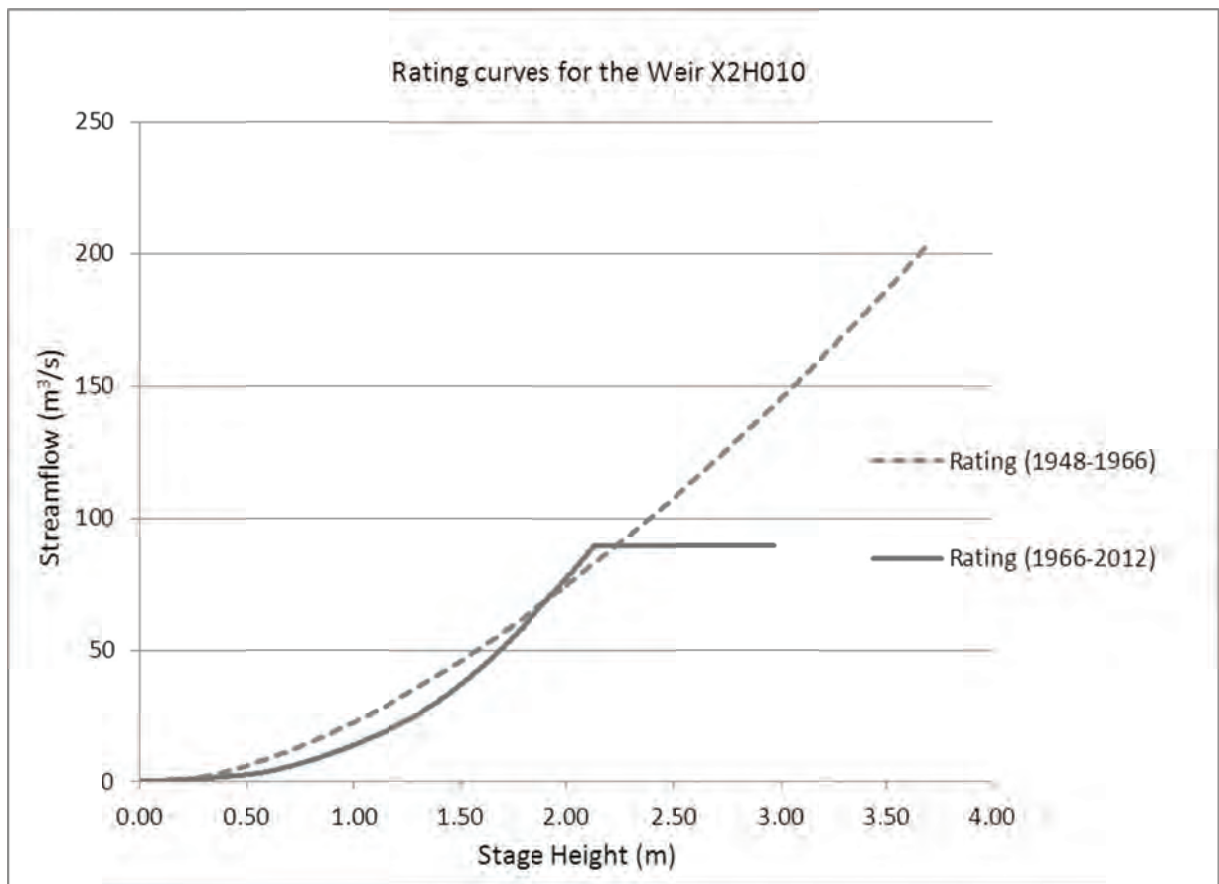


Figure 7.17 Rating curves for Weir X2H010

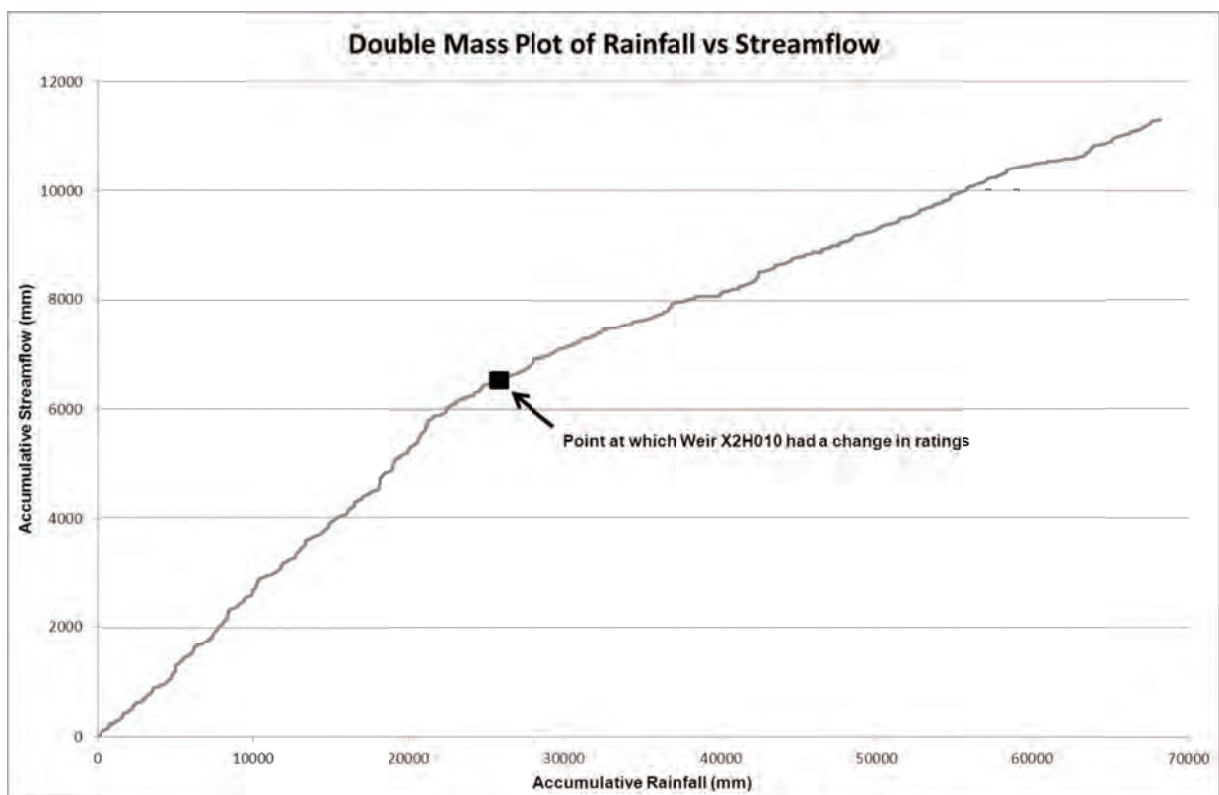


Figure 7.18 Double mass plot of the rainfall and observed streamflow

The Quaternary Catchment X23C is at the head waters of the Suidkaap River, with flows from the catchment recorded by the DWA Weir X2H024, shown in Figure 7.19. There is a flow record from 1964 till 2012 available for this weir from the DWA website (DWA, 2012). The daily mean observed discharge at Weir X2H024 is shown in Figure 7.20. The observed daily flow for Weir X2H024 indicates a lower frequency of high flow events after 1978, this again could be due to observation errors or increased abstractions. The statistics of accumulative observed monthly rainfall and streamflow, and simulated streamflow for Weir X2H024 are illustrated in Figure 7.21 and Figure 7.22. The observed streamflow indicates a change in gradient in the 1960s possibly related to the introduction of forestry in the catchment. In the configuration used for this study the Quaternary Catchment X23C was divided into two subcatchments, SubCatchment\_05 and SubCatchment\_06.



Figure 7.19 Flow gauging weir X2H024 (DWA, 2012)

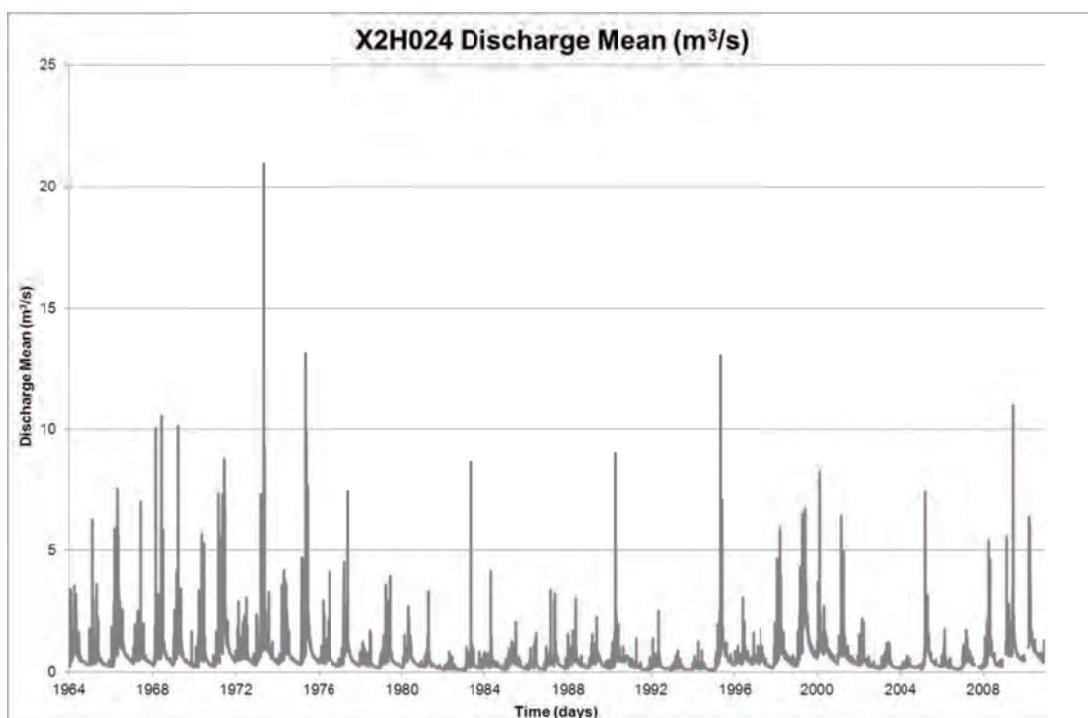


Figure 7.20 Daily mean observed discharge for X2H024

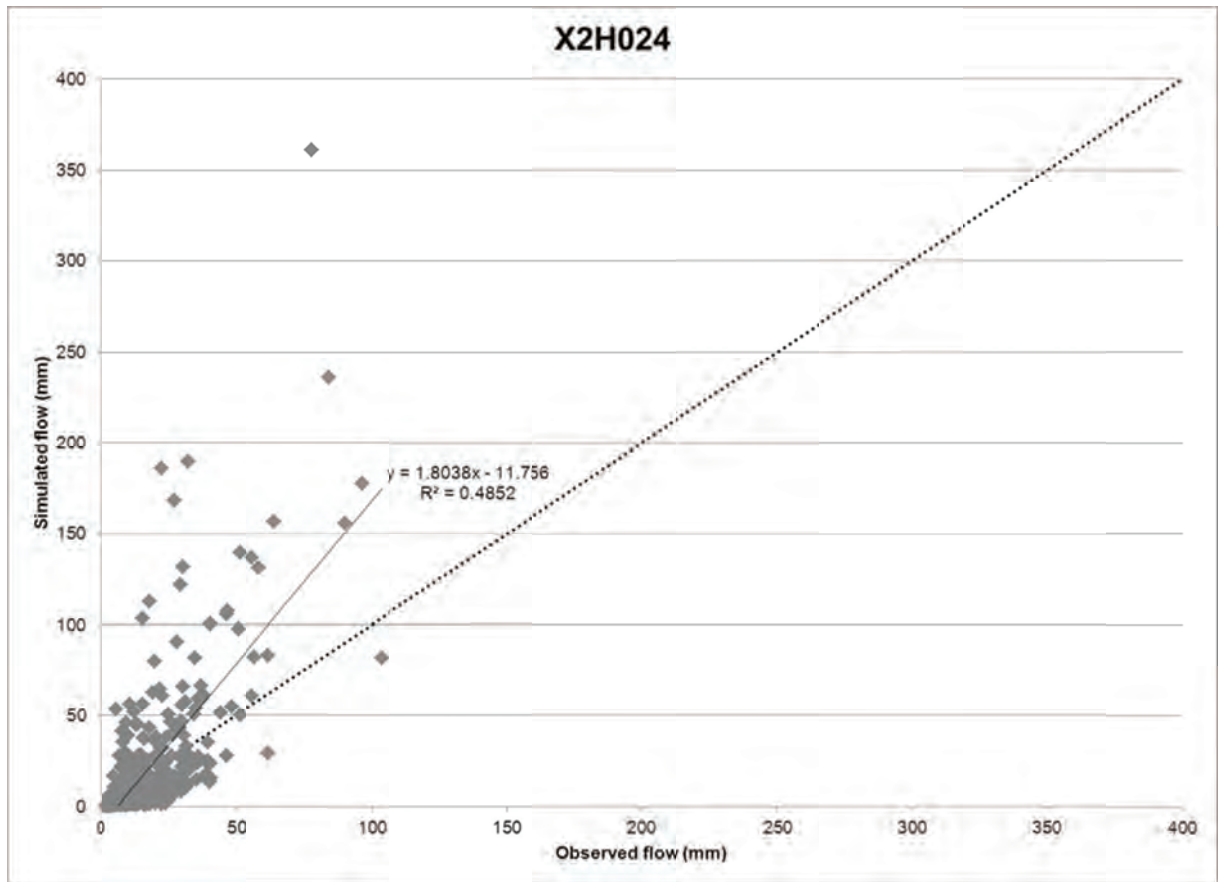


Figure 7.21 Simulated vs. observed monthly streamflow for Weir X2H024

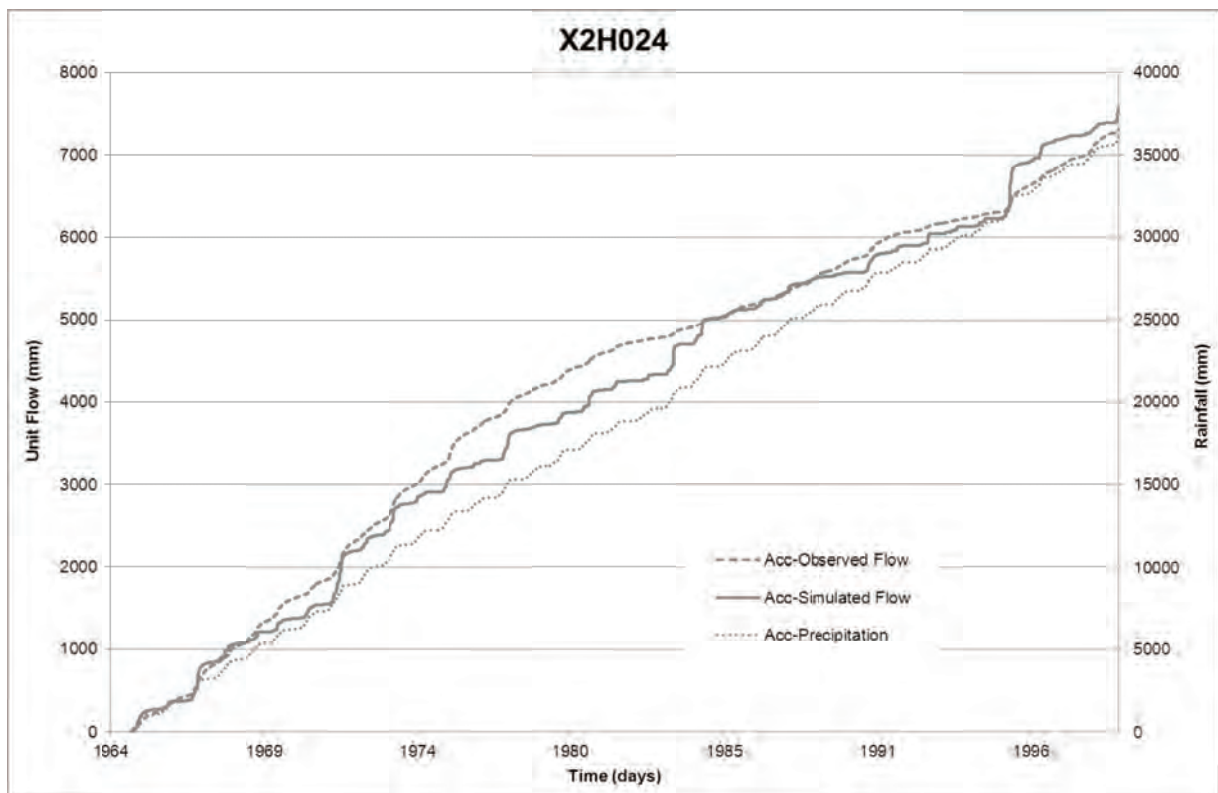


Figure 7.22 Accumulative rainfall, observed and simulated streamflow for Weir X2H024

The Quaternary Catchment X23E is at the head waters of the Queens River, with flows from the catchment recorded by the DWA Weir X2H008, shown in Figure 7.23. There is a flow record from 1948 up to 2012 available for this weir from the DWA website (DWA, 2012). The daily mean observed discharge at Weir X2H008 is shown in Figure 7.24. The large flows in 1984 were as a result of 324 mm of rain over two days; and in 1996 as a result of 334.8 mm over 7 days. The statistics of accumulative observed monthly totals of rainfall and streamflow, and simulated streamflow for Weir X2H024 are illustrated in Figure 7.25 and Figure 7.26. There was a marked over simulation of the streamflow, which could be due to the rain gauge selected or abstraction within the catchment. In the configuration used for this study the Quaternary Catchment X23A was divided into two subcatchments, SubCatchment\_01 and SubCatchment\_02.



