OBJECT-ORIENTATION AND INTEGRATION FOR MODELLING WATER RESOURCE SYSTEMS USING THE *ACRU* MODEL

DAVID JOHN CLARK

Submitted in fulfilment of the requirements for the degree of
Doctor of Philosophy in Engineering

Bioresources Engineering
School of Engineering
College of Agriculture, Engineering and Science
University of KwaZulu-Natal

November 2018

Supervisor: Prof. J.C. Smithers
Co-supervisor: Prof. G.P.W. Jewitt

EXAMINER'S COPY

ABSTRACT

Water is a limiting resource in South Africa, with demand in many catchments exceeding supply, necessitating transfers of water between catchments. This situation requires detailed and integrated management of the country's water resources, considering environmental, social and economic aspects as outlined in the National Water Act (Act 36 of 1998). Integrated water resources management (IWRM) will require better data and information through monitoring and integrated water resources modelling.

The *ACRU* hydrological model is an important repository of South African water research and knowledge. In recent years there have been technological advances in computer programming techniques and model integration. The thesis for this study was that the valuable knowledge already existing in the *ACRU* model could be leveraged to provide a better hydrological model to support IWRM in South Africa by: (i) restructuring the model using object-oriented design and programming techniques, and (ii) implementing a model interface standard.

Object-oriented restructuring of the *ACRU* model resulted in a more flexible model enabling better representation of complex water resource systems. The restructuring also resulted in a more extensible model to facilitate the inclusion of new modules and improved data handling. A new model input structure was developed using Extensible Markup Language (XML) to complement the object-oriented structure of the *ACRU* model.

It was recognised that different models have different purposes and strengths. The OpenMI 2.0 model interface standard was implemented for *ACRU*, enabling integration with other OpenMI 2.0 compliant specialised models representing different domains to provide a more holistic IWRM view of water resource systems. Model integration using OpenMI was demonstrated by linking *ACRU* to the eWater Source river network model.

A case study in the upper uMngeni Catchment in South Africa demonstrated: (i) the benefits of the object-oriented design of the restructured *ACRU* model, in the context of using *ACRU* to create modelled catchment-scale water resource accounts, and (ii) the integration of *ACRU* with another model using OpenMI. The case study also demonstrated that despite the improvements to the *ACRU* model, the simulations are only as good as the model input data.

DECLARATION 1 – PLAGIARISM

I, *David John Clark*, declare that

(i) The research reported in this thesis, except where otherwise indicated, is my original work;

(ii) This thesis has not been submitted for any degree or examination at any other university;

(iii) This thesis does not contain other persons' data, pictures, graphs or other information, unless specifically acknowledged as being sourced from other persons; and

(iv) This thesis does not contain other persons' writing, unless specifically acknowledged as being sourced from other researchers. Where other written sources have been quoted, then:

a) their words have been re-written, but the general information attributed to them has been referenced;

b) where their exact words have been used, their writing has been placed inside quotation marks, and referenced.

(v) This thesis does not contain text, graphics or tables copied and pasted from the internet, unless specifically acknowledged, and the source being detailed in the thesis and in the References section.

Signed:……..
D.J. Clark

Signed:……..
Professor J.C. Smithers
Supervisor

Signed:……..
Professor G.P.W. Jewitt
Co-supervisor

DECLARATION 2 – PUBLICATIONS

DETAILS OF CONTRIBUTION TO PUBLICATIONS that form part of and/or include research presented in this thesis (including publications submitted and published, giving details of the contributions of each author to the research and writing of each publication):

A large part of the research presented in this thesis was completed as part of Water Research Commission (WRC) funded research projects on which I was the principal researcher and has been reported on in final reports reviewed and published by the WRC. These reports each have several authors representing the research team that contributed to the research. Some of the research has also been reported in published journal papers. In this declaration I have tried to make it clear for each publication where portions from these reports and papers has been reproduced in part or in full in this document and the extent to which the research was my own work. The publications are listed below in the chronological order in which the research and writing was done, and not by publication date.

Publication 1 - Clark *et al.* (2001):

Clark, DJ, Kiker, GA and Schulze, RE. 2001. Object-oriented restructuring of the ACRU agrohydrological modelling system. *Proceedings of the 10th South African National Hydrology Symposium*, Pietermaritzburg, RSA, 26-28.

In this paper the rationale for restructuring *ACRU* and the design of the new object-oriented structure for *ACRU* 2000 are described. The paper was written by me with assistance from Dr GA Kiker and Prof. RE Schulze. The *ACRU* code restructuring work was done by me under supervision by Dr GA Kiker. Clark *et al.* (2001) is referenced in this document.

Publication 2 - Kiker and Clark (2001a):

Kiker, GA and Clark, DJ. 2001a. Development and testing of a natural vegetation, herbivore, and fire model for southern African rangeland management. ASAE Paper No. 017025. ASAE, St. Joseph, Michigan, USA.

In this paper the development of the *ACRU-Veld* module as part of the restructured *ACRU* 2000 model is described. The paper was primarliy written by Dr GA Kiker. Dr GA Kiker provided the conceptual basis and algorithms for the module, and I assisted in the coding of the module. Kiker and Clark (2001a) is referenced in this document.

Publication 3 - Kiker and Clark (2001b):

Kiker, GA and Clark, DJ. 2001b. The development of a Java-based, object-oriented modeling system for simulation of southern African hydrology. ASAE Paper No. 012030. ASAE, St. Joseph, Michigan, USA.

In this paper the rationale for restructuring *ACRU* and the design of the new object-oriented structure for *ACRU* 2000 are described. The paper was primarily written by Dr GA Kiker. The *ACRU* code restructuring work was done by me under supervision by Dr GA Kiker. Kiker and Clark (2001b) is referenced in this document.

Publication 4 - Kiker and Clark (2001c):

Kiker, GA and Clark, DJ. 2001c. Testing and validation of a Java-based, object-oriented modeling system in the Mgeni River watershed, KwaZulu-Natal, South Africa. ASAE Paper No. 012031. ASAE, St. Joseph, Michigan, USA.

In this paper the restructured *ACRU* 2000 version of the *ACRU* model was tested and verified in the uMngeni Catchment. The testing, validation and writing of the paper was primarily done by Dr GA Kiker. Kiker and Clark (2001c) is referenced in this document.

Publication 5 - Campbell *et al.* (2001):

Campbell, KL, Kiker, GA and Clark, DJ. 2001. Development and testing of a nitrogen and phosphorus process model for Southern African water quality issues. ASAE Paper No. 012085. ASAE, St. Joseph, Michigan, USA.

In this paper the development of the *ACRU-NP* module for *ACRU* is described. Prof. KL Campbell did most of the coding of the *ACRU-NP* module. I provided advice to Prof. KL Campbell regarding the coding of the *ACRU-NP* module and together with Prof. KL Campbell and Dr GA Kiker played a key role in developing the design for representing nutrient fluxes in *ACRU* as either a nutrient transport, nutrient transformation or a combination of these. My contribution to the writing of the paper was small. Campbell *et al.* (2001) is referenced in this document.

Publication 6 - Kiker *et al.* (2006):

Kiker, GA, Clark, DJ, Martinez, CJ and Schulze, RE. 2006. A Java-based, object-oriented modeling system for Southern African hydrology. *Transactions of the ASABE* 49 (5): 1419-1433.