Figure 7.23 Flow gauging weir X2H008 (DWA, 2012)

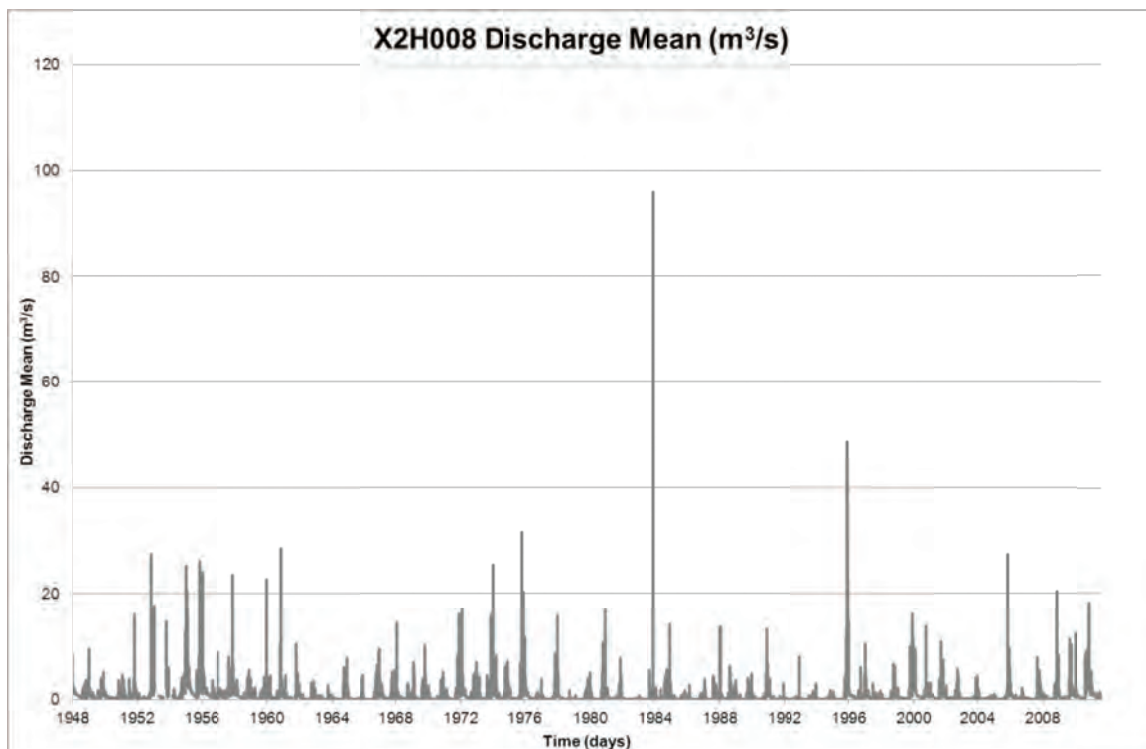


Figure 7.24 Daily mean observed discharge for X2H008

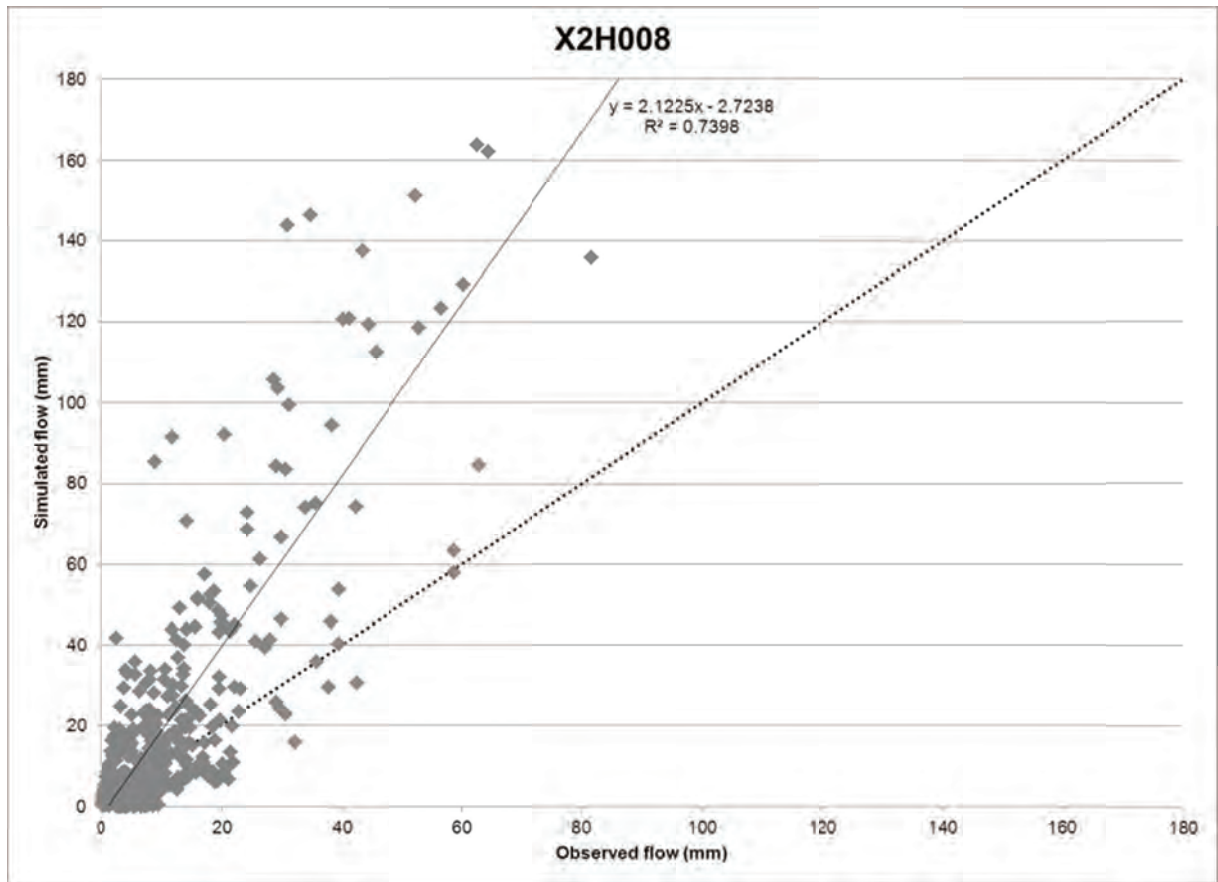


Figure 7.25 Simulated vs. observed monthly streamflow for Weir X2H008

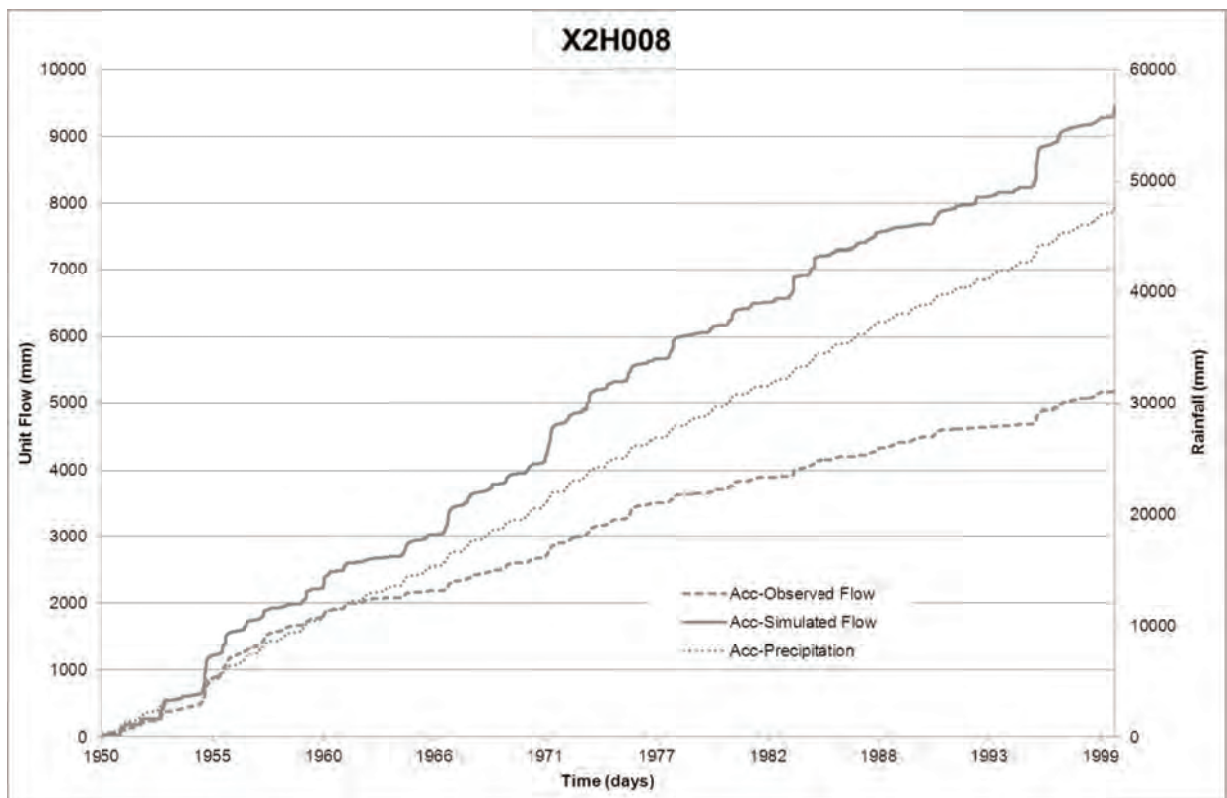


Figure 7.26 Accumulative rainfall, observed and simulated streamflow for Weir X2H008

An analysis of the accumulative observed rainfall and streamflow for a common period with no missing data indicated that X2H010 and X2H024 had a runoff:rainfall ratio of 16%, as opposed to that of 13% for X2H008 which appears to be low for this region. The best simulation results were obtained from Subcatchment X23A (Weir X2H010).

#### **7.4 Use Cases**

To demonstrate the use of the integrated *ACRU* and MIKE BASIN models, these linked models were run for three use cases for the Kaap River Catchment. In the first use case (Streamflow) simple uni-directional links were created between the models such that *ACRU* provides simulated streamflow inputs to MIKE BASIN, which models water allocation and use. The second use case is similar to the first use case, except that MIKE BASIN now also simulates flow routing down river reaches to represent lag and attenuation of flows as they move through the river network. The third use case demonstrates bi-directional links between the models, where *ACRU* provides simulated streamflow inputs and irrigation requirements to MIKE BASIN, which models water allocation and use and informs *ACRU* of the water allocated to irrigation water users. For these use cases both models were run at a daily time step, though OpenMI does enable linking models running at different time steps. As described in the Chapter 5, OpenMI wrappers were created for the *ACRU* and MIKE BASIN models. Tests were run using simple hypothetical catchment configurations to validate that the models could successfully be linked using these OpenMI wrappers. However, setting up and running the three use cases presented in this chapter provided valuable experience in applying the integrated models to a real catchment by highlighting difficulties and areas for further research.

Although OpenMI makes it relatively easy to link two OpenMI compliant models, the linking requires expert knowledge of both models in order to set up the links correctly and not compromise the integrity of either model. When setting up the individual models it is important to first have a detailed plan of how the models are to be linked as it may be necessary for the individual models to be configured in a particular way to facilitate the links.

In these use cases, wetlands were modelled as ordinary HRUs in *ACRU*, as in *ACRU* river reaches spill onto the wetland areas when flow exceed channel capacity. This represents another instance of a feedback between the systems represented by the individual models and requires further investigation. This is not expected to have a significant effect on the

results from these use cases, as wetlands represent a very small area of the whole Kaap River Catchment.

In the verification studies discussed in Section 7.3, it was observed that the catchment appeared to have experienced significant land use changes over time, possibly due to afforestation, prior to 1972. For this reason it was decided that the use case simulations would be run for 28 years, from 1972 to 1999. Initial attempts at running the use cases failed due to *System.OutOfMemoryException* errors. When run individually, both models ran for the full 28 year simulation period without errors and in a relatively short space of time. It was anticipated that integrating the models, exchanging data on a daily basis via OpenMI, would result in some performance penalties and would require additional memory resources, though the full extent of this only became apparent when running these relatively large model configurations. For the purpose of these use cases it was necessary to reduce the simulation period to just 10 years, from 1990 to 1999, to avoid the *System.OutOfMemoryException* errors.

In addition to the three use cases described below, the *ACRU* model was configured and run as a standalone model in which both irrigation water users and water transfers into the catchment were represented. This *ACRU* configuration provided a reference against which to compare the Irrigation use case in Section 7.4.3

#### **7.4.1 Streamflow links**

In this use case the *ACRU* model is used to simulate runoff from the 22 subcatchments and these runoff depths are then provided as input to the MIKE BASIN model which models all water allocation and use, and flows down the river network. The catchment configuration for MIKE BASIN is shown in Figure 7.27. Each catchment in MIKE BASIN was assigned a specific runoff time series containing zero values which get overwritten by the runoff quantities simulated by *ACRU*. For this use case the water users and their demands set up in MIKE BASIN were obtained from DHI (Frezghi, 2012b) and were based on a report by Mallory and Beater commissioned by DWAF (DWAF, 2009). There was no return flows specified for any of the irrigation water users. In the *ACRU* model, irrigation was turned off for all irrigated areas, effectively making these dryland cropping areas. In this use case simple uni-directional links were created between the *ACRU* subcatchment *UnitRunoff* (*URFLOW*) variables and the MIKE BASIN *Specific Runoff* variables as listed in Table 7.5. There was a special case in *Catchment11* in MIKE BASIN where some of the runoff flows into the dam named *Lumped Dam X23A-2* which is used for irrigation and some runoff flows



into the river reach down stream of this dam. In this situation two catchments, *Catchment11\_01* and *Catchment11\_02*, needed to be created in MIKE BASIN to receive separate runoff values from *ACRU* river nodes *11.WaterbodyInflowNode\_01* and *RiverInflowNode\_11* respectively.

Table 7.5 The linked model variables for the streamflow use case

<b>ACRU</b>	<b>MIKE BASIN</b>
SubCatchment_01: UnitRunoff (mm/d)	Catchment01: Specific Runoff (l/s/km <sup>2</sup> )
SubCatchment_02: UnitRunoff (mm/d)	Catchment02: Specific Runoff (l/s/km <sup>2</sup> )
SubCatchment_03: UnitRunoff (mm/d)	Catchment03: Specific Runoff (l/s/km <sup>2</sup> )
SubCatchment_04: UnitRunoff (mm/d)	Catchment04: Specific Runoff (l/s/km <sup>2</sup> )
SubCatchment_05: UnitRunoff (mm/d)	Catchment05: Specific Runoff (l/s/km <sup>2</sup> )
SubCatchment_06: UnitRunoff (mm/d)	Catchment06: Specific Runoff (l/s/km <sup>2</sup> )
SubCatchment_07: UnitRunoff (mm/d)	Catchment07: Specific Runoff (l/s/km <sup>2</sup> )
SubCatchment_08: UnitRunoff (mm/d)	Catchment08: Specific Runoff (l/s/km <sup>2</sup> )
SubCatchment_09: UnitRunoff (mm/d)	Catchment09: Specific Runoff (l/s/km <sup>2</sup> )
SubCatchment_10: UnitRunoff (mm/d)	Catchment10: Specific Runoff (l/s/km <sup>2</sup> )
11.WaterbodyInflowNode_01: UnitRunoff (mm/d)	Catchment11_01: Specific Runoff (l/s/ km <sup>2</sup> )
RiverInflowNode_11: UnitRunoff (mm/d)	Catchment11_02: Specific Runoff (l/s/ km <sup>2</sup> )
SubCatchment_12: UnitRunoff (mm/d)	Catchment12: Specific Runoff (l/s/km <sup>2</sup> )
SubCatchment_13: UnitRunoff (mm/d)	Catchment13: Specific Runoff (l/s/km <sup>2</sup> )
SubCatchment_14: UnitRunoff (mm/d)	Catchment14: Specific Runoff (l/s/km <sup>2</sup> )
SubCatchment_15: UnitRunoff (mm/d)	Catchment15: Specific Runoff (l/s/km <sup>2</sup> )
SubCatchment_16: UnitRunoff (mm/d)	Catchment16: Specific Runoff (l/s/km <sup>2</sup> )
SubCatchment_17: UnitRunoff (mm/d)	Catchment17: Specific Runoff (l/s/km <sup>2</sup> )
SubCatchment_18: UnitRunoff (mm/d)	Catchment18: Specific Runoff (l/s/km <sup>2</sup> )
SubCatchment_19: UnitRunoff (mm/d)	Catchment19: Specific Runoff (l/s/km <sup>2</sup> )
SubCatchment_20: UnitRunoff (mm/d)	Catchment20: Specific Runoff (l/s/km <sup>2</sup> )
SubCatchment_21: UnitRunoff (mm/d)	Catchment21: Specific Runoff (l/s/km <sup>2</sup> )
SubCatchment_22: UnitRunoff (mm/d)	Catchment22: Specific Runoff (l/s/km <sup>2</sup> )

This use case was successfully run for the simulation period 1990 to 1999, and the results are discussed in Section 7.4.4. This use case could be achieved using series linking, but is now easier and should produce identical results.