In this paper the scientific basis of the *ACRU* model, the object-oriented design of the *ACRU* 2000 version of the *ACRU* model, and the testing and verification of the restructured model in the uMngeni Catchment are described. The paper was primarily written by Dr GA Kiker and Dr CJ Martinez. The *ACRU* code restructuring work was done by me under supervision by Dr GA Kiker. Kiker *et al.* (2006) is referenced in this document.

Publication 7 - Clark *et al.* (2009):

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.

This publication is a reviewed final report to the WRC for WRC Project K5/1490. Prof. JC Smithers was the project leader and I was the principal researcher. The research was done jointly by a project team from various organisations and the report was jointly authored by the authors referenced above. Chapter 2 included the review of modelling framworks, selection of the SPATSIM framework and its further development as the SPATSIM-HDSF framework. The research reported in Chapter 2 was done jointly by me, Prof. DA Hughes, Mr DA Forsysth, Mr KB Meier and Mr G Page-Wood. Chapter 3 described the development of the *ACRU* 4 version of the *ACRU* model engine and associated utilities. The development of the code of the *ACRU* 4 model engine structure were primarily my work. The initial development of the Extensible Markup Language (XML) model input data files was primarily my work. The initial development of the XML model configuration files was primarily my work with some contributions by Mr SLC Thornton-Dibb. The initial concept and development of the rule-set file and code to validate model input values was done by Mr KB Meier and Mr G Page-Wood, but further developed by Mr SLC Thornton-Dibb and myself. The *ACRU* Configuration Editor tool for editing model input files was developed by Mr SLC Thornton-Dibb under my supervision. The TSAnalysis time series analysis tool was developed by Mr A Lutchminarain under my supervision, based on the SPATSIM TSOFT tool and the *ACRUView* tool. Clark *et al.* (2009) is referenced in this document.

Publication 8 - Clark *et al.* (2012a):

Clark, DJ, Hughes, DA, Smithers, JC, Thornton-Dibb, SLC, Lutchminarain, A and Forsyth, DA. 2012a. *Deployment, maintenance and further development of SPATSIM-HDSF: Volume 1 - SPATSIM-HDSF modelling framework*. WRC Report No. 1870/1/12. Water Research Commission, Pretoria, South Africa.

This publication is Volume 1 of a reviewed final report to the WRC for WRC Project K5/1870 funded by the former Department of Water Affair and Forestry (DWAF). Prof. JC Smithers was the project leader. The research was done jointly by a project team from various organisations and the report was jointly authored by the authors referenced above. Chapter 2 describes the further development of the SPATSIM-HDSF framework. The research reported in Chapter 2 was done jointly by myself, Prof. DA Hughes and Mr DA Forsysth. Chapter 4 described the further development of the *ACRU* 4 model engine and associated utilities. The changes made to the code of the *ACRU* 4 model engine were primarily my work. The minor further development of the XML model input files and XML model configuration files was primarily my work. The development of a model input file converter tool was primarily my work. The further development of the *ACRU* Configuration Editor tool was done by Mr Thornton-Dibb under my supervision. The further development of the TSAnalysis time series analysis tool was done by Mr A Lutchminarain under my supervision. Clark *et al.* (2012a) is referenced in this document.

Publication 9 -Clark *et al.* (2012b):

Clark, DJ, Smithers, JC, Thornton-Dibb, SLC and Lutchminarain, A. 2012b. *Deployment, maintenance and further development of SPATSIM-HDSF: Volume 3 - ACRU agrohydrological model*. WRC Report No. 1870/3/12. Water Research Commission, Pretoria, South Africa.

This publication is Volume 3 of a reviewed final report to the WRC for WRC Project K5/1870 funded by the former Department of Water Affair and Forestry (DWAF). Prof. JC Smithers was the project leader. The research was done jointly by a project team from various organisations and the report was jointly authored by the authors referenced above. Volume 3 consisted of project reports and user manuals authored jointly by the project team for the SPATSIM-HDSF software and *ACRU* software utilities. Clark *et al.* (2012b) is referenced in this document.

Publication 10 - Clark and Smithers (2013):

Clark, DJ and Smithers, JC. 2013. *Model integration for operational water resources planning and management*. WRC Report No. 1951/1/12. Water Research Commission, Pretoria, South Africa.

This publication is a reviewed final report to the WRC for WRC Project K5/1951 and was edited by myself and Prof. JC Smithers, my PhD supervisor. Prof. JC Smithers was the

project leader and I was the principal researcher. Individual chapters were authored by different researchers on the project.

Chapter 2 - Thornton-Dibb *et al.* (2013):

Thornton-Dibb, SLC, Smithers, JC and Clark, DJ. 2013. Review and evaluation of river network models. In: eds. Clark, DJ and Smithers, JC, *Model integration for operational water resources planning and management*. WRC Report No. 1951/1/12, Chapter 2. Water Research Commission, Pretoria, South Africa.

Chapter 2 reports on the review and evaluation of third-party river network models to identify a suitable model to be linked to *ACRU*. The research and writing for this chapter was primarily done by Mr SLC Thornton-Dibb, with supervision, advice and editing by Prof. JC Smithers and myself. Thornton-Dibb *et al.* (2013) is referenced in this document.

Chapter 3 - Clark *et al.* (2013):

Clark, DJ, Lutchminarain, A and Smithers, JC. 2013. Review and evaluation of model linkage mechanisms. In: eds. Clark, DJ and Smithers, JC, *Model integration for operational water resources planning and management*. WRC Report No. 1951/1/12, Chapter 3. Water Research Commission, Pretoria, South Africa.

Chapter 3 reports on the review and evaluation of approaches to model integration and third-party model linkage systems to identify a suitable model linkage system to enable integration of *ACRU* with a river network model. The preliminary literature search was conducted by Mr A Lutchminarain, but the literature review in Section 3.1 was primarily written by me with contributions from Mr A Lutchminarain and editing by Prof. JC Smithers. The evaluation of shortlisted model linkage systems and the reporting of this evaluation (Section 3.2) was done jointly by Mr A Lutchminarain and myself. Clark *et al.* (2013) is referenced in this document and parts of the literature review have been summarised or reproduced in Section 4.1, Section 4.2 and Section 4.3 of this document.

Chapter 4 - Clark (2013):

Clark, DJ. 2013. *ACRU* model development. In: eds. Clark, DJ and Smithers, JC, *Model integration for operational water resources planning and management*. WRC Report No. 1951/1/12, Chapter 4. Water Research Commission, Pretoria, South Africa.

Chapter 4 includes some background to the development history of *ACRU* (Section 4.1), changes to the XML model input file structure (Section 4.2) and changes to the model

structure (Section 4.3). This chapter was written by myself with some minor editing by Prof. JC Smithers. The design and coding of the changes to the *ACRU* model and model input files was done by me based on requirements identified by myself and colleagues Mr MJC Horan and Mr SLC Thornton-Dibb. Clark (2013) is referenced in this document and parts have been summarised or reproduced in Chapter 2, Section 3.2, Section 3.3 and Appendix 8.1 of this document.

Chapter 5 - Clark and Lutchminarain (2013):

Clark, DJ and Lutchminarain, A. 2013. Implementation of OpenMI for model linking. In: eds. Clark, DJ and Smithers, JC, *Model integration for operational water resources planning and management*. WRC Report No. 1951/1/12, Chapter 5. Water Research Commission, Pretoria, South Africa.