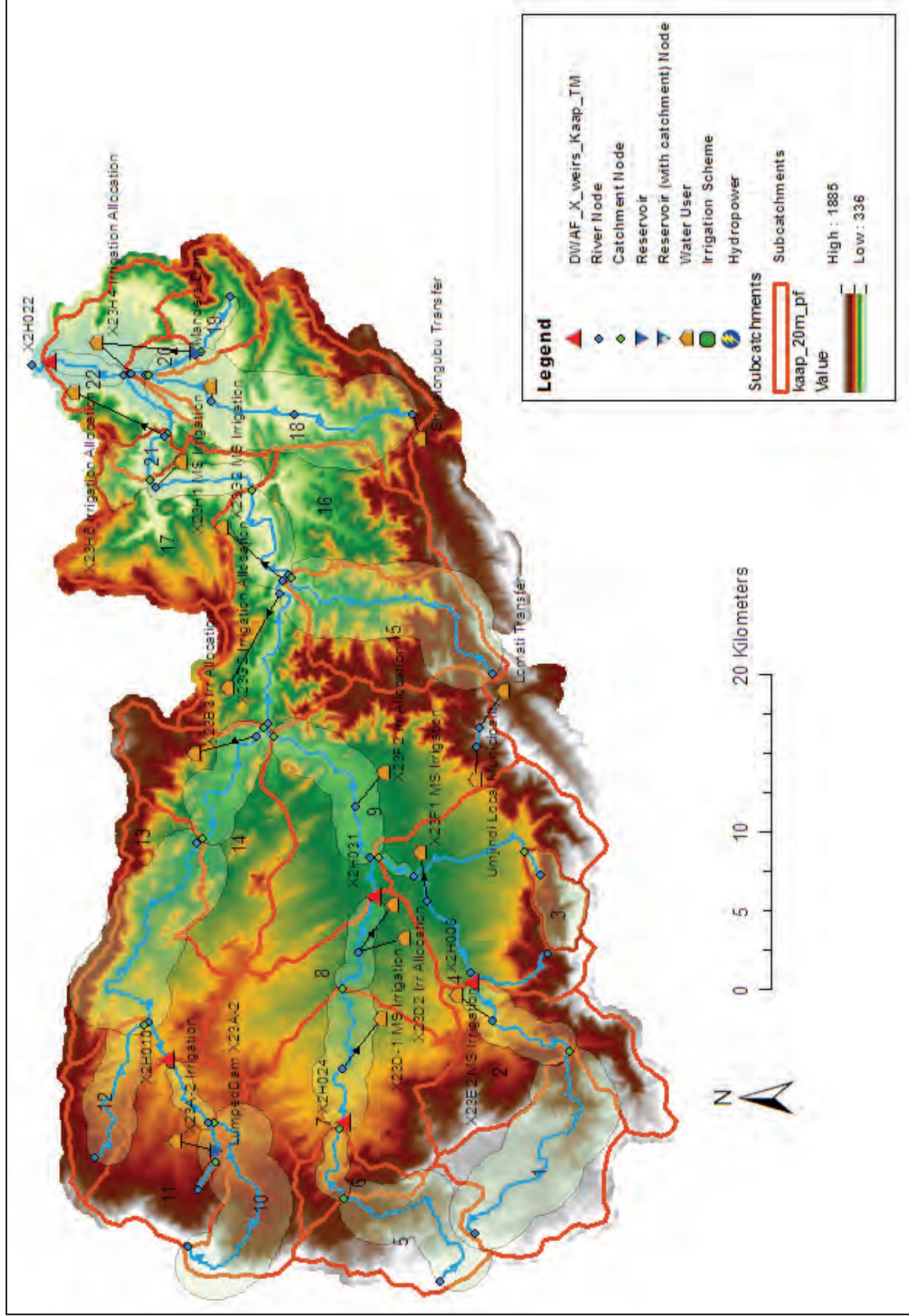


Figure 7.27 The MIKE BASIN configuration for the streamflow use case

## 7.4.2 Streamflow links with flow routing

The *ACRU* and MIKE BASIN configurations for this use case were almost identical to the previous use case (Section 7.4.1), except that in MIKE BASIN flow routing was turned on for the river reaches and the necessary flow routing variables were specified. Some minor changes were also required in the configuration of the river reaches as MIKE BASIN does not route flows down river reaches directly below confluences, water abstraction points and dams. To resolve this problem it was necessary to add river nodes so as to make short river reaches, in which no flow routing takes place, immediately downstream of confluences, water abstraction points and dams.

MIKE BASIN provides three different routing options: wave translation, linear routing and Muskingum routing. Initially it was proposed that the Muskingum routing option method would be used, however, the lengths of the river reaches together with the daily time step resulted in the Muskingum method being unsuitable, which led to a decision to use the simpler linear routing method for the purpose of this use case. The average slope of each river reach was estimated using the length of the reach and the start and end elevations estimated from the DEM for the catchment. The average slope for each river reach was used together with the Uplands nomograph (Schulze and Arnold (1979), cited by Schulze *et al.* (1992)), to determine the average flow velocity. The average flow velocity was multiplied by 11/9 (Viessman *et al.*, 1989) to determine the wave celerity in each reach. Finally, the flow routing delay factor *K* was estimated for each river reach by dividing the reach length by the wave celerity. In the MIKE BASIN configuration the delay parameter *K* was specified for each river reach.

This use case was successfully run for the simulation period 1990 to 1999, and the results are discussed in Section 7.4.4. This use case demonstrates the advantage of linking to MIKE BASIN to provide additional functionality and is expected to improve modelling results due to the lag and attenuation of flows in long river networks. Flow routing will be necessary when using the models for short term operational decisions.

### 7.4.3 Streamflow and irrigation links

In this use case the *ACRU* model is used to simulate runoff and irrigation requirements from the 22 subcatchments and these runoff and irrigation requirement values are then provided as input to the MIKE BASIN model. MIKE BASIN models all water allocation and non-irrigation water use, returns irrigation supply quantities to *ACRU*, and flows down the river network. The supplied irrigation quantities are applied to the irrigated areas modelled by *ACRU* and any return flows are included in the runoff quantities simulated by *ACRU*. The catchment configuration for MIKE BASIN is shown in Figure 7.28. The non-irrigation water users were the same as those modelled in the streamflow use case (Section 7.4.1). In the *ACRU* model, irrigation was turned on for all irrigated areas by switching the values in the *IrrigMonths (IRRMON)* variable to 1. To model irrigation from an external water supply in *ACRU* it was necessary to associate an *ExternalWaterSourceNode* component with each *Irrigated Area* component using the *WaterSupplyPath (WSPATH)* variable. In MIKE BASIN the irrigation water users were represented by a water user node in each catchment for which *ACRU* simulates irrigated land use. These irrigation water user nodes were assigned a water demand time series containing zero values which get overwritten by the irrigation water requirement quantities simulated by *ACRU*. In this use case bi-directional links were created between the *ACRU* model and the MIKE BASIN model to represent the irrigation demand and supply feedbacks. The links between the *ACRU* subcatchment *UnitRunoff (URFLOW)* variables and the MIKE BASIN catchment *Specific Runoff* variables were configured as listed in Table 7.5 for the streamflow use case. Links between the *ACRU* irrigated area *RequestQuantity (IRREQ)* variables and the MIKE BASIN water user node *Water Demand* variables, and links between the *ACRU* external water source node *ExternalWaterSourceQuantity (EXTWSQ)* variables and the MIKE BASIN supply channel *Flow* variables were configured as listed in Table 7.6.

This use case was successfully run for the simulation period 1990 to 1999, and the results are discussed in Section 7.4.4. This use case demonstrates the ability to represent irrigation feedbacks between the components modelled by the individual models, which would not be possible if the models were linked in series. In this case these two models both offer irrigation modelling functionality, but if *ACRU* were integrated with a river network model without irrigation modelling functionality, the parallel link would be necessary. Though the irrigation links configured for this use case, these links need to be investigated further to see if there is an easier way of setting these up and taking multiple water sources into account.

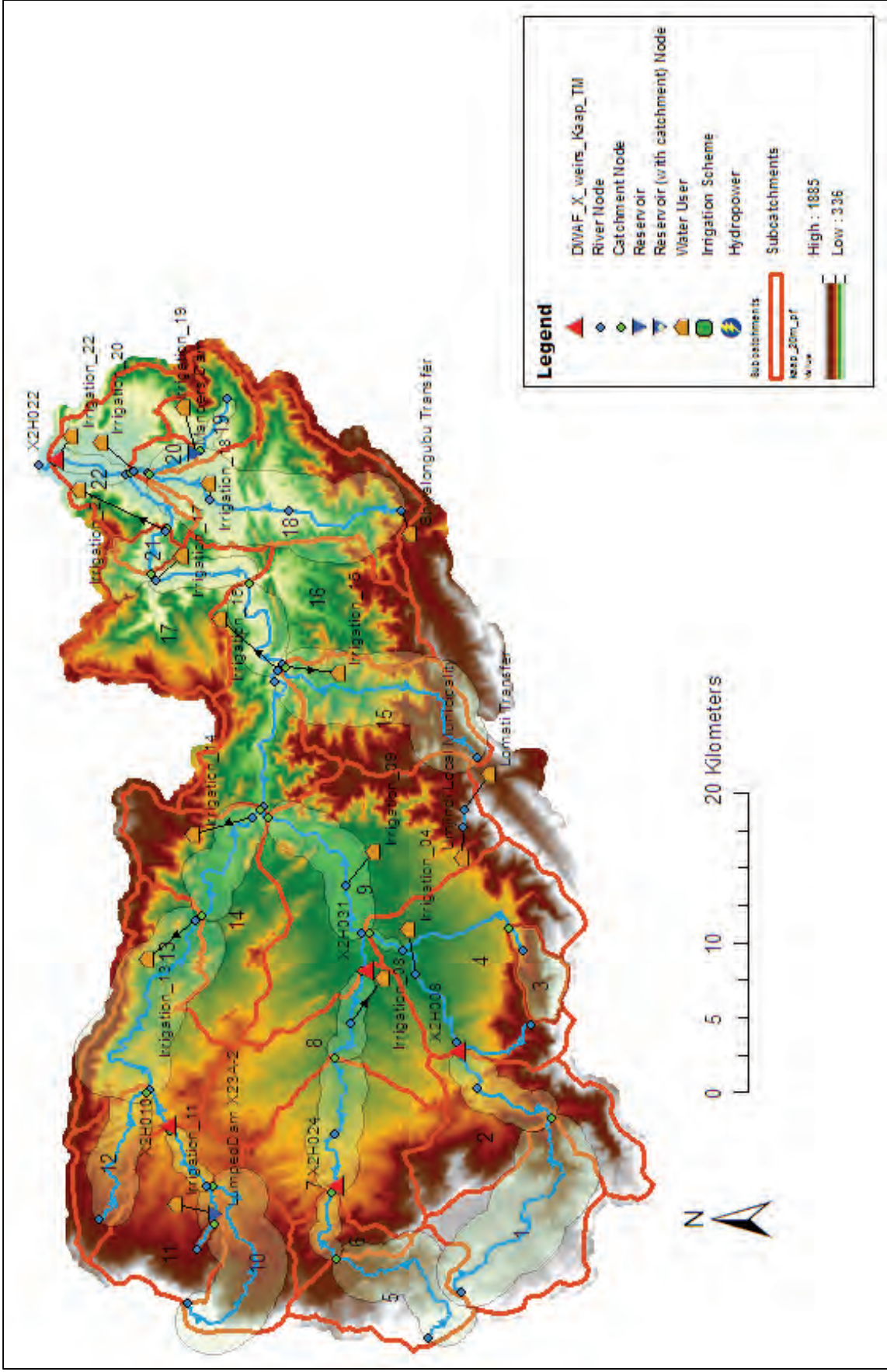


Figure 7.28 The MIKE BASIN configuration for the streamflow and irrigation use case

Table 7.6 The additional linked model variables for the irrigation use case

<b>ACRU</b>	<b>MIKE BASIN</b>
IrrigatedArea_04: RequestQuantity (m <sup>3</sup> /d)	Irrigation_04: Water Demand (m <sup>3</sup> /s)
04.ExternalWaterSourceNode_01: ExternalWaterSourceQuantity (m <sup>3</sup> /d)	Reach (E120): Flow (m <sup>3</sup> /s)
IrrigatedArea_08: RequestQuantity (m <sup>3</sup> /d)	Irrigation_08: Water Demand (m <sup>3</sup> /s)
08.ExternalWaterSourceNode_01: ExternalWaterSourceQuantity (m <sup>3</sup> /d)	Reach (E114): Flow (m <sup>3</sup> /s)
IrrigatedArea_09: RequestQuantity (m <sup>3</sup> /d)	Irrigation_09: Water Demand (m <sup>3</sup> /s)
09.ExternalWaterSourceNode_01: ExternalWaterSourceQuantity (m <sup>3</sup> /d)	Reach (E123): Flow (m <sup>3</sup> /s)
IrrigatedArea_11: RequestQuantity (m <sup>3</sup> /d)	Irrigation_11: Water Demand (m <sup>3</sup> /s)
11.ExternalWaterSourceNode_01: ExternalWaterSourceQuantity (m <sup>3</sup> /d)	Reach (E95): Flow (m <sup>3</sup> /s)
IrrigatedArea_13: RequestQuantity (m <sup>3</sup> /d)	Irrigation_13: Water Demand (m <sup>3</sup> /s)
13.ExternalWaterSourceNode_01: ExternalWaterSourceQuantity (m <sup>3</sup> /d)	Reach (E147): Flow (m <sup>3</sup> /s)
IrrigatedArea_14: RequestQuantity (m <sup>3</sup> /d)	Irrigation_14: Water Demand (m <sup>3</sup> /s)
14.ExternalWaterSourceNode_01: ExternalWaterSourceQuantity (m <sup>3</sup> /d)	Reach (E132): Flow (m <sup>3</sup> /s)
IrrigatedArea_15: RequestQuantity (m <sup>3</sup> /d)	Irrigation_15: Water Demand (m <sup>3</sup> /s)
15.ExternalWaterSourceNode_01: ExternalWaterSourceQuantity (m <sup>3</sup> /d)	Reach (E148): Flow (m <sup>3</sup> /s)
IrrigatedArea_16: RequestQuantity (m <sup>3</sup> /d)	Irrigation_16: Water Demand (m <sup>3</sup> /s)
16.ExternalWaterSourceNode_01: ExternalWaterSourceQuantity (m <sup>3</sup> /d)	Reach (E135): Flow (m <sup>3</sup> /s)
IrrigatedArea_17: RequestQuantity (m <sup>3</sup> /d)	Irrigation_17: Water Demand (m <sup>3</sup> /s)
17.ExternalWaterSourceNode_01: ExternalWaterSourceQuantity (m <sup>3</sup> /d)	Reach (E120): Flow (m <sup>3</sup> /s)
IrrigatedArea_18: RequestQuantity (m <sup>3</sup> /d)	Irrigation_18: Water Demand (m <sup>3</sup> /s)
18.ExternalWaterSourceNode_01: ExternalWaterSourceQuantity (m <sup>3</sup> /d)	Reach (E89): Flow (m <sup>3</sup> /s)
IrrigatedArea_19: RequestQuantity (m <sup>3</sup> /d)	Irrigation_19: Water Demand (m <sup>3</sup> /s)
19.ExternalWaterSourceNode_01: ExternalWaterSourceQuantity (m <sup>3</sup> /d)	Reach (E151): Flow (m <sup>3</sup> /s)
IrrigatedArea_20: RequestQuantity (m <sup>3</sup> /d)	Irrigation_20: Water Demand (m <sup>3</sup> /s)
20.ExternalWaterSourceNode_01: ExternalWaterSourceQuantity (m <sup>3</sup> /d)	Reach (E144): Flow (m <sup>3</sup> /s)
IrrigatedArea_21: RequestQuantity (m <sup>3</sup> /d)	Irrigation_21: Water Demand (m <sup>3</sup> /s)
21.ExternalWaterSourceNode_01: ExternalWaterSourceQuantity (m <sup>3</sup> /d)	Reach (E141): Flow (m <sup>3</sup> /s)
IrrigatedArea_22: RequestQuantity (m <sup>3</sup> /d)	Irrigation_22: Water Demand (m <sup>3</sup> /s)
22.ExternalWaterSourceNode_01: ExternalWaterSourceQuantity (m <sup>3</sup> /d)	Reach (E150): Flow (m <sup>3</sup> /s)

#### 7.4.4 Results

The simulation results for the three use cases (Streamflow, Flow Routing and Irrigation) were compared with the observed flows from Weir X2H022, which is situated near the exit of

the Kaap River Catchment, and with simulation results from *ACRU* on its own (*ACRU\_Standalone*). A graph of accumulated flow volumes for the full 10 year simulation period is shown in Figure 7.29. As expected the Streamflow and FlowRouting use cases produced similar flow volumes as the same catchment configurations were used, and flow routing simply lags and attenuates flows to some extent. Flow routing resulted in the peak flows being lagged by approximately one day. Also, as expected, the Irrigation and *ACRU\_Standalone* use cases produced similar flow volumes as the same catchment configurations were used, and importantly this confirms that the bi-directional links between the *ACRU* and MIKE BASIN models worked as expected. The difference in accumulated flows between the two different catchment configurations was larger than expected and is solely due to the difference in water abstracted from the river network by irrigation water users. This discrepancy in water use quantities highlights the difficulty of doing model verifications in developed operational catchments, especially those that contain dams and irrigated land use, and also highlights an area for further investigation for this catchment. Of even greater concern is the difference between observed and simulated flow volumes, especially in early 1996, though these results were not unexpected, given the results of the verification studies in the headwater subcatchments discussed in Section 7.3.

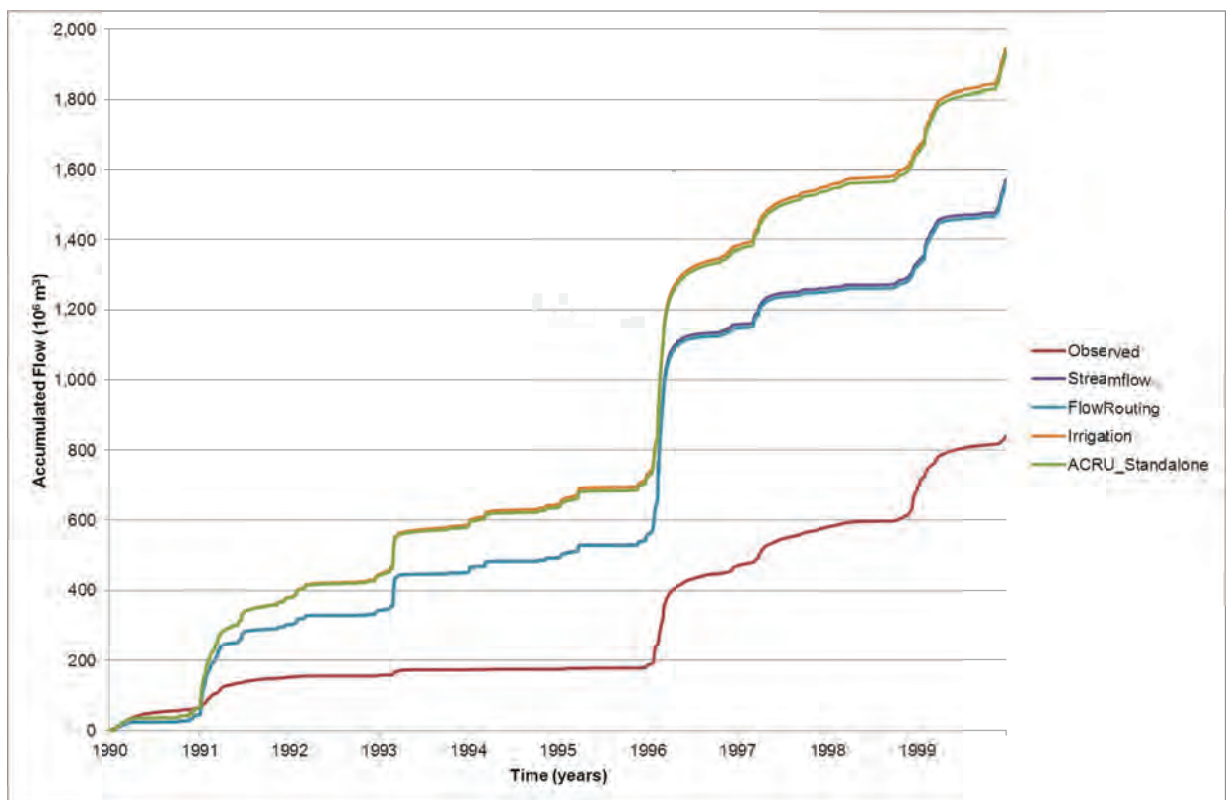


Figure 7.29 Accumulated flow ( $10^6 \text{ m}^3$ ) at Weir X2H022 for the period 1990 to 1999

The flow rates for the first four months of 1996 are shown in Figure 7.30. Severe flooding occurred in this region in February 1996 and observed data is missing between 12 and 14 February over the peak of the flood. The simulated flows start to increase earlier than the observed flows, which partly explains the difference in flow volumes over this period. The X2H022 weir was replaced in 2001 with a bigger weir with a structural capacity of 61.86 m<sup>3</sup>/s. The capacity of the weir at X2H022 prior to 2001 is not known, but it was noted that other marked differences between accumulated observed and simulated flows occur on days when the average flow rate exceeds approximately 20.0 m<sup>3</sup>/s. These observations lead to the conclusion that periods where observed average daily flows exceed 20.0 m<sup>3</sup>/s should be used with caution. The way in which the *ACRU* model simulates flows during high rainfall events in this catchment also needs to be investigated further. The accumulated flows for the period April 1996 to December 1999 are shown in Figure 7.31. For this period there appears to be a good correlation between observed flows and the simulated flows for the Streamflow and Flow Routing use cases. Again, differences in flow accumulation occurred when the average flow rates exceeded 20.0 m<sup>3</sup>/s, such as in March 1997, December 1998 to March 1999, and December 1999.

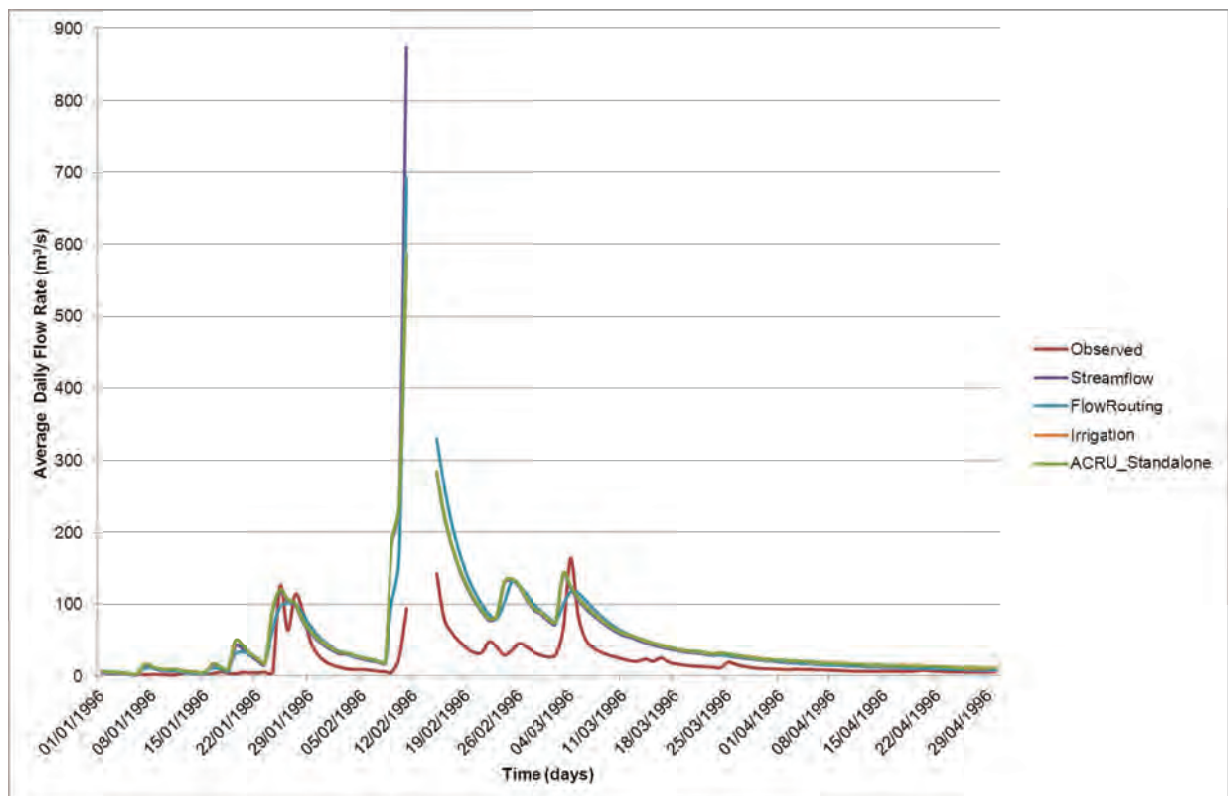


Figure 7.30 Flow rates for Weir X2H022 for early 1996



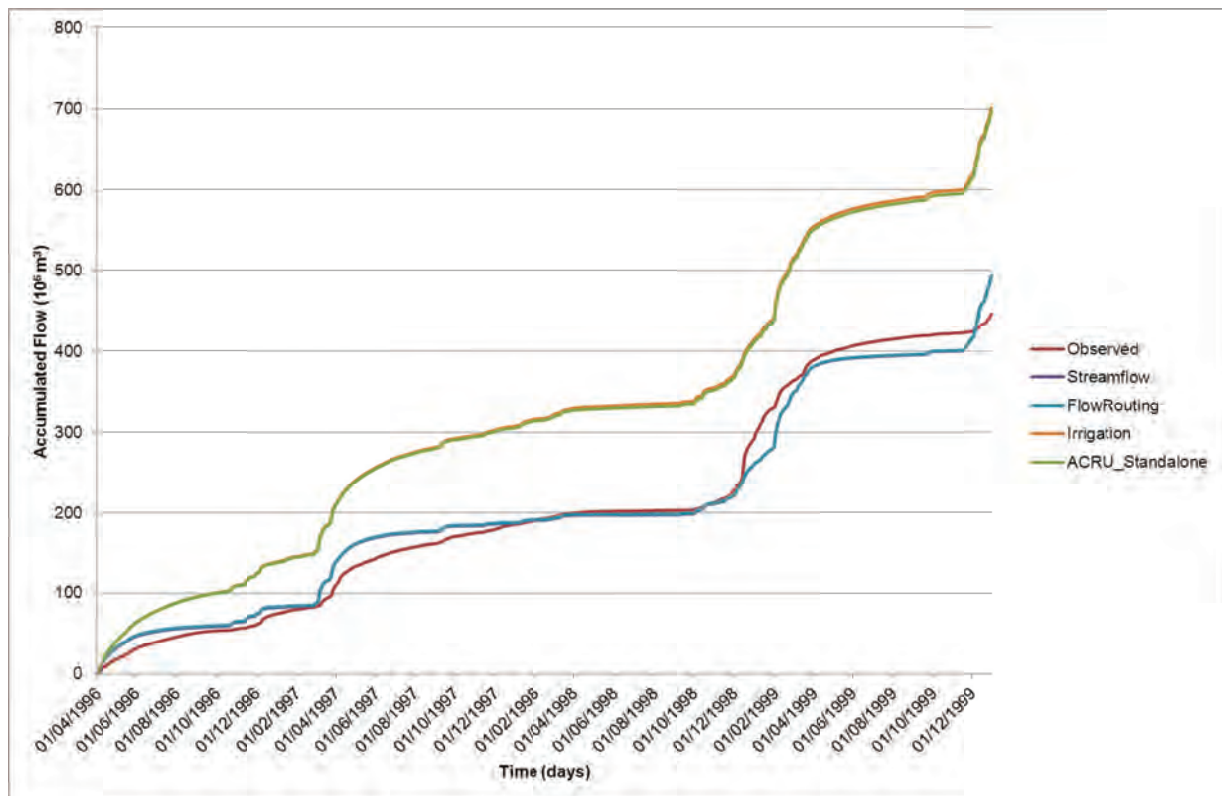


Figure 7.31 Accumulated flow ( $10^6 \text{ m}^3$ ) at Weir X2H022 for the period 1996 to 1999

Flow rates for the 1997-1998 rainy season are shown in Figure 7.32, and for the 1998-1999 rainy season is shown in Figure 7.33. The 1997-1998 rainy season is characterised by low observed peak flows and the 1998-1999 rainy season is characterised by a few relatively high peak flows. In a season with lower flows, the Irrigation use case results in a better simulation of flows than the Streamflow use case. This is due to actual irrigation requirements being modelled in the *ACRU* model on a day-by-day basis in the Irrigation use case, compared to the constant water use quantities used as input to MIKE BASIN for the Streamflow use case. In the 1998-1999 rainy season, when there is more water available, both in the soil and in the river network, this difference is less obvious and both the Streamflow and Irrigation use case simulations produce reasonable results.

In Figure 7.32 and Figure 7.33 it can be seen that in most cases the simulations represent the same high flow events as were observed, though the magnitude of the flows may be quite different. However, there are a few instances where the models are not simulating events that are apparent in the observed flows. This problem is possibly due to localised rainfall events that are not present in the rainfall time series for the rain gauges selected for use in these case studies.