Chapter 5 describes the development of OpenMI linkable components for the *ACRU* and MIKE BASIN models. This chapter was written jointly by myself and Mr A Lutchminarain with some editing by Prof. JC Smithers. The development of the OpenMI 1.4 linkable component for *ACRU* was done by myself. The development of the OpenMI 1.4 and OpenMI 2.0 linkable components for MIKE BASIN was done by Mr A Lutchminarain with some assistance from myself. Clark and Lutchminarain (2013) is referenced in this document.

Publication 11 - Clark (2015a):

Clark, DJ. 2015a. *Development and assessment of an integrated water resources accounting methodology for South Africa*. WRC Report 2205/1/15. Water Research Commission (WRC), Pretoria, South Africa.

This publication is a reviewed final report to the WRC for WRC Project K5/2205 and was edited by myself. I was the project leader and principal researcher. Individual chapters were authored by different researchers on the project.

Chapter 2 - Clark *et al.* (2015):

Clark, DJ, Bastiaanssen, WGM, Smithers, JC and Jewitt, GPW. 2015. A review of water accounting frameworks for potential application in South Africa. In: ed. Clark, DJ, *Development and assessment of an integrated water resources accounting methodology for South Africa*. WRC Report No. 2205/1/15, Chapter 2. Water Research Commission (WRC), Pretoria, South Africa.

Chapter 2 is a review of water accounting frameworks. The research and writing for this chapter was primarily done by me, with some expert contribution from Prof. Bastiaanssen and editing by Prof. JC Smithers and Prof GPW Jewitt. Clark *et al.* (2015) is referenced in this document.

Chapter 4 - Clark (2015b):

Clark, DJ. 2015b. Development of a methodology for water use quantification and accounting. In: ed. Clark, DJ, *Development and assessment of an integrated water resources accounting methodology for South Africa*. WRC Report No. 2205/1/15, Chapter 4. Water Research Commission (WRC), Pretoria, South Africa.

Chapter 4 describes the modifications made to the WA+ Resource Base Sheet, and the development of a methodology to quantify water use and compile catchment-scale water resource accounts using *ACRU*. The research and writing for this chapter was done by myself. Clark (2015b) is referenced in this document and parts have been summarised or reproduced in Section 3.4 and Appendix 8.8.

Chapter 5 - Clark (2015d):

Clark, DJ. 2015d. uMngeni Catchment case study. In: ed. Clark, DJ, *Development and assessment of an integrated water resources accounting methodology for South Africa*. WRC Report No. 2205/1/15, Chapter 5. Water Research Commission (WRC), Pretoria, South Africa.

Chapter 5 describes the case study in the uMngeni Catchment. The research and writing for this chapter was done by myself. Clark (2015d) is referenced in this document.

Chapter 6 - Clark (2015c):

Clark, DJ. 2015c. Sabie-Sand Catchment case study. In: ed. Clark, DJ, *Development and assessment of an integrated water resources accounting methodology for South Africa*. WRC Report No. 2205/1/15, Chapter 6. Water Research Commission (WRC), Pretoria, South Africa.

Chapter 6 describes the case study in the Sabie-Sand Catchment. The research and writing for this chapter was done by myself. Clark (2015c) is referenced in this document .

Publication 12 - Clark (2016):

Clark, DJ. 2016. *Further development and assessment of an integrated water resources accounting methodology for South Africa: Deliverable 3: Progress report - Year 1*. Unpublished report to the Water Research Commission (WRC) for Deliverable 3 of WRC Project K5/2512. Centre for Water Resources Research (CWRR), University of KwaZulu-Natal, Pietermaritzburg, South Africa.

This document in an unpublished progress report to the WRC for WRC Project K5/2512 and was edited by me with contributions from various colleagues. I was the project leader and principal researcher. The research and writing in Chapter 4 of Clark (2016) was primarily done by myself and is referenced in this document.

Publication 13 - Clark (2017a):

Clark, DJ. 2017a. *Further development and assessment of an integrated water resources accounting methodology for South Africa: Deliverable 4: Annual report - Year 1*. Unpublished report to the Water Research Commission (WRC) for Deliverable 4 of WRC Project K5/2512. Centre for Water Resources Research (CWRR), University of KwaZulu-Natal, Pietermaritzburg, South Africa.

This document is an unpublished progress report to the WRC for WRC Project K5/2512 and was edited by me with contributions from various colleagues. I was the project leader and principal researcher. The research and writing in Chapter 3 of Clark (2017a) was primarily done by myself and is referenced in this document.

Publication 14 - Clark (2017b):

Clark, DJ. 2017b. *Further development and assessment of an integrated water resources accounting methodology for South Africa: Deliverable 5: Progress report - Year 2*. Unpublished report to the Water Research Commission (WRC) for Deliverable 5 of WRC Project K5/2512. Centre for Water Resources Research (CWRR), University of KwaZulu-Natal, Pietermaritzburg, South Africa.

This document is an unpublished progress report to the WRC for WRC Project K5/2512 and was edited by myself with contributions from various colleagues. I was the project leader and principal researcher. The research and writing in Chapter 3 of Clark (2017b) was primarily done by myself and is referenced in this document.

Publication 15 - Clark (2018):

Clark, DJ. 2018. Investigation of satellite remotely sensed rainfall for use in hydrological modelling in the upper uMngeni Catchment, South Africa. Paper in preparation. Centre for Water Resources Research, University of KwaZulu-Natal, Pietermarithburg, South Africa.

This document is a paper in preparation describing an investigation completed as part of WRC Project K5/2512 and was written by myself.

PREFACE

The work described in this thesis was carried out in the School of Engineering, University of KwaZulu-Natal (UKZN), Pietermaritzburg, South Africa under the supervision of Professor JC Smithers and Professor GPW Jewitt.

These studies represent original work by the author and have not otherwise been submitted in any form for any degree or diploma to any tertiary institution. Where use has been made of the work of others it is duly acknowledged below or in the text.

A brief background to the development of the *ACRU* model is included in Appendix 8.1. The algorithms representing hydrological processes that form the basis of the *ACRU* model have been developed over many years by many people. These algorithms remain largely unchanged in the restructured *ACRU* model and I can make no claim to this work. With regard to the object-oriented restructuring of the *ACRU* model, the initial conceptual design including the concept of *MModel*, *CComponent*, *PProcess* and *DData* classes, the associations between these classes and the initial skeleton code of the *CNode* class were the work of Dr GA Kiker and Dr O David in 1998. This initial conceptual design was further developed by myself, in consultation with Dr GA Kiker, who supervised the work, and to a lesser extent with Dr O David. This further developed conceptual design was implemented by myself under supervision by Dr GA Kiker between 1998 and 2002 to create the restructured object-oriented *ACRU* 2000 version of the model. Subsequent to 2002, changes to the design of the structure of the *ACRU* 2000, *ACRU* 4 and *ACRU* 5 versions of the model are almost solely my work. Likewise, the design of the Extensible Markup Language (XML) based model input structure for *ACRU* is almost solely my work, though inevitably enhanced through the constructive criticism of colleagues. Since 2002, several researchers and students have made changes to and created new classes representing hydrological processes and associated variables in *ACRU*, but these process representations are not part of the work reported in this document. Similarly software tools have been developed as part of the *ACRU* modelling system, including the Configuration Editor and the TSAnalysis tools, and although I supervised the development of these tools, they are not primarily my work and are thus not reported in detail this document.

By convention of the *ACRU* model developers, the name *ACRU* is italicised. In this document the names of software packages, classes, interfaces, attributes, methods and XML elements are also italicised.