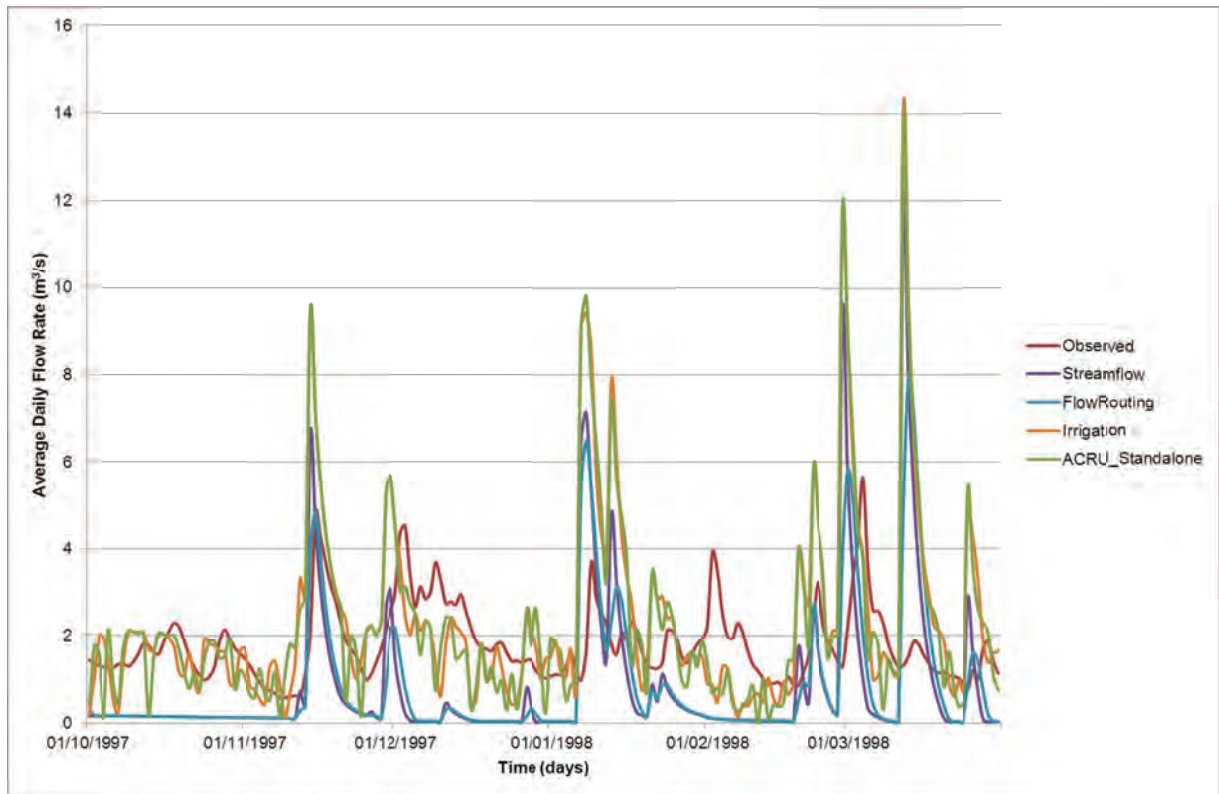


Figure 7.32 Flow rates for Weir X2H022 for the 1997-1998 rainy season

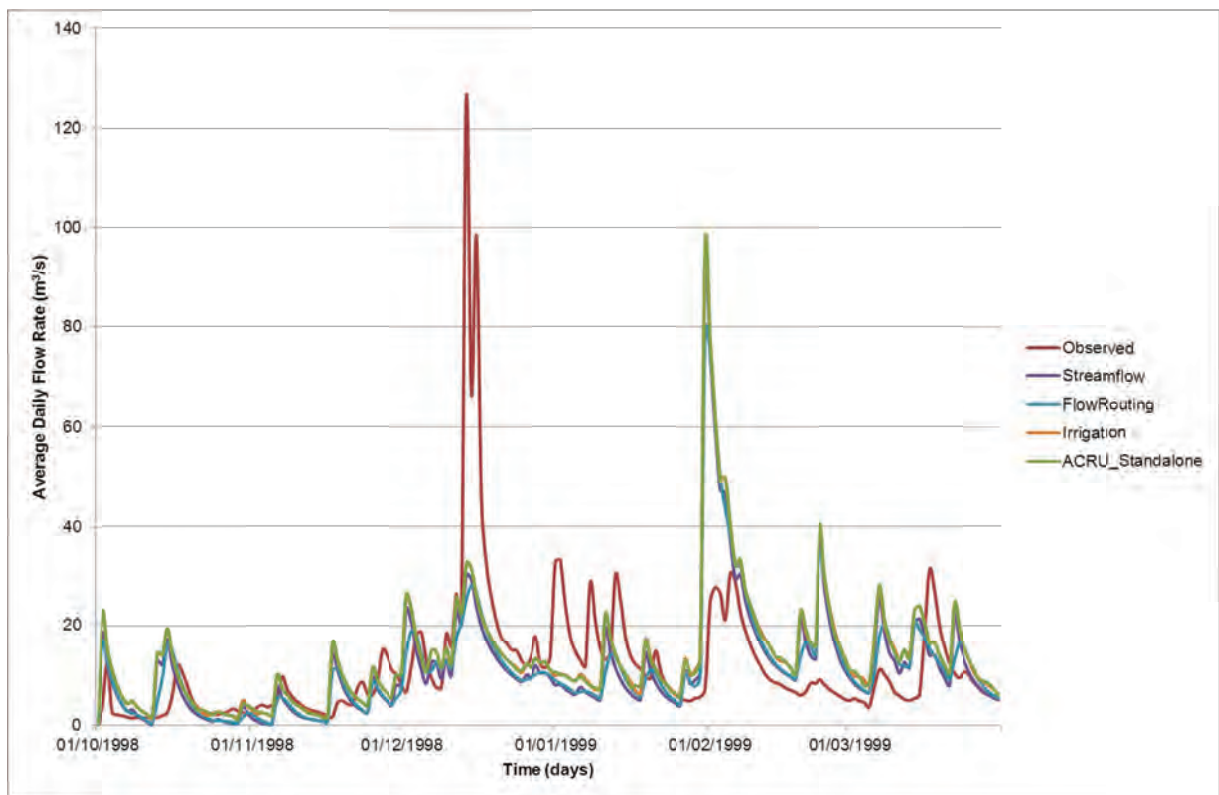


Figure 7.33 Flow rates for Weir X2H022 for the 1998-1999 rainy season

In addition to the over simulation of streamflow volumes, the simulated peak average daily flows are often significantly higher than the observed peaks. For large events this may be due to the capacity of the weir being exceeded, but parameters and variables in the *ACRU* model, such as soil properties and the quick flow response factor, that have an effect on the response characteristics of the catchment will also have an effect on the peak flows. One of the advantages of integrating the *ACRU* and MIKE BASIN models is to gain additional functionality such as flow routing. An example of the advantage of being able to do flow routing is shown in Figure 7.34, where the peak of the hydrograph in the Streamflow use case is too early and too high, and the hydrograph in the Flow Routing use case is much closer to the observed flow hydrograph. Flow routing can provide a significant improvement in representing event hydrographs, especially in big catchments, and can be important for operations modelling.

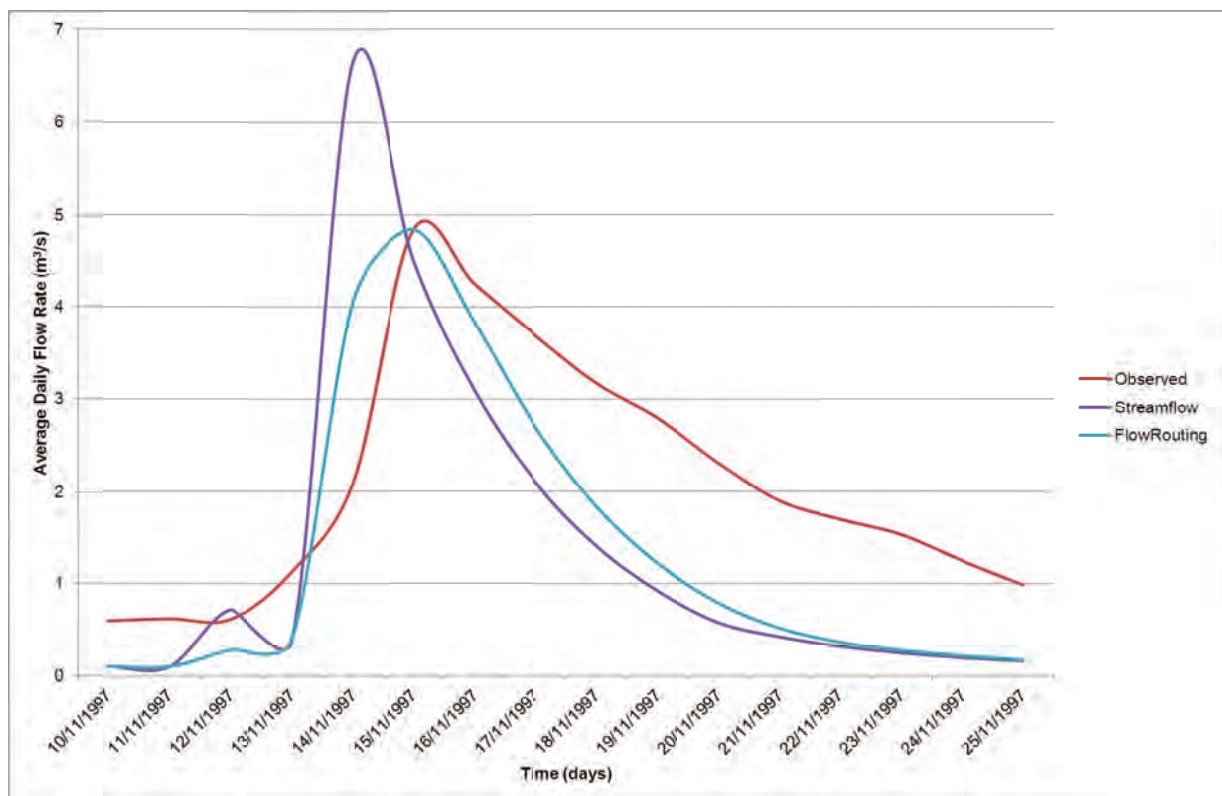


Figure 7.34 Example of the effect of flow routing on flow rates for Weir X2H022

In addition to possible errors in the measurement of high flows at Weir X2H022, some of the factors possibly affecting the simulations include the estimation of water use in the catchment, transfers into the catchment, local rainfall events not represented in the selected rainfall time series, static land cover for the duration of the simulation, and *ACRU* parameters and variables affecting the response of the catchment to rainfall events. Although the simulated streamflow volumes in these use cases do not closely match the

observed streamflow volumes, the objective of demonstrating the use of the integrated models in a real catchment has been achieved. Flow routing in MIKE BASIN enables better modelling of flow peaks, and modelling irrigation requirements in *ACRU* enables better modelling of irrigation water demand than the constant demand quantities use in the Streamflow used case.

## 8 DISCUSSION AND CONCLUSIONS

DJ Clark and JC Smithers

The modelling requirements for water resources planning and operations to meet the specifications of the National Water Act (NWA, 1998) of South Africa were identified by Pott *et al.* (2008b). It was recognised that it is unlikely that any one model would be able to meet all these requirements. To meet these requirements a collection of models covering all aspects (hydrology, environmental, economic and social) of water resource systems is required, and these models need to be integrated to model real world complexity and to ensure that any important feedbacks within the system are represented. The project thus aimed to demonstrate the integration of different domain models, with the linking of a hydrological model and a river network model, as a case study, in order to meet some of the modelling requirements identified for water resources planning and operations. The more specific objectives of this project were to:

- Review river network models which are suitable for water resource planning and operations and select one to be integrated with the *ACRU* hydrological model.
- Investigate methods for linking different domain models such as a hydrological model and a network model, and select a suitable method for integrating the hydrological model and the selected river network model.
- Further develop the *ACRU* daily time step hydrological model in order to realistically represent the varying hydrological responses within the terrestrial hydrological system.
- Configure and apply the integrated hydrological model and selected river network model for selected catchments within the Inkomati WMA.

There are a number of river network models available internationally. The initial review of these models, documented in Section 2.1, resulted in a recommendation that the MIKE BASIN, MODSIM and RiverWare models be evaluated in more detail. The result of this detailed evaluation, described in Section 2.2, was that the MIKE BASIN model was recommended for use in the project largely due to its ease of use, strong GIS support through ArcGIS and availability of local user support and training.

A review of available mechanisms for linking models was conducted and is documented in Section 3.1. Based on this review it was concluded that the OpenMI, TIME and OMS systems would be suitable for use in the project and OpenMI was selected for the following reasons: it is generally accepted as a *de facto* standard, is strongly supported by the

OpenMI Association, has been widely adopted by key research and commercial model developers, the provision of a useful set of compliant models, and is well documented.

Further development of the *ACRU* model and its associated model input files, as described in Chapter 4, has resulted in the model being better suited for use in both water resources planning and operations modelling and is now capable of more realistically representing real world complexity. Several changes were made to the *ModelData* and *ModelConfiguration* XML schemas used for *ACRU* model input and to the model itself to refine the design and to include new functionality such as scenario management, the storage of state variables required to hot-start the model, a means of storing dynamic data, use of forecast data and improved linkages to external data files. The revised *ModelData* and *ModelConfiguration* schemas are more robust and capable of providing the *ACRU* model with model input functionality necessary for both planning and operations modelling. The changes made to the *ACRU* model itself, included: a revised internal data structure, the concept of resources, and new functionality such as scenarios, hotstarting and the storage of state data, dynamic variables and flexible spatial component configurations.

The OpenMI model linkage framework was successfully implemented to create an OpenMI 1.4 .Net wrapper for the MIKE BASIN model and both OpenMI 1.4 Java and .Net wrappers for the *ACRU* model. Both models were structured such that it was relatively easy to make them OpenMI compliant using the wrapping tools provided by the OpenMI Association. An important lesson learned while setting up and testing the integrated models was that though OpenMI may make it easy to link compliant models, a detailed understanding of the models being linked is required to ensure that valid links are created without compromising the integrity of either model. To aid in the application of the integrated models a number of use cases have been described with details of which variables should be linked in each model and important points to note to ensure the models are correctly configured.

To demonstrate the application of the integrated models, the models were configured for the Kaap River Catchment which is part of the Inkomati WMA. The poor verifications of simulated streamflow against observed streamflow highlighted the need for more accurate data and at a finer spatial and temporal resolution, including: rainfall, streamflow, land cover, land use practices, soils, water transfers and water abstractions.

The project was successful in demonstrating the implementation of OpenMI by successfully linking the *ACRU* and MIKE BASIN models which represent two often separately modelled domains within water resource systems. The use of these linked models is expected to be a

useful tool for water resources modelling for planning and operations in South Africa. This project was a test case for model integration of legacy models using OpenMI and, given the successes achieved, there is no apparent technical reason why other models representing other domains cannot also be made OpenMI compliant. In addition to the fact that the *ACRU* model can now be easily linked with MIKE BASIN, OpenMI compliance means that these models can be linked to a range of other OpenMI compliant models, many from well-known developers of software for water resources modelling.

In this project, the advantages of linking models in parallel to provide a more holistic systems view of water resources and better representation of feedbacks between components in the different domains being modelled, were demonstrated. Some potential limitations of linking models include, the requirement for expert knowledge of all models to be linked, reduction in performance in running simulations, due to the linkage mechanism, and increased uncertainty in the simulation results introduced by linking the models.

## 9 RECOMMENDATIONS

DJ Clark and JC Smithers

This project has demonstrated that integration of independent domain models using OpenMI is possible, and has explained and demonstrated the advantages of model integration in being able to better represent real world complexity and thus to provide a systems view of water resource systems. The application of the integrated *ACRU* and MIKE BASIN models by users outside of the project team would not be easy, as an understanding of the OpenMI model linkage mechanism and the individual models is required. An open modelling environment named Delta Shell is being developed by the Dutch research institute Deltares. This integrated modelling environment will include OpenMI tools to enable models to be linked, but also facilitate communication between models and the modelling environment which will provide GIS, data management and analysis tools. Delta Shell should be investigated further once it is released, both for the modelling environment itself and the approach adopted to facilitate use of OpenMI.

The performance penalty and memory requirements when linking models using OpenMI needs to be further investigated. Once a stable version of OpenMI 2.0 SDK has been released by the OpenMI Association, the development of OpenMI 2.0 compliant wrappers for the *ACRU* and MIKE BASIN models should be considered as it is expected to offer better performance and improved user interface tools for linking models.

The integration of additional models, such as groundwater, water quality and economics models, using OpenMI would enable the OpenMI model linkage mechanism to be tested further. The integration of additional models, representing other domains, would also enable the investigation of the advantages and potential problems associated with modelling feedbacks between the various domains.

Considerable expertise has been developed through this project in the use of OpenMI to dynamically link legacy models. While the linked models have been demonstrated to operate on a real catchment, it is recommend that the expertise developed in the project be used to install and operationalize the linked models such that they can be used by water resource managers (e.g. by a CMA). It is anticipated that this will lead to further developments and refinements in order to meet the requirements of the water resource managers. This will also utilise the expertise developed during the project which, with no



follow up research or operationalization project, is in danger of dispersing and being lost to the water community in South Africa.

## 10 CAPACITY BUILDING

### DJ Clark and JC Smithers

The three staff employed by this project belong to the newly formed Centre for Water Resources Research (CWRR) at the University of KwaZulu-Natal, and as such are involved in assisting, advising and supervising postgraduate students from the disciplines of Hydrology and Bioresources Engineering.

This project has provided support for the four postgraduate students listed in Table 10.1. During this project the students have grown in knowledge and experience and are all due to submit their dissertations for examination in 2013.

Table 10.1 Students supported by the project.

Student	Full/Part Time	Degree
Mr DJ Clark	Part time	PhD Engineering
Mr A Lutchminarain	Part time	MSc Bioresources Systems
Mr SLC Thornton-Dibb	Part time	MSc Hydrology
Mr R Winckworth	Part time	MSc Hydrology

More specifically, considerable expertise has been developed during this project in the use of OpenMI to dynamically link legacy models, and in its application by integrating the *ACRU* and *MIKE BASIN* models. The integration of models representing different domains within water resource systems is seen to be a key requirement in providing water managers with the tools and information necessary to make sound decisions, taking into consideration environmental, economic and social aspects of water as required by the National Water Act of South Africa. To the best of the author's knowledge this is the first time that OpenMI has been applied in South Africa, which means that as a result of this project, important capacity has been developed in South Africa and not just at the University of KwaZulu-Natal. The reviews of river network models and model linkage mechanisms are also expected to serve as a useful reference for other water resources modellers in South Africa. The Kaap River Catchment in the Inkomati WMA was used for the case study in this project and the Inkomati CMA has expressed interest in building capacity in the use of the integrated *ACRU* and *MIKE BASIN* models. This will enable expertise developed during the project to be used, which, with no follow up research or operationalization project, is in danger of dispersing and being lost to the water community in South Africa.

## 11 REFERENCES

- Ahuja, LR, Ascough II, JC and David, O. 2005. Developing natural resource models using the object modelling system: feasibility and challenges. *Adv. Geosci.* 4: 29-36.
- Argent, R, Perraud, JM, Rahman, J, Grayson, R and Podger, G. 2009. A new approach to water quality modelling and environmental decision support systems. *Environmental Modelling & Software* 24 (7): 809-818.
- Argent, R and Rizzoli, A. 2004. Development of multi-framework model components. 365-370. Citeseer.
- Armstrong, R, Kumfert, G, McInnes, LC, Parker, S, Allan, B, Sottile, M, Epperly, T and Dahlgren, T. 2006. The CCA component model for high-performance scientific computing. *Concurrency and Computation: Practice and Experience* 18 (2): 215-229.
- Ascough, JC, David, O, Krause, P, Fink, M, Kralisch, S, Kipkac, H and Wetzel, M. 2010. Integrated Agricultural System Modelling Using OMS 3: Component Driven Stream Flow and Nutrient Dynamics Simulations. In: eds. Swayne, D, Yang, W, Voinov, A, Rizzoli, A and Filatova, T, *International Environmental Modelling and Software Society (iEMSs), 2010 International Congress on Environmental Modelling and Software – Modelling for Environment’s Sake*, Ottawa, Canada.
- Assaf, H, van Beek, E, Borden, C, Gijsbers, P, Jolma, A, Kaden, S, Kaltofen, M, Labadie, JW, Loucks, DP, Quinn, NWT, Sieber, J, Sulis, A, Werick, WJ and Wood, DM. 2008. Chapter Thirteen Generic Simulation Models for Facilitating Stakeholder Involvement in Water Resources Planning and Management: A Comparison, Evaluation, and Identification of Future Needs. In: eds. A.J. Jakeman, AAVAER and Chen, SH, *Developments in Integrated Environmental Assessment*. Volume 3. Elsevier.
- Barthel, R, Göttinger, J, Hartmann, G, Jagelke, J, Rojanschi, V and Wolf, J. 2006. Integration of hydrological models on different spatial and temporal scales. *Advances in Geosciences* 9: 1.
- Basson, MS, Allen, RB, Pegram, GGS and van Rooyen, JA. 1994. *Probabilistic Management of Water Resources and Hydropower Systems*. Water Resources Publications, Highlands Ranch, Colorado, USA.
- Biddle, SH. 2001. Optimizing the TVA Reservoir System Using RiverWare. In: eds. Don, P and Gerald, S, 149. ASCE.
- Blind, M, Dirksen, F, Gavardinas, C, Gijsbers, P, Gregersen, J and Westen, S. 2005. OpenMI Document Series: Part B Guidelines for the OpenMI (version 1.0). [Internet]. The OpenMI Association. Available from: [https://sites.google.com/a/openmi.org/home/learning-more/B\\_Guidelines.pdf?attredirects=0](https://sites.google.com/a/openmi.org/home/learning-more/B_Guidelines.pdf?attredirects=0). [Accessed: 25 August 2012].
- Blind, M and Gregersen, J. 2005. Towards an Open Modelling Interface (OpenMI) the HarmonIT project. *Advances in Geosciences* 4: 69-74.
- BoM. 2010a. Australian Hydrological Geospatial Fabric (Geofabric) Product Guide. [Internet]. Bureau of Meteorology. Available from: <http://www.bom.gov.au/water/geofabric/documentation.shtml>. [Accessed: 14 December 2010].
- BoM. 2010b. The Australian Water Resources Information System. [Internet]. Bureau of Meteorology. Available from: [www.bom.gov.au/water/about/publications/document/InfoSheet\\_3.pdf](http://www.bom.gov.au/water/about/publications/document/InfoSheet_3.pdf). [Accessed: 14 December 2010].
- Bramley, R, Chiu, K, Diwan, S, Gannon, D, Govindaraju, M, Mukhi, N, Temko, B and Yechuri, M. 2000. A component based services architecture for building distributed applications. 51. Published by the IEEE Computer Society.
- Branger, F, Braud, I, Debionne, S, Viallet, P, Dehotin, J, Henine, H, Nedelec, Y and Anquetin, S. 2010a. Towards multi-scale integrated hydrological models using the

- LIQUID (R) framework. Overview of the concepts and first application examples. *Environmental Modelling & Software* 25 (12): 1672-1681.
- Branger, F, Debionne, S, Viallet, P, Braud, I, Jankowfsky, S, Vannier, O, Rodriguez, F and Anquetin, S. 2010b. Advances in integrated hydrological modelling with the LIQUID framework. In: eds. Swayne, D, Yang, W, Voinov, A, Rizzoli, A and Filatova, T, *International Environmental Modelling and Software Society (iEMSs), 2010 International Congress on Environmental Modelling and Software – Modelling for Environment’s Sake*, Ottawa, Canada.
- Bulatewicz, T, Yang, X, Peterson, JM, Staggenborg, S, Welch, SM and Steward, DR. 2010. Accessible integration of agriculture, groundwater, and economic models using the Open Modelling Interface (OpenMI): methodology and initial results. *Hydrology and Earth System Sciences* 14 (3): 521-534.
- CADSWES. 2010a. RIVERWARE : Technical Documentation Version 6.0, Batch Mode and RCL. Center for Advanced Decision Support for Water and Environmental Systems, The University of Colorado, Colorado, USA.
- CADSWES. 2010b. RIVERWARE : Technical Documentation Version 6.0, Data Management Interface. Technical Documentation. Center for Advanced Decision Support for Water and Environmental Systems, The University of Colorado, Colorado, USA.
- CADSWES. 2011. RiverWare version 6.0.3 – Mar 8 2011 10:53:10 [Software]. [Software]. Center for Advanced Decision Support for Water and Environmental Systems, Boulder, Colorado, USA.
- Castronova, AM and Goodall, JL. 2010. A generic approach for developing process-level hydrologic modelling components. *Environmental Modelling & Software* 25 (7): 819-825.
- Chetty, K. 2009. An Assessment Of Scale Issues Related To The Configuration Of The ACRU Model For Design Flood Estimation. Unpublished Unpublished MSc Thesis, School of Bioresources Engineering and Environmental Hydrology, Pietermaritzburg, South Africa.
- Christensen, FD. 2004. Coupling Between the River Basin Management model (MIKE BASIN) and the 3D Hydrological Model (MIKE SHE) with use of the OpenMI System. In: eds. Liong, S, Phoon, K and Babovic , V, *6th International Conference of Hydroinformatics*, Singapore.
- Clark, DJ, Smithers, JC, Hughes, DA, Meier, KB, Summerton, MJ and Butler, AJE. 2009. *Design and Development of a Hydrological Decision Support Framework*. WRC Report No. 1490/1/09. Water Research Commission, Pretoria, South Africa.
- Collins, N, Theurich, G, DeLuca, C, Suarez, M, Trayanov, A, Balaji, V, Li, P, Yang, W, Hill, C and da Silva, A. 2005. Design and Implementation of Components in the Earth System Modelling Framework. *International Journal of High Performance Computing Applications* 19 (3): 341-350.
- Cook, F, Jordan, P, Waters, D and Rahman, J. 2009. WaterCAST – Whole of Catchment Hydrology Model: An Overview. 3492-3498.
- Dahmann, JS, Fujimoto, RM and Weatherly, RM. 1997. The Department of Defense High Level Architecture. IEEE Computer Society.
- David, O, Ascough, J, Leavesley, G and Ahuja, L. 2010. Rethinking Modelling Framework Design: Object Modelling System 3.0. In: eds. Swayne, D, Yang, W, Voinov, A, Rizzoli, A and Filatova, T, *International Environmental Modelling and Software Society (iEMSs), 2010 International Congress on Environmental Modelling and Software – Modelling for Environment’s Sake*, Ottawa, Canada.
- David, O, Lloyd, W, Carlson, J, Leavesley, G and Geter, F. 2009. Use of Annotations for Component and Framework Interoperability. 1044.
- David, O, Schneider, I and Leavesley, G. 2004. Metadata and modelling frameworks: The object modelling system example. *Transactions of the 2nd Biennial Meeting of the International Environmental Modelling and Software Society, iEMSs 2004*, Osnabrück, Germany, 439-443. Citeseer.

- Delgado, P, Kelley, P and Murray, N. 2011. Source User Guide. eWater Cooperative Research Centre, Canberra, Australia.
- Deltares. 2010. RIBASIM. [Internet]. Available from: <http://www.wldelft.nl/soft/ribasim/int/index.html>. [Accessed: 15 February ].
- DHI. 2009. MIKE BASIN 2009 with service pack 5 [Software]. [Software]. Mike by DHI, Horsholm, Denmark.
- DHI. 2010. GIS-based water resource management. [Internet]. Available from: <http://mikebydhi.com/sitecore/content/Microsites/MIKEbyDHI/Products/WaterResources/MIKEBASIN.aspx>. [Accessed: 15 December 2010].
- DHI. 2011a. Mike Basin Training Manual. DHI. Water and Environment, Johannesburg, RSA.
- DHI. 2011b. Mike Basin Training Manual. DHI South Africa 1-3 June 2011. DHI South Africa., Johannesburg, South Africa.
- DHI. 2011c. MIKE BASIN User Manual. DHI, Horsholm, Denmark.
- DHI. 2011d. Mike by DHI. [Internet]. Available from: <http://www.mikebydhi.com/>. [Accessed: 14 October 2011].
- Donchyts, G, Hummel, S, Vaneček, S, Groos, J, Harper, A, Knapen, R, Gregersen, J, Schade, P, Antonello, A and Gijssbers, P. 2010. OpenMI 2.0 – What's new? In: eds. Swayne, D, Yang, W, Voinov, A, Rizzoli, A and Filatova, T, *International Environmental Modelling and Software Society (iEMSs) 2010 International Congress on Environmental Modelling and Software Modelling for Environment's Sake*, Ottawa, Canada.
- Donchyts, G and Jagers, B. 2010. DeltaShell – an open modelling environment. In: eds. Swayne, D, Yang, W, Voinov, A, Rizzoli, A and Filatova, T, *International Environmental Modelling and Software Society (iEMSs), 2010 International Congress on Environmental Modelling and Software – Modelling for Environment's Sake*, Ottawa, Canada.
- DWA. 2012. Department of Water Affairs, Hydrological Services – Surface Water (Data, Dams, Floods and Flows). [Internet]. Available from: <http://www.dwaf.gov.za/Hydrology>. [Accessed: 11 May 2012].
- DWAF. 2004. *National Water Resources Strategy*. Department of Water Affairs and Forestry, Pretoria, RSA.
- DWAF. 2009. *Inkomati Water Availability Assessment*. Report No. PWMA 05/X22/00/0808. Pretoria, R.S.A.
- EPA. 2007. BASINS. [Internet]. Available from: <http://water.epa.gov/scitech/datatit/models/basins/>. [Accessed: 14 December 2010].
- EPA. 2010. Better Assessment Science Integrating Point and Nonpoint Sources BASINS Version 4.0. [Software]. EPA.
- Ershadi, A, Khiabani, H. and Lorup, J.K. 2005. Applications of remote sensing, GIS and river basin modelling in integrated water resource management of Kabul River Basin. *ICID 21st European Regional Conference*, Frankfurt, Germany and Slubice, Poland.
- Evans, T, Oakley, B, Cotter, J and Zagona, E. 2006. INTEGRATION OF RIVERWARE INTO THE CORPS WATER MANAGEMENT SYSTEM. [Internet]. Hydrologic Engineering Center, US Army Corps of Engineers. Center for Advanced Decision Support for Water and Environmental Systems, University of Colorado. Available from: [http://www.gcmrc.gov/library/reports/physical/Fine\\_Sed/8thFISC2006/3rdFIHMC/8E\\_Evans.pdf](http://www.gcmrc.gov/library/reports/physical/Fine_Sed/8thFISC2006/3rdFIHMC/8E_Evans.pdf). [Accessed: 15 July 2011].
- eWater. 2010a. eWater Source. [Internet]. eWater CRC. Available from: <http://www.ewater.com.au/products/ewater-source/>. [Accessed: 21 Feb 2010].
- eWater. 2010b. *Source Catchments User Guide*. eWater Cooperative Research Centre, Canberra, Australia. Available from: [Accessed: ]
- Fischer, C, Kralisch, S, Krause, P, Fink, M and Flügel, WA. 2009. Calibration of hydrological model parameters with the JAMS framework. In: eds. Anderssen, RS, Braddock, RD and Newham, LTH, *18th World IMACS / MODSIM Congress*, Cairns, Australia.

- Frezghi, MS. 2007. Water Resoure System Yield Assesement. Unpublished Unpublished PhD Seminar, School of Bioresources Engineering and Environmental Hydrology, School of Bioresources Engineering and Environmental Hydrology, University of KwaZulu-Natal, Pietermartizburg, South Africa.
- Frezghi, MS. 2012a. E-mail Communication | DHI Kaap Catchment MIKE BASIN setup. Johannesburg, RSA. 2012/11/05.
- Frezghi, MS. 2012b. E-mail Communication | RE: Coverages. Johannesburg, RSA. 2012/05/14.
- Gijsbers, P, Gregersen, J, Westen, S, Dirksen, F, Gavardinas, C and Blind, M. 2005. OpenMI Document Series: Part B Guidelines for the OpenMI (version 1.0). [Internet]. Stephen Morris, Butford Technical Publishing Ltd. Available from: <http://www.openmi.org/reloaded/about/publications-documents.php>. [Accessed: 17 February 2011].
- Gijsbers, P, Hummel, S, S., V, Groos, J, Harper, A, Knapen, R, Gregersen, J, Schade, P, Antonello, A and Donchyts, G. 2010. From OpenMI 1.4 to 2.0.
- Gregersen, JB, Gijsbers, PJA and Westen, SJP. 2007. OpenMI: Open modelling interface. *Journal of Hydroinformatics* 9 (3): 175-191.
- Gregersen, JB, Gijsbers, PJA, Westen, SJP and Blind, M. 2005. OpenMI: the essential concepts and their implications for legacy software. *Adv. Geosci.* 4: 37-44.
- Hallowes, J. 2011. Personal communication via e-mail on 05/08/2011. DHI. SA. Johannesburg, RSA. 05/08/2011.
- Haumann, J, K. 2008. *Crocodile (East) River Development, Reconnaissance Study*. PD Naidoo & Associates (Pty) Ltd.
- HEC-5. 1998. HEC-5. [Internet]. US Army Corp of Engineers. Available from: <http://www.hec.usace.army.mil/software/legacysoftware/hec5/hec5.htm>. [Accessed: 07 February].
- HEC-ResSim. 2010. HEC-ResSim. [Internet]. US Army Corp of Engineers. Available from: <http://www.hec.usace.army.mil/software/hec-ressim/index.html>. [Accessed: 10 September ].
- Hill, C, DeLuca, C, Balaji, Suarez, M and Da Silva, A. 2004. The architecture of the Earth System Modelling Framework. *Computing in Science & Engineering* 6 (1): 18-28.
- Hofman, D. 2005. LIANA Model Integration System – architecture, user interface design and application in MOIRA DSS. *Adv. Geosci.* 4: 9-16.
- Hutchings, C, Struve, J, Westen, S, Millard, K and Fortune, D. 2002. State of the Art Review, Work Package 1, IT Frameworks (HarmoniIT). [Internet]. Available from: [http://www.harmonit.org/docs/repu\\_01\\_09\\_soa\\_review\\_approved.pdf](http://www.harmonit.org/docs/repu_01_09_soa_review_approved.pdf). [Accessed: 28 February 2011].
- Jackson, B. 2009. Personal communication on 11/01/2009. Council for Scientific and Industrial Research (CSIR). Pretoria, RSA. 2009.
- Jackson, B. 2012. Personal communication on 05/12/2012.
- Jagers, H. 2010. Linking Data, Models and Tools: An Overview. In: eds. Swayne, D, Yang, W, Voinov, A, Rizzoli, A and Filatova, T, *International Environmental Modelling and Software Society (iEMSs), 2010 International Congress on Environmental Modelling and Software – Modelling for Environment’s Sake*, Ottawa, Canada.
- Jankowfsky, S, Branger, F, Braud, I, Viallet, P, Debionne, S and Rodriguez, F. 2010. Development of a suburban catchment model within the LIQUID® framework. In: eds. Swayne, D, Yang, W, Voinov, A, Rizzoli, A and Filatova, T, *International Environmental Modelling and Software Society (iEMSs), 2010 International Congress on Environmental Modelling and Software – Modelling for Environment’s Sake*, Ottawa, Canada.
- Kiker, GA. 2001. ACRU2000 Model. In: eds. Lynch, SD and Kiker, GA, *ACRU Model Development and User Support*. WRC Report No. 636/1/00. Water Research Commission, Pretoria, South Africa.