ACKNOWLEDGEMENTS

My sincere appreciation goes to all the people who encouraged, advised and supported me during this study, but especially:

- My family, especially my wife Gaynor, for their love and patience, and for the many sacrifices that they have made during this study.
- My supervisors Professor JC Smithers and Professor GPW Jewitt for their valuable advice and guidance.
- Professor GA Kiker for his supervision during the initial restructuring of the *ACRU* model into object-oriented Java.
- My current and former colleagues for their advice, guidance and encouragement, especially Professor RE Schulze, Professor GA Kiker and Professor PWL Lyne.

My sincere appreciation also goes to the following organisations:

- The WRC for funding research projects (636, K5/1490, K5/1870, K5/1951, K5/2205 and K5/2512) wihin which the majority of the research that contributed to this PhD study was conducted.
- Ezemvelo KZN Wildlife for the 2011 Land Cover Dataset for KwaZulu-Natal.
- Umgeni Water for data on streamflow and the Mooi-uMngeni transfer.
- The Satellite Applications Hydrology Group at UKZN for reference potential evaporation data.
- The Department of Water and Sanitation (DWS) for data including: streamflow, rainfall, abstractions from Midmar Dam, the Mooi-uMngeni transfer, registered dams and rating curves and volume-area relationships for Midmar Dam and Albert Falls Dam.
- The Agricultural Research Council (ARC) for rainfall data.
- The South African Weather Service (SAWS) for rainfall data.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AAHMS	*ACRU* Agrohydrological Modelling System
ACRU	Agricultural Catchment Research Unit (former)
API	Application Programming Interface
ARC 2.0	African Rainfall Climatology (Version 2)
BEEH	School of Bioresources Engineering and Environmental Hydrology (former)
CCA	Common Component Architecture
CMORPH	Climate Prediction Center Morphing Technique
COM	Component Object Model
CRC	Cooperative Research Centre
CSIR	Council for Scientific and Industrial Research
CSV	Comma Separated Value
CWRR	Centre for Water Resources Research, University of KwaZulu-Natal
Delft-FEWS	Delft Flood Early Warning System
DEM	Digital Elevation Model
DLL	Dynamic Link Library
DOS	Disk Operating System
DSO	Dam Safety Office
DWA	Department of Water Affairs (former)
DWAF	Department of Water Affairs and Forestry (former)
DWS	Department of Water and Sanitation
FAO	Food and Agriculture Organisation
FEWS	Famine Early Warning System
GIS	Geographic Information System
GUI	Graphical User Interface
HLA	High Level Architecture
HRU	Hydrological Response Unit
IFR	In-stream Flow Requirement
IWRM	Integrated Water Resource Management
JAMS	Jena Adaptable Modelling System
JNI	Java Native Interface
MAP	Mean Annual Precipitation
MMS	Modular Modelling System
NWA	National Water Act
OATC	OpenMI Association Technical Committee

OGC®	Open Geospatial Consortium
OMG	Object Management Group
OMS	Object Modelling System
OOA	Object-Oriented Analysis
OOD	Object-Oriented Design
OOP	Object-Oriented Programming
OpenMI	Open Modelling Interface
PAW	Plant Available Water
PI	Published Interface
RFE	Rainfall Estimator
RSM	Regional Simulation Model
SAHG	Satellite Applications Hydrology Group
SANQCD	South African National Quaternary Catchment Database
SAWS	South African Weather Service
SCS	Soil Conservation Service
SDK	Software Development Kit
SLIM	Spatial & Land Information Management
SRTM	Shuttle Radar Topography Mission
TIME	The Invisible Modelling Environment
TRMM	Tropical Rainfall Measuring Mission
UH	Unit Hydrograph
UKZN	University of KwaZulu-Natal
UML	Unified Modelling Language
UN	United Nations
USA	United States of America
USDA	United States Department of Agriculture
USGS	United States Geological Survey
WA+	Water Accounting Plus
WARMS	Water Authorisation Registration and Management System
WMA	Water Management Area
WRC	Water Research Commission
WTW	Water Treatment Works
WWF-SA	World Wildlife Fund – South Africa
XML	Extensible Markup Language

1 INTRODUCTION

Water plays a key role in social and economic wellbeing (Colvin *et al.*, 2008). Increasing demand for water as a result of population and economic growth, together with pollution of water resources and climate change, has resulted in increased water scarcity in many catchments. Molden *et al.* (2007) state that globally 1.2 billion people live in catchments where utilisation of water resources is no longer sustainable, resulting in physical water scarcity. In addition, a further 1.6 billion people live under conditions of economic water scarcity, where lack of infrastructure limits access to available water due to limited human, institutional, or financial capital (Molden *et al.*, 2007).

1.1 Water Resources in South Africa

Both physical and economic water scarcity are prevalent in South Africa. DWA (2013) states that South Africa is the 30th driest country in the world and, although it has higher average rainfall than neighbouring countries, Namibia and Botswana, the per capita water availability in South Africa is lower. The following statistics from WWF-SA (2017) provide the context to the water situation in South Africa. South Africa is a water scarce country with an average annual rainfall of 490 mm compared to the global average of 814 mm, with 21% of the country receiving less than 200 mm of rainfall per year. Average potential evaporation is 1800 mm per year. The water resources are not equally distributed spatially, with the main water source areas being just 8% of the country, but contributing 50% of the runoff. Approximately 20% of the 49 billion m^3 of mean annual runoff are available at a high assurance (98%) (DWA, 2013). Estimated groundwater use is approximately 2 billion m^3 per annum out of an estimated high assurance potential annual yield of 7 billion m^3 (DWA, 2013). Water storage infrastructure in South Africa is highly developed to assist in managing the high intra- and inter- annual variability in rainfall and runoff. There are over 5000 registered dams (DSO, 2016) and numerous smaller farm dams. Of the registered dams, 794 are classified as large dams, having a wall height ≥ 15 m, or a wall height > 5 m and storage capacity > 3 million m^3 (DWS, 2015). These large dams have a total storage capacity of 31 billion m^3, which is approximately 63% of mean annual runoff. To augment water resources in high demand areas, there are 29 large-scale inter-catchment transfers with a total capacity of 7.5 billion m^3 per year (DWS, 2015). There are limited additional economically feasible sites for dams and inter-catchment transfer schemes (DWA, 2013; DWS, 2015). WWF-SA (2017) estimates that physical water scarcity will be reached in South Africa by as early as 2025.

The water situation in South Africa, described by the statistics quoted above, indicates that South Africa is a water scarce and water stressed country that has entered an era where options for further water infrastructure development are becoming less physically and economically feasible, which means that water resources will need to be managed better and innovative solutions to reducing demand will need to be sought. An Integrated Water Resource Management (IWRM) approach will be key to providing a systems view of water resources at a range of spatial scales including field, catchment, national and regional scales. IWRM recognises that there are hydrological, engineering, ecological, economic, political, social and institutional aspects to water resources management, which need to be considered to ensure sustainable and equitable use. This paradigm has been captured in Section 3 of the National Water Act (NWA, 1998) of South Africa (Act 36 of 1998) which recognises "*the need for the integrated management of all aspects of water resources*". Pollard and du Toit (2008) state that IWRM is central to being able to achieve the two main principles of the NWA, namely equity and sustainability. However, although the theory of IWRM is widely accepted, implementation is difficult (Molina *et al.*, 2010), and despite the good intentions of the NWA, implementation remains a challenge. IWRM will require multidisciplinary integrated modelling of water resource systems, however integrated water resource modelling tools do not yet exist in South Africa. In this thesis the term "water resource system" is used in the context of IWRM, and includes the interrelated hydrological, engineering, ecological, economic and social aspects of water which form the whole system.