- Kiker, GA, Clark, DJ, Martinez, CJ and Schulze, RE. 2006. A Java-based, Object-Oriented Modelling System for Southern African Hydrology. *Transactions of the ASABE* 49 (5): 1419-1433.
- Kime, DB. 2010. The Development and Assessment of a Prototype Water Accounting System for South Africa Using the *ACRU2000* and MIKE BASIN Models. Unpublished Dissertation, School of Bioresources Engineering and Environmental Hydrology, University of KwaZulu-Natal, Pietermaritzburg.
- Knapen, M, Verweij, P, Wien, J and Hummel, S. 2009. OpenMI – The universal glue for integrated modelling? *18th World IMACS Congress and MODSIM09 International Congress on Modelling and Simulation*, Cairns, Australia.
- Kralisch, S. 2011. Personal communication via e-mail on 3/3/2011.
- Kralisch, S and Krause, P. 2006. JAMS – A Framework for Natural Resource Model Development and Application. In: eds. Gourbesville, P, Cunge, J, Guinot, V and Liong, SY, *7th International Conference on Hydroinformatics*, Nice, France.
- Kralisch, S, Krause, P and David, O. 2005. Using the object modelling system for hydrological model development and application. *Advances in Geosciences* 4 (1-2): 75-81.
- Kralisch, S, Krause, P, Fink, M, Fischer, C and Flügel, W. 2007. Component based environmental modelling using the JAMS framework. In: eds. Oxley, L and Kulasiri, D, *MODSIM 2007 International Congress on Modelling and Simulation, Modelling and Simulation Society of Australia and New Zealand*, Christchurch, New Zealand.
- Kralisch, S, Zander, F and Krause, P. 2009. Coupling the RBIS Environmental Information System and the JAMS Modelling Framework. In: eds. Anderssen, RS, Braddock, RD and Newham, LTH, *18th World IMACS / MODSIM Congress*, Cairns, Australia.
- Krause, P, Kralisch, S, Flügel, W, Haas, A, Jaeger, C, Hofman, D, Pullar, D, Lotze-Campen, H, Lucht, W and Müller, C. 2005. Model integration and development of modular modelling systems. *Advances in Geosciences* 4: 1-2.
- Labadie, JW. 2004. Optimal Operation of Multireservoir Systems: State-of-the-Art Review. *Journal of Water Resources Planning and Management* 130 (2): 93-111.
- Labadie, JW. 2005. Optimal Operation of Multireservoir Systems: State-of-the-Art Review. *Journal of Water Resources Planning and Management* 131 (5): 407-408.
- Labadie, JW. 2006a. MODSIM: Decision Support System for Integrated River Basin Management. In: eds. Singh, VP and Frevert, DK, *Watershed Models*. Taylor and Francis Group, New York.
- Labadie, JW. 2006b. MODSIM: Decision Support System for Integrated River Basin Management. In: eds. Voinov, A, Jakeman, AJ and Rizzoli, AE, *Proceedings of the iEMSs Third Biennial Meeting: "Summit on Environmental Modelling and Software"*. Burlington, USA. International Environmental Modelling and Software Society.
- Labadie, JW. 2010a. MODSIM 8.1: River Basin Management Decision Support System [Software]. [Software]. Department of Civil and Environmental Engineering, Colorado State University, Fort Collins, USA.
- Labadie, JW. 2010c. MODSIM 8.1: River Basin Management Decision Support System, Tutorials and Example Networks. Training Manual. Department of Civil and Environmental Engineering, Colorado State University, Fort Collins, Colorado, USA.
- Labadie, JW. 2011. E-mail Communication. MODSIM-DSS | Colorado State University, USA.
- Labadie, JW, Fontane, DG, Lee, J-H and Ko, IH. 2007. Decision Support System for Adaptive River Basin Management: Application to the Geum River Basin, Korea. *Water International* 32 (3): Pg. 397-414.
- Labadie, JW. 2006. MODSIM: Decision Support System for Integrated River Basin Management. the International Environmental Modelling and Software Society.
- Leavesley, G, Markstrom, S, Restrepo, P and Viger, R. 2002. A modular approach to addressing model design, scale, and parameter estimation issues in distributed hydrological modelling. *Hydrological Processes* 16 (2): 173-187.

- Lecler, N. 2004. Fractional Water Allocation and Capacity Sharing/Water Banking. *South African Sugar Technologists' Association – Proceedings (78)*: 6.
- Lévite, H, Sally, H and Cour, J. 2002. Water demand management scenarios in a water-stressed basin in South Africa.
- Lindenschmidt, KE, Hesser, FB and Rode, M. 2005. Integrating water quality models in the High Level Architecture (HLA) environment. *Adv. Geosci.* 4: 51-56.
- Lloyd, W, David, O, Ascough II, JC, Rojas, KW, Carlson, JR, Leavesley, GH, Krause, P, Green, TR and Ahuja, LR. 2009. An exploratory investigation on the invasiveness of environmental modelling frameworks. 909-915. TR, Ahuja, LR 2009. An Exploratory Investigation on the Invasiveness of Environmental Modelling Frameworks. World IMACS Congress and MODSIM09 International Congress on Modelling and Simulation.
- Lynch, SD. 2004. *Development of a Raster Database of Annual, Monthly and Daily Rainfall for Southern Africa*. Report 1156/1/04. Pretoria, R.S.A.
- Lynch, SD and Kiker, GA. 2001. *ACRU Model Development and User Support*. WRC Report No. 636/1/01. Water Research Commission, Pretoria, South Africa.
- Makropoulosa, C, Safioleaa, E, Bakia, S, Doukaa, E, Stamoua, A and Mimikou, M. 2010. An integrated, multi-modelling approach for the assessment of water quality: lessons from the Pinios River case in Greece. In: eds. Swayne, D, Yang, W, Voinov, A, Rizzoli, A and Filatova, T, *International Environmental Modelling and Software Society (iEMSs), 2010 International Congress on Environmental Modelling and Software – Modelling for Environment's Sake*, Ottawa, Canada.
- Manqoyi, N and Nyabeze, WR. 2006. *Situational Assessment of Operating Rules for Existing DWAF Dams*. Makgaleng Projects CC, Midrand, RSA.
- Markstrom, S. 2011. Personal communication via e-mail on 2/3/2011.
- Marston, F, Argent, R, Vertessy, R, Cuddy, S and Rahman, J. 2002. *The Status of Catchment Modelling in Australia*. Cooperative Research Centre for Catchment Hydrology.
- McCartney, M and Arranz, R. 2009. Evaluation of water demand Scenarios for the Olifants River catchment, South Africa. *International Journal of River Basin Management* 7 (4): 379-390.
- Mckenzie, RS and Van Rooyen, PG. 2003. *Management of large water resources systems*. Water Resources Planning Pty Ltd, Pretoria, RSA.
- Miller, A. 2011. E-mail Communication. eWater Cooperative Research Centre, Australia.
- Moore, RV and Tindall, Cl. 2005. An overview of the open modelling interface and environment (the OpenMI). *Environmental Science & Policy* 8 (3): 279-286.
- Murray, N, Perraud, J-M, Rahman, J, Bridgart, R, Davis, G, Watson, F and Hotham, H. 2007. *TIME Workshop Notes 4.2*. CSIRO Land and Water, and eWater CRC.
- Nadomba, PM, Azab, A, Mulungu, MM, Malisa, J, Yousif, N, Atti, MA, Latief, AA, Deogratias, M, Attia, AEA, Sayed, M and Gad, M. 2005. *GIS-based watershed modelling in the Nile basin countries*. Nile Basin Capacity Building Network for River Engineering.
- Neumann, D. 2011a. Personal communication via e-mail on 16/10/2011. Center for Advanced Decision Support for Water and Environmental Systems, The University of Colorado, Colorado, USA. 16/10/2011.
- Neumann, D. 2011b. Personal communication via e-mail on 23/09/2011. Center for Advanced Decision Support for Water and Environmental Systems, The University of Colorado, Colorado, USA. 23/09/2011.
- NWA. 1998. *National Water Act (Act No. 36 of 1998)*. Government Printers, Pretoria, South Africa.
- OATC. 2010. OpenMI Document Series: Migrating Models for the OpenMI (Version 2.0). [Internet]. The OpenMI Association Technical Committee. Available from: <https://4310b1a9-a-cb397f23-s-sites.googlegroups.com/a/openmi.org/home/learning-more/OpenMI%2BStandard%2B2%2BMigrating%2BModels.pdf>. [Accessed: 01 March 2012].



- OATC. 2012. New to OpenMI ? [Internet]. The OpenMI Association Technical Committee. Available from: <https://sites.google.com/a/openmi.org/home/new-to-openmi#TOC-Downloads>. [Accessed: 03 July 2012].
- OpenMI. 2011a. OpenMI Standard Specification. [Internet]. OpenMI Association. Available from: <http://www.openmi.org/reloaded/standard/specification.php>. [Accessed: 28 February 2011].
- OpenMI. 2011b. OpenMI Compliant Software. [Internet]. OpenMI Association. Available from: <http://www.openmi.org/reloaded/users/compliant-software.php>. [Accessed: 28 February 2011].
- OpenMI. 2012. What's New in OpenMI 2.0. [Internet]. OpenMI Association. Available from: <https://4310b1a9-a-cb397f23-s-sites.google.com/a/openmi.org/home/learning-more/WhatsNewinOpenMI2.pdf>. [Accessed: 01 March 2012].
- Perera, BJC, James, B and Kularathna, MDU. 2005. Computer software tool REALM for sustainable water allocation and management. *Journal of Environmental Management* 77 (4): 291-300.
- Pott, AJ. 2011. DHI presentations uploaded to youtube for students. Johannesburg, RSA. 2011/10/10.
- Pott, AJ, Benadé, N, van Heerden, P, Grové, B, Annandale, J and Steyn, M. 2008a. *Technology Transfer and Integrated Implementation of Water Management Models in Commercial Farming*. WRC Report No TT 267/08. Water Research Commission, Pretoria, RSA.
- Pott, AJ, Hallowes, JS, Mtshali, SS and Smithers, JC. 2008b. *A Review of National Water Resource Planning for Operational Needs*. WRC Consultancy Report No K8 /740/01. Water Research Commission, Pretoria, RSA.
- Rahman, J, Perraud, J, Hotham, H, Murray, N, Leighton, B, Freebairn, A, Davis, G and Bridgart, R. 2005. Evolution of TIME. In: eds. Zenger, A and Argent, RM, *MODSIM 2005 International Congress on Modelling and Simulation. Modelling and Simulation Society of Australia and New Zealand*, 697-703.
- Rahman, J, Seaton, S, Perraud, J, Hotham, H, Verrelli, D and Coleman, J. 2003. It's TIME for a new environmental modelling framework. 1727-1732.
- Rahman, JM, Seaton, SP and Cuddy, SM. 2004. Making frameworks more useable: using model introspection and metadata to develop model processing tools. *Environmental Modelling & Software* 19 (3): 275-284.
- Reußner, F, Alex, J, Bach, M, Schütze, M and Muschalla, D. 2009. Basin-wide integrated modelling via OpenMI considering multiple urban catchments. *Water Science & Technology* 60 (5): 1241-1248.
- Schellekens, J, Sprengers, CJ and Gijsbers, PJA. 2003. *Water system research Yellow River Basin Improving agro-hydrological models*. Delft University of Technology, Delft, Netherlands.
- Schreidera, SY, Jamesb, B, Sekerc, MP and Weinmanna, PE. 2003. Sensitivity and Error Propagation Analysis for the Goulburn Simulation Model built by REALM. The Modelling and Simulation Society of Australia and New Zealand Inc.
- Schulze, R and Arnold, H. 1979. *Estimation of Volume and Rate of Runoff in Small Catchments in South Africa, based on the SCS Technique*. Agricultural Catchments Research Unit, Report No. 40. Department of Agricultural Engineering, University of Natal, Pietermaritzburg, South Africa.
- Schulze, RE. 1975. Catchment evapotranspiration in the Natal Drakensburg. Unpublished PhD thesis. Department of Geography, University of Natal, Pietermaritzburg, South Africa.
- Schulze, RE. 1995. Verification Studies. In: ed. Schulze, RE, *Hydrology and Agrohydrology: A Text to Accompany the ACRU 3.00 Agrohydrological Modelling System*. Report TT69/95, Water Research Commission, Pretoria, R.S.A.
- Schulze, RE, Angus, G, R. and Guy, RM. 1995a. In: Schulze, R.E. *Hydrology and Agrohydrology: A Text to Accompany the ACRU 3.00 Agrohydrological Modelling System*. Report TT69/95. Water Research Commission, Pretoria, R.S.A.

- Schulze, RE, Angus, GR, Lynch, SD and Smithers, JC. 1995b. ACRU: Concepts and Structure. In: ed. Schulze, RE, *Hydrology and Agrohydrology: A Text to Accompany the ACRU 3.00 Agrohydrological Modelling System*. WRC Report No. TT69/95. Water Research Commission, Pretoria, South Africa.
- Schulze, RE, Horan, MJC, Kunz, RP, Lumsden, TG and Knoesen, DM. 2010. Development of the Southern African Quinary Catchments Database. In: eds. Schulze, RE, Hewitson, BC, Barichiev, KR, Tadross, MA, Kunz, RP, Horan, MJC and Lumsden, TG, *Methodological Approaches to Assessing Eco-Hydrological Responses to Climate Change in South Africa*. Water Research Commission, Pretoria, RSA, WRC Report 1562/1/10. Chapter 7.
- Schulze, RE, Schmidt, EJ and Smithers, JC. 1992. *SCS-SA user manual. PC-based SCS design flood estimates for small catchments in Southern Africa*. Agricultural Catchments Research Unit, Report No. 40. Department of Agricultural Engineering, University of Natal, Pietermaritzburg, South Africa.
- SEI. 2011. Welcome to WEAP! [Internet]. Stockholm Environment Institute. Available from: <http://www.weap21.org/index.asp>. [Accessed: 21 Feb].
- Smithers, JC and Caldecott, RE. 1995. Hydrograph routing. In: ed. Schulze, RE, *Hydrology and Agrohydrology : A Text to Accompany the ACRU 3.00 Agrohydrological Modelling System*. Report TT69/95. Water Research Commission, Pretoria.
- Tollenaar, D. 2009. Simulation of present and future discharges at the Nile River upstream Lake Nasser. Unpublished Water Engineering & Management, University of Twente, Netherlands.
- Triana, E and Labadie, JW. 2007. GEO-MODSIM: Spatial Decision Support System for River Basin Management. *ESRI International User Conference*, San Diego Convention Center, San Diego, California.
- Triana, E, Labadie, JW and Gates, TK. 2010. River GeoDSS for Agroenvironmental Enhancement of Colorado's Lower Arkansas River Basin. I : Model Development and Calibration. *Journal of Water Resources Planning and Management* 136 (2):
- Valerio, AM. 2008. Modelling Groundwater-Surface Water Interactions in an Operational Setting by Linking RiverWare with MODFLOW. Unpublished Department of Civil, Environmental, and Architectural Engineering, University of Colorado, Colorado, USA,
- van der Krogt, WNM. 2011. E-mail Communication. Deltares | delft hydraulics, Delft, the Netherlands.
- Viessman, W, Lewis, GL and Knapp, JW. 1989. *Introduction to Hydrology*. Harper and Row, New Yourk, USA.
- Wallbrink, P. 2008. A New National River Modelling Platform. *Water | Journal of the Australian Water Association* 35 (7): 6-8.
- Welsh, W. 2011. E-mail Communication. CSIRO | Land and Water, Australia.
- Welsh, WD and Black, D. 2010. Engaging stakeholders for a software development project: River Manager model. [Internet]. Available from: <http://www.iemss.org/iemss2010/proceedings.html>. [Accessed:
- Welsh, WD and Podger, GM. 2008. *Australian Hydrological Modelling Initiative: River System Management Tool (AHMI: RSMT) functionality specifications*. eWater Technical Report, eWater Cooperative Research Centre, Canberra.
- Wurbs, RA. 2005. *Comparative Evaluation of Generalized Reservoir/River System Models*. Technical Report No. 282. Department of Civil Engineering, Texas University, Texas, USA.
- Zagona, E, Kandl, E, Carron, J and Bowser, S. 2010. Water Accounting and Allocation in Riverware.
- Zagona, EA. 2011. Personal communication via e-mail on 18/07/2011. Center for Advanced Decision Support for Water and Environmental Systems, The University of Colorado, Colorado, USA. 18/07/2011.

- Zagona, EA, Fulp, TJ, Shane, R, Magee, T and Goranflo, HM. 2001. RiverWare: A Generalised Tool for Complex Reservoir System Modelling. *Journal of the American Water Resources Association* 37 (4): 913-929.
- Zagona, EA, Fulp, TJ, Goranflo, HM and Shane, RM. 1998. RiverWare: A General River and Reservoir Modelling Environment. *Proceedings of the First Federal Interagency Hydrologic Modelling Conference*, Las Vegas, Nevada, USA.
- Zhou, S. 2006. Coupling climate models with the earth system modelling framework and the common component architecture. *Concurrency and Computation: Practice and Experience* 18 (2): 203-213.

## APPENDIX:

### Appendix A – Details Of How The Subcatchments Were Further Subdivided

The *Area* column represents the total area for each of these representative HRUs.