Water accounting is a field of water resource management that has developed rapidly in the past few years, to help address the need to quantify, describe, understand, compare and communicate the status of water resource systems. The need for better management of stressed water resource systems and the recognition of the need for integrated assessment have been the catalyst for the developments in water accounting. South Africa has recently been building capacity in developing water-economic accounts at a national and Water Management Area (WMA) scale (Maila *et al.*, 2018), and also catchment-scale water resource accounts (Clark, 2015a).

1.2 Water Resource System Complexity

The hydrology of catchments is complex even when they are in their natural state and anthropogenic development within these systems adds to the complexity (Kiker *et al.*, 2006). It is widely recognised that water, ecological and social systems are complex in themselves and that the interrelationships and interactions between these systems increases the complexity (Pollard and du Toit, 2008). Pollard and du Toit (2008) explain that complex

systems are distinguished from simple, though potentially complicated, systems by attributes such as non-linearity, uncertainty, scale, self-organisation and feedback loops. The complex nature of water resources systems is largely due to their inherent spatial and temporal variability, and it is important to consider these systems holistically as a hierarchy of interdependent systems and sub-systems (Pollard and du Toit, 2008). Liu and Stewart (2004) state that natural resource management is also complex, requiring communication and consensus amongst various stakeholders who often have conflicting social, economic and political interests. Pollard and du Toit (2008) conclude that the NWA provides an enabling environment for the management of complex water resource systems in South Africa.

1.3 Water Resource System Modelling

Modelling is a critical part of water resource planning and management, as models enable complex hydrological problems to be studied (Elshorbagy and Ormsbee, 2006). There are two broad groups of hydrological models: (i) deterministic models for which the model output is fully determined by the model input parameters and the physical rules or mathematical relationships on which the model is based, and (ii) stochastic models which use statistical methods to relate measured time series variables, such as rainfall and streamflow, in terms of probabilities (Shaw, 1994; Ghashghaei *et al.*, 2013). Deterministic physically based hydrological models are useful for: (i) representing locations where detailed measurements are not available or where conditions may change with time, (ii) building understanding of complex water resource systems, (iii) providing estimates of water resource variables that are difficult or expensive to measure, (iv) evaluating the impact of anticipated climate change, (v) evaluating proposed water resource policy and management strategies, and (vi) providing forecasts of hydrological conditions using forecast meteorological driver variables. This section, and the study as a whole, focusses on deterministic physically based hydrological modelling.

Catchments display a wide range of heterogeneity in their spatial attributes and complexity in response to climate inputs that vary both spatially and with time (Bian, 2007; McDonnell *et al.*, 2007; Kumar *et al.*, 2010). When developing a hydrological model, it is necessary to understand the relevant components of the catchment being modelled, the key variables describing each component, the physical processes that take place and the interactions between the components (Elshorbagy and Ormsbee, 2006; Javadi *et al.*, 2009). A key challenge in hydrological modelling is to adequately represent the complex and complicated characteristics of catchments, which also depends on the objectives of the model or study.

The result is that hydrological models are a simplified representation of real catchments, limited by: (i) our understanding of catchment processes, (ii) our ability to represent the complex and complicated characteristics of catchments in computer code, and (iii) the availability of data describing both discrete and continuous catchment characteristics and phenomena (Wang *et al.*, 2000; Silvert, 2001; Wang *et al.*, 2005a). In the context of water resources management it also important to be able to represent the engineered systems consisting of water storage and transfer infrastructure in hydrological models. A key challenge for IWRM is its multi-disciplinary nature, requiring: (i) the integration and analysis of data from several sources, and (ii) the integration of models and analysis tools developed by experts in different disciplines (Kokkonen *et al.*, 2003).

From literature a list of desired characteristics for a hydrological model, for use as a tool in IWRM, were identified for use as a broad guideline to model development. These characteristics, focussing on the model engine and excluding the model user interface, are briefly listed and described in Table 1-1.

Table 1-1 Hydrological model characteristics identified from literature

Characteristic	**Description**
Representation of complexity	• Represent the spatial variability and temporal dynamics within a catchment (Elshorbagy and Ormsbee, 2006; Bian, 2007). • Represent complexity in a realistic manner including both linear and non-linear processes (Elshorbagy and Ormsbee, 2006). • Represent interactions between processes (feedbacks) (Wang *et al.*, 2005a; Elshorbagy and Ormsbee, 2006).
Appropriate simplification	• Represent complexity in as simple a manner as possible to simplify model application (Wang *et al.*, 2005a; Elshorbagy and Ormsbee, 2006).
Flexibility	• Design model to be transferrable to new situations and locations (Argent *et al.*, 2000; Argent and Houghton, 2001; Wang *et al.*, 2005a). • Support simple and advanced options for data structures and process representation to adapt to different levels of data availability (Garrote and Becchi, 1997). • Be suitable for application in both water resource planning and operations contexts, including forecasting (Javadi *et al.*, 2009; Carr and Podger, 2012; Dutta *et al.*, 2013; Yang *et al.*, 2017)
Extensibility	• Design model such that scientific representation of physical processes can be modified, extended or replaced separately without needing to be concerned with the non-science aspects of model construction (Alfredsen and Saether, 2000; Jones *et al.*, 2001; Javadi *et al.*, 2009).
Data handling	• Distributed hydrological models are data intensive, thus efficient data storage, access and management are important (Garrote and Becchi, 1997; Alfredsen and Saether, 2000). • Make provision for the use of new data tools and sources, including GIS and remote sensing (Kang and Merwade, 2011; Ghashghaei *et al.*, 2013; Kang *et al.*, 2016)

Table 1-1 (cont.) Hydrological model characteristics identified from literature

Characteristic	Description
Modern software engineering	• Use modern software engineering practices (Elshorbagy and Ormsbee, 2006).
Integration with other models	• Provide a means for integrating with models representing other domains to enable integrated water resource modelling to support IWRM (Argent *et al.*, 2000; Argent and Houghton, 2001; Kokkonen *et al.*, 2003; Elshorbagy and Ormsbee, 2006; Kiker *et al.*, 2006; Carr and Podger, 2012).

Developers of environmental models are increasingly realising: (i) the benefits of adopting modern approaches to software engineering, which have contributed greatly to the assessment and management of water resources, and (ii) that object-orientation is at the forefront of these approaches (Spanou and Chen, 2000; Argent *et al.*, 2002; Javadi *et al.*, 2009). Barthel *et al.* (2006) state that assessment of complex hydrological systems often requires detailed process-oriented models and that the object-oriented approach to model design and implementation is ideally suited for use in this type of model and meets the need to better model real-world complexity and the integration of models. The object-oriented approach to modelling helps in managing complexity by representing such systems in a more intuitive manner (Band *et al.*, 2000; Clark *et al.*, 2001; Kiker and Clark, 2001b ; Wang *et al.*, 2005b ; Kiker *et al.*, 2006). The object-oriented modelling approach will also enable these models to be easily extended as understanding of the modelled system develops (Elshorbagy and Ormsbee, 2006).