Table A.1 Details of the *Components* contained within each subcatchment

Subcatchment	Component Type	Description	Area (km <sup>2</sup> )
01	HRU	Forest Plantations (clear-felled)	1.929
	HRU	Forest Plantations (Eucalyptus spp)	7.429
	HRU	Forest Plantations (Other / mixed spp)	6.456
	HRU	Forest Plantations (Pine spp)	45.476
	HRU	Thicket, Bushland, Bush Clumps, High Fynbos	4.968
	HRU	Unimproved (natural) Grassland	20.432
	Dam	Waterbodies	0.022
	Wetland	Wetlands	0.011
02	HRU	Cultivated, temporary, commercial, dryland	0.073
	HRU	Forest (indigenous)	1.230
	HRU	Forest Plantations (clear-felled)	1.361
	HRU	Forest Plantations (Eucalyptus spp)	35.945
	HRU	Forest Plantations (Other / mixed spp)	6.589
	HRU	Forest Plantations (Pine spp)	7.466
	HRU	Thicket, Bushland, Bush Clumps, High Fynbos	17.564
	HRU	Unimproved (natural) Grassland	23.445
03	HRU	Degraded Unimproved (natural) Grassland	0.123
	HRU	Forest Plantations (clear-felled)	1.236
	HRU	Forest Plantations (Eucalyptus spp)	1.297
	HRU	Forest Plantations (Other / mixed spp)	0.171
	HRU	Forest Plantations (Pine spp)	0.514
	HRU	Mines & Quarries (surface-based mining)	0.051
	HRU	Thicket, Bushland, Bush Clumps, High Fynbos	5.384
	HRU	Unimproved (natural) Grassland	6.763
	HRU	Urban / Built-up (residential)	0.075
04	IrrigArea	Cultivated, commercial, irrigated	0.094
	HRU	Cultivated, permanent, commercial, sugarcane	0.009
	HRU	Cultivated, temporary, commercial, dryland	7.409
	HRU	Degraded Thicket, Bushland, etc.	0.158
	HRU	Degraded Unimproved (natural) Grassland	0.871
	HRU	Forest Plantations (Acacia spp)	0.258
	HRU	Forest Plantations (clear-felled)	1.023

	HRU	Forest Plantations (Eucalyptus spp)	4.482
	HRU	Forest Plantations (Other / mixed spp)	0.748
	HRU	Forest Plantations (Pine spp)	5.023
	HRU	Improved Grassland	0.442
	HRU	Mines & Quarries (surface-based mining)	0.184
	HRU	Thicket, Bushland, Bush Clumps, High Fynbos	86.593
	HRU	Unimproved (natural) Grassland	12.464
	HRU	Urban / Built-up (residential)	3.549
	HRU	Urban / Built-up (residential, formal suburbs)	0.678
	HRU	Urban / Built-up, (industrial / transport : light)	0.257
	Dam	Waterbodies	0.115
	Wetland	Wetlands	0.047
	HRU	Woodland (previously termed Forest and Woodland)	2.607
	05	HRU	Forest (indigenous)
HRU		Forest Plantations (clear-felled)	3.528
HRU		Forest Plantations (Eucalyptus spp)	19.542
HRU		Forest Plantations (Other / mixed spp)	6.920
HRU		Forest Plantations (Pine spp)	11.801
HRU		Thicket, Bushland, Bush Clumps, High Fynbos	5.161
HRU		Unimproved (natural) Grassland	10.394
HRU		Urban / Built-up (residential)	0.058
Dam		Waterbodies	0.256
06	HRU	Forest Plantations (clear-felled)	1.982
	HRU	Forest Plantations (Eucalyptus spp)	13.205
	HRU	Forest Plantations (Other / mixed spp)	2.451
	HRU	Forest Plantations (Pine spp)	1.238
	HRU	Thicket, Bushland, Bush Clumps, High Fynbos	1.224
	HRU	Unimproved (natural) Grassland	2.832
07	HRU	Cultivated, temporary, commercial, dryland	3.321
	HRU	Forest Plantations (clear-felled)	4.942
	HRU	Forest Plantations (Eucalyptus spp)	50.429
	HRU	Forest Plantations (Other / mixed spp)	2.415
	HRU	Forest Plantations (Pine spp)	0.170
	HRU	Thicket, Bushland, Bush Clumps, High Fynbos	29.527
	HRU	Unimproved (natural) Grassland	7.248
	Dam	Waterbodies	0.238
08	Wetland	Wetlands	0.132
	HRU	Cultivated, temporary, commercial, dryland	10.860
	IrrigArea	Cultivated, temporary, commercial, irrigated	0.414
	HRU	Forest Plantations (clear-felled)	0.637
	HRU	Forest Plantations (Eucalyptus spp)	18.602
	HRU	Forest Plantations (Other / mixed spp)	0.163
	HRU	Thicket, Bushland, Bush Clumps, High Fynbos	51.957

	HRU	Unimproved (natural) Grassland	0.068
	Dam	Waterbodies	0.122
	Wetland	Wetlands	0.095
	HRU	Woodland (previously termed Forest and Woodland)	0.530
09	HRU	Bare Rock and Soil (natural)	0.049
	HRU	Cultivated, permanent, commercial, sugarcane	6.051
	HRU	Cultivated, temporary, commercial, dryland	5.195
	IrrigArea	Cultivated, commercial, irrigated	3.806
	HRU	Degraded Thicket, Bushland, etc.	1.094
	HRU	Degraded Unimproved (natural) Grassland	1.895
	HRU	Forest Plantations (Acacia spp)	0.025
	HRU	Forest Plantations (Eucalyptus spp)	0.640
	HRU	Forest Plantations (Pine spp)	0.088
	HRU	Mines & Quarries (mine tailings, waste dumps)	0.264
	HRU	Mines & Quarries (surface-based mining)	0.505
	HRU	Thicket, Bushland, Bush Clumps, High Fynbos	104.64
	HRU	Unimproved (natural) Grassland	22.509
	HRU	Urban / Built-up (residential)	2.824
	HRU	Urban / Built-up (residential, formal suburbs)	2.853
	HRU	Urban / Built-up, (commercial, mercantile)	0.332
	HRU	Urban / Built-up, (industrial / transport : light)	0.175
	Dam	Waterbodies	0.162
	Wetland	Wetlands	0.043
		HRU	Woodland (previously termed Forest and Woodland)
10	HRU	Cultivated, temporary, commercial, dryland	0.038
	HRU	Forest (indigenous)	8.948
	HRU	Forest Plantations (clear-felled)	3.598
	HRU	Forest Plantations (Eucalyptus spp)	17.766
	HRU	Forest Plantations (Other / mixed spp)	3.284
	HRU	Forest Plantations (Pine spp)	6.464
	HRU	Thicket, Bushland, Bush Clumps, High Fynbos	2.770
	HRU	Unimproved (natural) Grassland	8.729
	Dam	Waterbodies	0.022
	Wetland	Wetlands	0.018
11	HRU	Cultivated, temporary, commercial, dryland	4.918
	IrrigArea	Cultivated, temporary, commercial, irrigated	0.231
	HRU	Forest (indigenous)	5.540
	HRU	Forest Plantations (clear-felled)	1.075
	HRU	Forest Plantations (Eucalyptus spp)	30.223
	HRU	Forest Plantations (Other / mixed spp)	6.663
	HRU	Forest Plantations (Pine spp)	6.995
	HRU	Thicket, Bushland, Bush Clumps, High Fynbos	16.210
HRU	Unimproved (natural) Grassland	3.185	

	HRU	Urban / Built-up, (industrial / transport : light)	0.097
	Dam	Waterbodies	0.020
	Wetland	Wetlands	0.020
12	HRU	Cultivated, temporary, commercial, dryland	0.930
	HRU	Forest Plantations (clear-felled)	0.226
	HRU	Forest Plantations (Eucalyptus spp)	19.045
	HRU	Forest Plantations (Other / mixed spp)	4.685
	HRU	Forest Plantations (Pine spp)	2.771
	HRU	Thicket, Bushland, Bush Clumps, High Fynbos	5.301
	HRU	Unimproved (natural) Grassland	0.964
13	HRU	Cultivated, temporary, commercial, dryland	5.877
	IrrigArea	Cultivated, temporary, commercial, irrigated	3.521
	HRU	Forest (indigenous)	17.791
	HRU	Forest Plantations (Eucalyptus spp)	2.805
	HRU	Forest Plantations (Other / mixed spp)	0.690
	HRU	Forest Plantations (Pine spp)	0.133
	HRU	Thicket, Bushland, Bush Clumps, High Fynbos	61.604
	HRU	Unimproved (natural) Grassland	1.978
	Dam	Waterbodies	0.120
	Wetland	Wetlands	0.047
	HRU	Woodland (previously termed Forest and Woodland)	2.721
14	HRU	Bare Rock and Soil (erosion : dongas / gullies)	0.034
	HRU	Bare Rock and Soil (natural)	0.019
	HRU	Cultivated, permanent, commercial, sugarcane	0.032
	HRU	Cultivated, temporary, commercial, dryland	1.674
	IrrigArea	Cultivated, temporary, commercial, irrigated	2.571
	HRU	Degraded Thicket, Bushland, etc.	0.134
	HRU	Forest (indigenous)	0.228
	HRU	Forest Plantations (Acacia spp)	0.014
	HRU	Forest Plantations (clear-felled)	0.427
	HRU	Forest Plantations (Eucalyptus spp)	3.013
	HRU	Forest Plantations (Pine spp)	0.061
	HRU	Mines & Quarries (mine tailings, waste dumps)	0.042
	HRU	Mines & Quarries (surface-based mining)	1.287
	HRU	Thicket, Bushland, Bush Clumps, High Fynbos	77.854
	Dam	Waterbodies	0.077
	Wetland	Wetlands	0.010
	HRU	Woodland (previously termed Forest and Woodland)	10.446
	15	HRU	Bare Rock and Soil (natural)
HRU		Cultivated, permanent, commercial, sugarcane	0.085
IrrigArea		Cultivated, temporary, commercial, irrigated	0.564
HRU		Degraded Unimproved (natural) Grassland	1.890
HRU		Forest Plantations (Acacia spp)	0.011

	HRU	Forest Plantations (clear-felled)	0.354
	HRU	Forest Plantations (Eucalyptus spp)	0.097
	HRU	Forest Plantations (Pine spp)	0.278
	HRU	Mines & Quarries (surface-based mining)	0.303
	HRU	Thicket, Bushland, Bush Clumps, High Fynbos	45.882
	HRU	Unimproved (natural) Grassland	26.378
	Dam	Waterbodies	0.024
16	HRU	Bare Rock and Soil (natural)	0.040
	HRU	Cultivated, permanent, commercial, sugarcane	5.813
	HRU	Cultivated, temporary, commercial, dryland	0.059
	IrrigArea	Cultivated, commercial, irrigated	0.908
	HRU	Degraded Thicket, Bushland, etc.	1.364
	HRU	Degraded Unimproved (natural) Grassland	0.780
	HRU	Forest Plantations (Eucalyptus spp)	0.103
	HRU	Forest Plantations (Pine spp)	0.021
	HRU	Mines & Quarries (surface-based mining)	0.085
	HRU	Thicket, Bushland, Bush Clumps, High Fynbos	92.485
	HRU	Unimproved (natural) Grassland	35.952
	Dam	Waterbodies	0.246
	HRU	Woodland (previously termed Forest and Woodland)	11.352
17	HRU	Bare Rock and Soil (natural)	0.047
	HRU	Cultivated, permanent, commercial, sugarcane	1.196
	IrrigArea	Cultivated, commercial, irrigated	2.847
	HRU	Degraded Thicket, Bushland, etc.	1.638
	HRU	Thicket, Bushland, Bush Clumps, High Fynbos	56.735
	HRU	Unimproved (natural) Grassland	0.428
	Dam	Waterbodies	0.011
	HRU	Woodland (previously termed Forest and Woodland)	18.41
18	IrrigArea	Cultivated, permanent, commercial, irrigated	0.839
	HRU	Cultivated, permanent, commercial, sugarcane	11.452
	HRU	Degraded Thicket, Bushland, etc.	0.954
	HRU	Degraded Unimproved (natural) Grassland	1.094
	HRU	Forest Plantations (Acacia spp)	0.010
	HRU	Forest Plantations (Eucalyptus spp)	11.343
	HRU	Forest Plantations (Pine spp)	0.648
	HRU	Mines & Quarries (surface-based mining)	0.329
	HRU	Thicket, Bushland, Bush Clumps, High Fynbos	72.292
	HRU	Unimproved (natural) Grassland	8.305
	Dam	Waterbodies	0.116
	Wetland	Wetlands	0.011
HRU	Woodland (previously termed Forest and Woodland)	2.778	
19	HRU	Bare Rock and Soil (natural)	0.026
	IrrigArea	Cultivated, permanent, commercial, irrigated	0.052



	HRU	Cultivated, permanent, commercial, sugarcane	0.027
	HRU	Degraded Thicket, Bushland, etc.	0.922
	HRU	Forest Plantations (Eucalyptus spp)	0.903
	HRU	Forest Plantations (Pine spp)	0.026
	HRU	Thicket, Bushland, Bush Clumps, High Fynbos	23.04
	HRU	Unimproved (natural) Grassland	4.129
	Dam	Waterbodies	0.151
	HRU	Woodland (previously termed Forest and Woodland)	0.764
20	IrrigArea	Cultivated, permanent, commercial, irrigated	0.277
	HRU	Cultivated, permanent, commercial, sugarcane	1.461
	HRU	Degraded Thicket, Bushland, etc.	0.573
	HRU	Thicket, Bushland, Bush Clumps, High Fynbos	6.203
	Dam	Waterbodies	0.009
	HRU	Woodland (previously termed Forest and Woodland)	2.494
21	HRU	Degraded Thicket, Bushland, etc.	0.013
	HRU	Thicket, Bushland, Bush Clumps, High Fynbos	6.258
	Dam	Waterbodies	0.131
	HRU	Woodland (previously termed Forest and Woodland)	2.771
22	IrrigArea	Cultivated, permanent, commercial, irrigated	0.789
	HRU	Cultivated, permanent, commercial, sugarcane	6.841
	HRU	Degraded Thicket, Bushland, etc.	4.575
	HRU	Degraded Unimproved (natural) Grassland	0.236
	HRU	Forest Plantations (Eucalyptus spp)	0.498
	HRU	Thicket, Bushland, Bush Clumps, High Fynbos	38.201
	HRU	Unimproved (natural) Grassland	3.387
	Dam	Waterbodies	0.190
	Wetland	Wetlands	0.017
	HRU	Woodland (previously termed Forest and Woodland)	9.614

## **Appendix B – Electronic Appendix**

Appendix B is an electronic appendix on the CD accompanying this report. Appendix B includes the OpenMI wrappers developed for the *ACRU* and MIKE BASIN models, configuration files for the *ACRU* and MIKE BASIN models for the use case simulations in the Kaap River Catchment and data used for the simulations.

In addition to the software included in Appendix B, it will be necessary to obtain third party software libraries and applications in order to link and run the Kaap River Catchment configurations for *ACRU* and MIKE BASIN. The websites where these software libraries and applications can be found are as follows:

- The MIKE BASIN software and a licence may be obtained from DHI [<http://www.dhigroup.com>]
- The OpenMI 1.4 Standard, and .Net and Java implementations of this, can be obtained from the OpenMI Association [<http://www.openmi.org>]
- The OpenMI 1.4 SDK for .Net can be obtained from the OpenMI Association [<http://www.openmi.org>]
- The OpenMI 1.4 SDK for Java, developed by Alterra can be obtained from SourceForge [<http://openmi.svn.sourceforge.net/viewvc/openmi/branches/OpenMI-Version-1-4-Trunk/MyOpenSource/Alterra/OpenMI-1.4-SDK>]
- The OpenMI Configuration Editor is a GUI for linking and running OpenMI compliant models and can be obtained from the OpenMI Association [<http://www.openmi.org>]
- The Pipistrelle application developed by HR Wallingford is a GUI for linking and running OpenMI compliant models and can be obtained from the FluidEarth Portal [<http://fluidearth.net/default.aspx>]

### **B.1 ACRU Model**

See folder *Appendix B\Software\ACRU* on the CD accompanying this report.

### **B.2 OpenMI 1.4 Java Wrapper For ACRU**

See folder *Appendix B\Software\ACRU\_OpenMI\_Java* on the CD accompanying this report.

### **B.3 OpenMI 1.4 .Net Wrapper For ACRU**

See folder *Appendix B\Software\ACRU\_OpenMI\_DotNet* on the CD accompanying this report.

**B.4 OpenMI 1.4 .Net Wrapper For MIKE BASIN**

See folder *Appendix B\Software\MIKEBASIN\_OpenMI* on the CD accompanying this report.

**B.5 Model Configurations And Data For The Streamflow Use Case**

See folder *Appendix B\UseCases\Streamflow* on the CD accompanying this report.

**B.6 Model Configurations And Data For The Flow Routing Use Case**

See folder *Appendix B\UseCases\FlowRouting* on the CD accompanying this report.

**B.7 Model Configurations And Data For The Irrigation Use Case**

See folder *Appendix B\UseCases\Irrigation* on the CD accompanying this report.