Typically legacy models have been designed for specific domains within the water resource system, such as rainfall-runoff, groundwater, urban stormflow, water quality, river network and ecosystem models (Carr and Podger, 2012). However, water managers applying IWRM require a new generation of models and modelling tools that enable integrated modelling across domains by multi-disciplinary modelling teams to provide a systems perspective for water management in both short-term operational decisions and long-term planning. (Argent *et al.*, 2000; Jones *et al.*, 2001; Kiker *et al.*, 2006; Carr and Podger, 2012). This integrated modelling will enable trade-offs for various policy and management scenarios to be evaluated. These models and modelling tools need to be able to handle and represent the inherent complexity of water resources systems and the need for integrated holistic assessment of water resource systems to enable management decisions that lead to better water allocation and improved water use efficiency. These models could be used to estimate components of water accounts for which measured data is not available, to facilitate better understanding of system water resource availability and use.

1.4 Thesis and Objectives

The motivation for this study was the requirement for, and potential development of, a suitable hydrological model for use as a tool to assist in IWRM in South Africa, through adequately representing water resource system complexity and enabling integration with other domain models for use in multi-disciplinary assessments. The *ACRU* agrohydrological model, described in Schulze *et al.* (1995), has been developed and applied in South Africa for over 40 years making it an important repository of local water research and knowledge. The physical conceptual nature of the model and its versatility, has also enabled the model to be applied internationally. However, to realise its full potential and meet the requirements listed in Table 1-1 the *ACRU* model structure needed to be revised.

The thesis for this study is that the valuable knowledge already existing in the *ACRU* model can be leveraged to provide a better hydrological model to support IWRM for both water resource planning and operations in South Africa by: (i) restructuring the model using object-oriented design and programming techniques to create a new more flexible and extensible model structure that will facilitate more realistic representation of real-world complexity of water resource systems, and (ii) implementing a model interface standard that will enable *ACRU* to be linked to other specialised models for different domains to provide new and improved information through a more holistic IWRM view of water resource systems.

The specific objectives for this research study were thus to:

(i) Restructure *ACRU* to make it more flexible, and thus to enable: (a) more realistic representation of the physical components of complex water resource systems, (b) better representation of engineered flows between catchments, and (c) options for representation of hydrological processes in varying degrees of detail depending on availability of data.

(ii) Restructure *ACRU* to make it more extensible, to facilitate easier inclusion of new functionality including: (a) improved representations of hydrological processes, and (b) new analysis tools, such as a module for compiling water resource accounts.

(iii) Restructure *ACRU* to make it easier to include: (a) new data sources and formats, and (b) better handing of time series data.

(iv) Develop a means of linking *ACRU* with other models to facilitate integrated modelling studies.

1.5 Overview of the Document

The context for the study has been provided in this chapter. In Chapter 2 an overview of the *ACRU* model is provided, including a brief background to the model and a description of the structure of the *ACRU* 3 version of the model that preceded the work described in this study. Chapter 3 starts with a review of object-oriented programming in the context of water resources modelling, and then includes a description of the design and development of an object-oriented structure for the *ACRU* model and a complementary Extensible Markup Language (XML) based model input structure. Chapter 4 includes: (i) a review of the potential methods available for integrating models, (ii) a description of the design and development of an Open Modelling Interface (OpenMI) wrapper for the *ACRU* model which will enable it to be linked with other models and modelling tools, and (iii) a description of the design and development of an OpenMI wrapper for the eWater Source river network model. In Chapter 5 a case study in the upper uMngeni Catchment in KwaZulu-Natal, South Africa is presented, demonstrating: (i) the application and some of the advantages of the restructured object-oriented *ACRU* model, and (ii) the linking of the *ACRU* and the eWater Source models using OpenMI. The outcome of the research is concluded in Chapter 6. The appendices in Chapter 8 include details of the *ACRU* development and case study discussed in Chapters 3 - 5. An overview of the document chapters is shown in Figure 1-1.

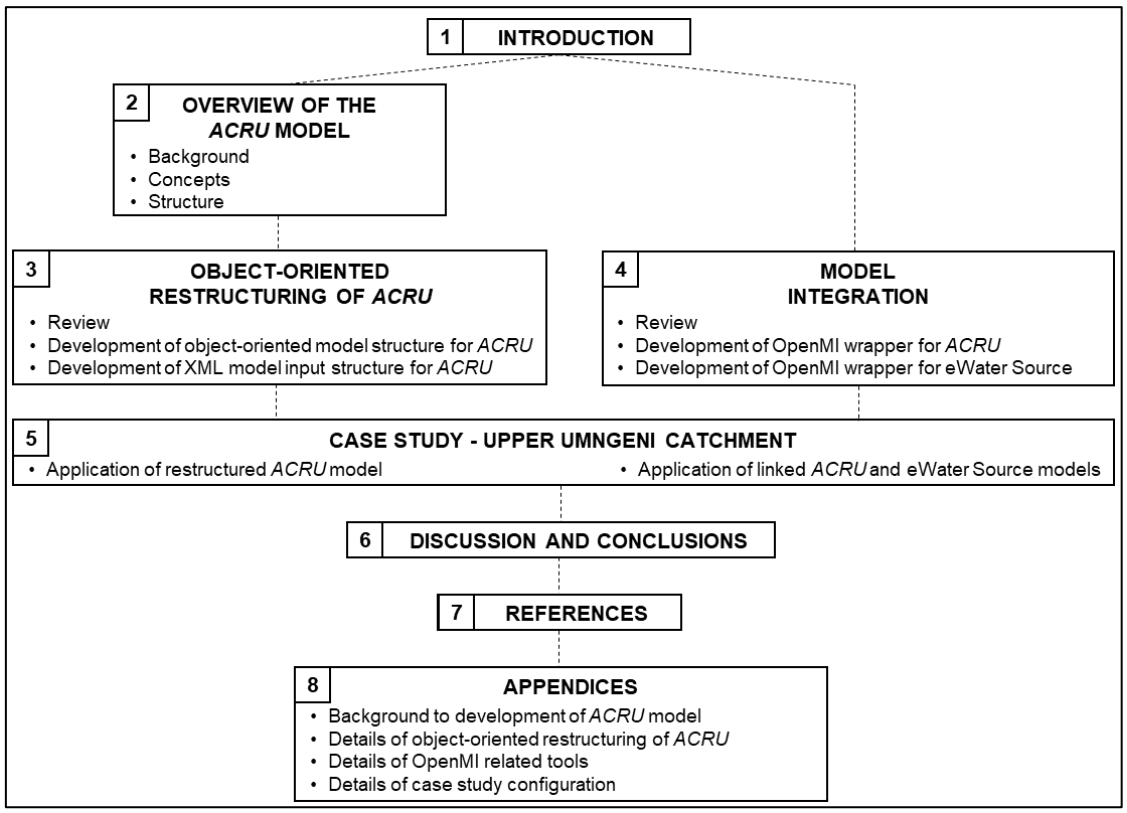

Figure 1-1 Overview of the document chapters

2 OVERVIEW OF THE *ACRU* MODEL

The *ACRU* model is described in Schulze *et al.* (1995) as a physical conceptual agrohydrological model operating at a daily timestep. The *ACRU* model is further described in Schulze *et al.* (1995) 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.*, 1995). A good overview of the development, structure, concepts and application of the *ACRU* modelling system up to 2002 is provided by Schulze and Smithers (2004). A summarised timeline of the development of the *ACRU* model is shown in Table 2-1. A brief background to the development of the *ACRU* model and a more detailed timeline summarising the history of the development of the *ACRU* model can be found in Appendix 8.1. Since its inception in the 1970s the *ACRU* model has continued to be widely used as an invaluable agrohydrological modelling tool. A comprehensive list of applications of the *ACRU* model, both in South Africa and internationally is provided in Schulze and Smithers (2004) and some more recent published applications include Kienzle and Schmidt (2008), Schmidt *et al.* (2009), Forbes *et al.* (2011), Kienzle (2011), Warburton (2011), Kienzle *et al.* (2012), Smithers *et al.* (2013), Schütte (2014), Schulze and Schütte (2015), Aduah *et al.* (2017), Kusangaya *et al.* (2017), Schütte and Schulze (2017), and Smithers *et al.* (2017).

Table 2-1 Summarised timeline of the development of the *ACRU* model

Time Period	**Version**	**Description**
Early 1970s -1984	*ACRU* 1	• Original development the model in FORTRAN
Late 1980s - 1990	*ACRU* 2	• Further development of the model
Early 1990s - 2002	*ACRU* 3	• Further development of the model
1998 - 2002	*ACRU* 2000	• Initial design and development of object-oriented version of the model in Java
2008 - 2011	*ACRU* 4	• Further development of object-oriented version of the model • Design and development of XML model input file structure
2012 - 2017	*ACRU* 5	• Further development of object-oriented version of the model • Development of OpenMI wrappers for *ACRU*

In this document the term '*ACRU model*' refers to the model engine and '*ACRU modelling system*' refers to the model engine together with the associated utility software developed to assist users in configuring the model and analysing model output. The work reported in this document focusses on the *ACRU* model, building on the initial design and development of the object-oriented *ACRU* 2000 version of the model by: (i) refining the design, (ii)

developing a model input file structure using XML, and (iii) developing OpenMI wrappers to enable linking with other models. The *ACRU* 3 version of the model is the reference point from which the further development of the *ACRU* model and modelling system, reported in this document, should be considered. The concepts and structure of the *ACRU* 3 version will be briefly described in Section 2.1 to provide the reader with a basic understanding of the model, and also to provide the context for the discussion in Section 2.2 and subsequent further development of the model.

2.1 Concepts and Structure of the *ACRU* 3 Version

The *ACRU* model is physical in the sense that it explicitly represents physical hydrological processes, and conceptual in that the significant processes characterising the hydrological system and also the feedbacks between them are idealised (Eagleson, 1983; cited by Schulze and Smithers, 2004). Schulze and Smithers (2004) state that *ACRU* is not intended to be a parameter fitting or optimising model, and that variables representing the physical characteristics of a catchment are used, rather than optimised parameter values.

ACRU can be used as: (i) a point model, (ii) a lumped model consisting of a single catchment with lumped (average or dominant) climate, vegetation, land use and soil characteristics, or (iii) a distributed cell-type model in large catchments, or catchments with complex lands uses and soils (Schulze and Smithers, 2004). In distributed mode, subcatchments are assumed to be homogeneous with regard to climate, vegetation, land use and soil characteristics, and should ideally not exceed 50 km^2. *ACRU* is a multi-level model in the sense that, for many of the hydrological processes represented, there are multiple options or pathways available to estimate modelled variables depending on the level of input data available and the relative accuracy and detail required for the simulated outputs (Schulze and Smithers, 2004). In distributed mode, individual subcatchments can be modelled using different levels of information. This document is focussed on the use of *ACRU* in distributed mode, although the catchment configuration, representation of hydrological processes and data requirements are similar for all modes.

2.1.1 Representation of Physical Components

The *ACRU* 3 version of the model (Schulze, 1995a; Smithers and Schulze, 1995) conceptually represents hydrological systems using a set of one or more subcatchments with cascading streamflow links between them. Each subcatchment consists of a number of layers including: a climate layer, a land cover/use layer, a topsoil layer (referred to as the A-

horizon), a subsoil layer (referred to as the B-horizon) and a baseflow store, as shown in Figure 2-1.

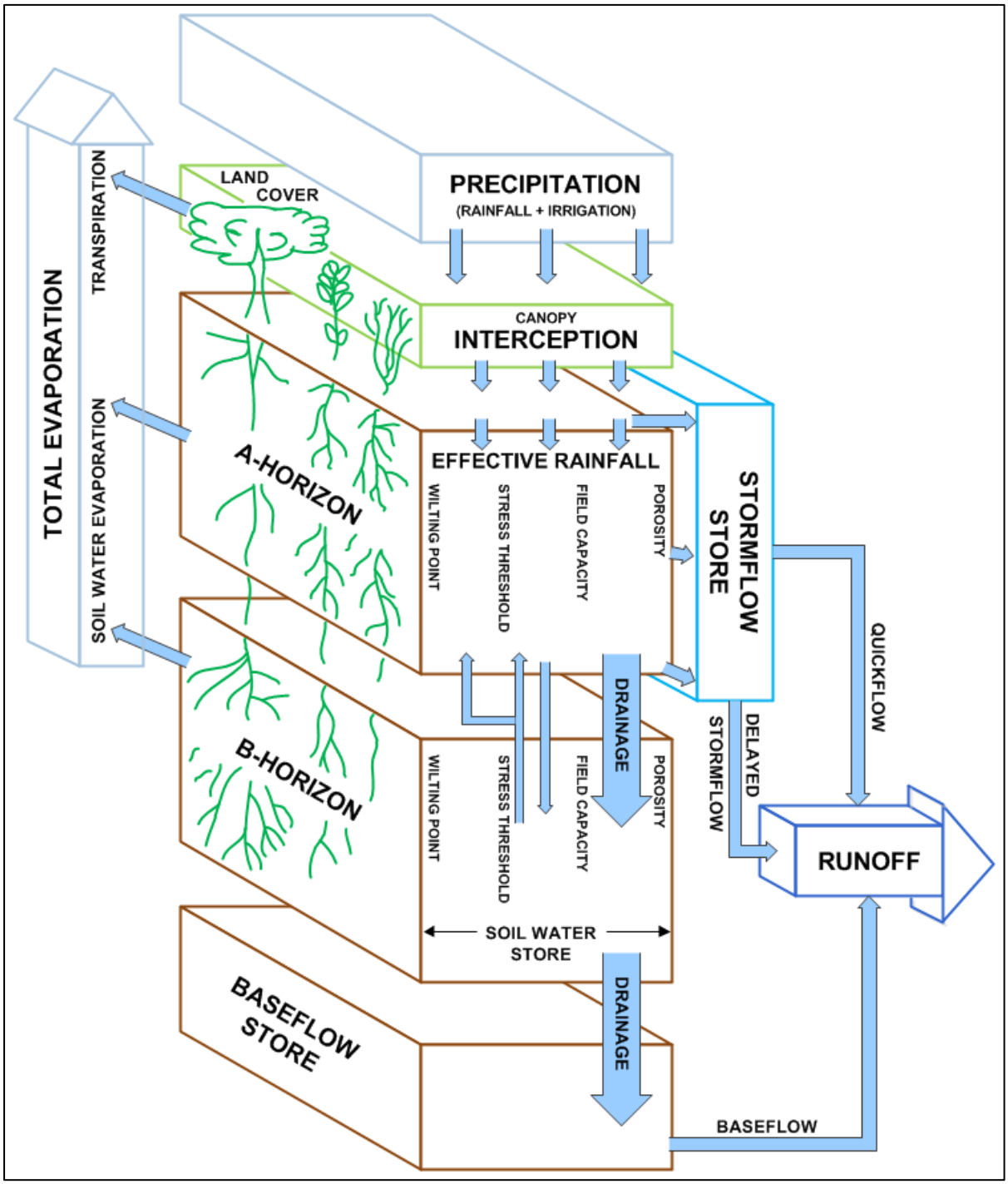

Figure 2-1 *ACRU* 3 vertical layer structure, fluxes and processes (after Schulze, 1984; Schulze, 1989)

In its simplest form each subcatchment is assumed to have homogenous climate, vegetation and soils characteristics. More complex subcatchment configurations are conceptualised

through a set of options that enable each subcatchment to have the following spatial subcomponents as shown in Figure 2-2:

- An impervious area that is adjunct to the streamflow network to which it directly contributes runoff.
- An impervious area that is disjunct from the streamflow network and contributes runoff to the surface of the surrounding subcatchment landscape.
- A dam, situated either: (i) at the downstream exit of the subcatchment where it receives streamflow from the entire subcatchment and streamflow from any upstream subcatchments, or (ii) internally, such as for smaller farm dams which receive streamflow from only a portion of the subcatchment, or (iii) an off-channel storage dam.
- An area of irrigated crops that receives water from run-of-river or a dam within the subcatchment and from which return flows may be generated.

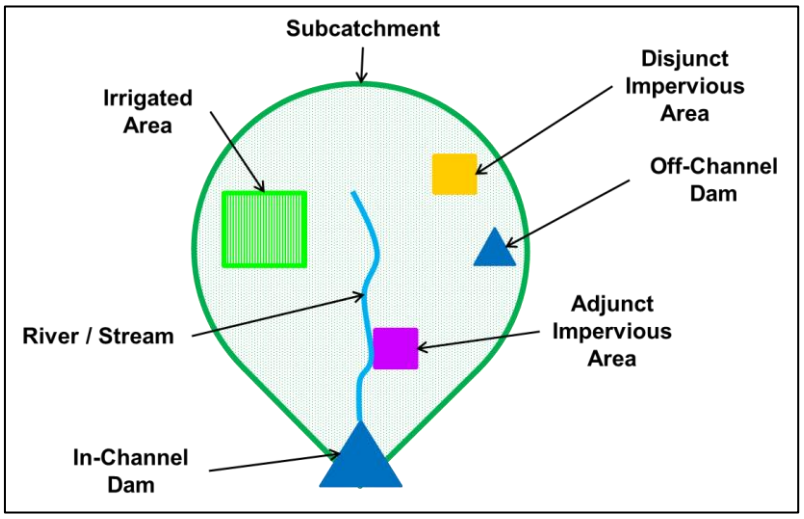

Figure 2-2 Conceptual spatial subcomponents of a subcatchment in *ACRU* 3

Options also exist to enable modelling of specialised wetland and riparian zone subcatchments (Schulze, 1995a; Smithers and Schulze, 1995). Wetland subcatchments behave similarly to normal subcatchments except that when the capacity of the river channel is exceeded the river channel overflows onto the land portion of the subcatchment. Wetland subcatchments would typically be modelled with a shallow dam at the exit of the subcatchment to represent the open water portion of a wetland. Similarly, riparian zone subcatchments also receive flood water from the river channel, but can also be configured to receive baseflow from upslope subcatchments into the subsoil layer and then into the topsoil layer if the subsoil layer is saturated.

Each subcatchment is assigned a unique integer identifier (1 - s), with the numbering starting at the most upstream subcatchment and increasing sequentially to the most downstream subcatchment, as shown in Figure 2-3. The streamflow network is created by specifying a downstream catchment for each subcatchment, except the lowest catchment.

The *ACRU* 3 version of the model simulates one subcatchment at a time, running an ordered list of processes for each day of the whole simulation period, before starting the simulation for the next subcatchment. The calculation order of subcatchments, processes and days are illustrated in Figure 2-3. The most upstream catchment (ID=1) is simulated first, followed by other subcatchments in order of their IDs, with the most downstream subcatchment (ID=s) being simulated last. Simulated streamflow values at the outlet of each subcatchment are stored by writing them to a direct access file which is read when modelling downstream catchments to enable the cascading of streamflows through the river network.

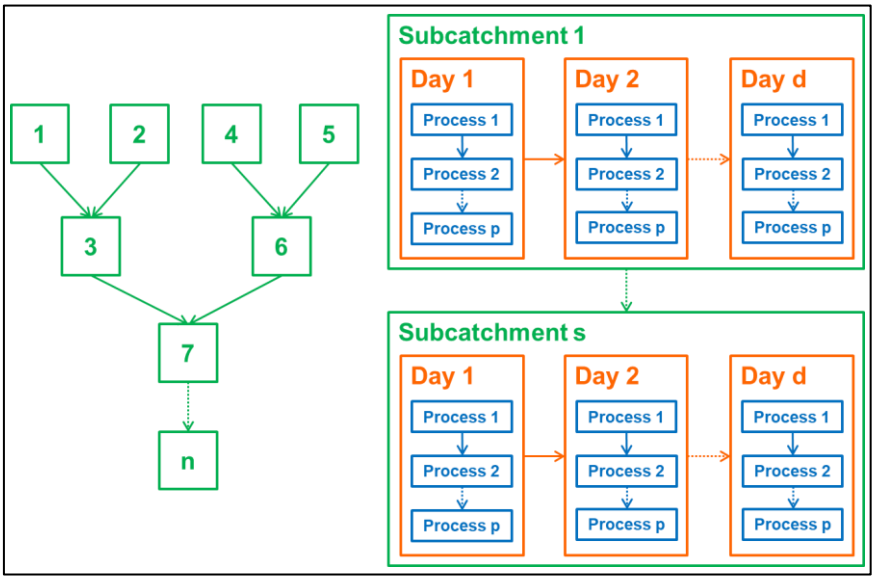

Figure 2-3 Calculation order of subcatchments, processes and days in *ACRU* 3

2.1.2 Representation of Hydrological Processes

The modelling of the water budget in *ACRU* has been designed to be sensitive to the effect of climate variables and to land cover/use changes on soil water and runoff conditions. The main processes represented in the *ACRU* 3 version of the model (Schulze, 1995a; Smithers and Schulze, 1995) for each subcatchment for each daily timestep are shown in Figure 2-1 and briefly described in this section. In any process-based hydrological model it is necessary to represent the hydrological processes as occurring in a set order within a model timestep, though in reality many of these processes may be occurring simultaneously. In

ACRU, as described in (Schulze, 1995b), the total evaporation processes occur first, followed by rainfall, interception, stormflow, infiltration, water movement through the soil profile and baseflow.

The A-pan reference potential evaporation (E_r), representing atmospheric demand, is determined, either as a measured value input by the user or estimated using one of several methods, including Penman (1948), Thornthwaite (1948), Blaney and Criddle (1950), Linacre (1977) and Hargreaves and Samani (1985), depending on the availability of data. Any remaining rainfall previously intercepted by the vegetation canopy or impervious surface cover is evaporated, limited by atmospheric demand, and the atmospheric demand is reduced accordingly. On vegetated areas the atmospheric demand is then multiplied by a crop coefficient to determine the maximum evaporation (E_m) which is partitioned into maximum transpiration (E_{tm}) and maximum soil water evaporation (E_{sm}). The estimation of soil water evaporation and transpiration is based on the method proposed by Ritchie (1972). Soil water evaporation occurs in the topsoil horizon and a two-stage process, based on soil water availability, is used to determine the actual soil water evaporation (E_s). The actual plant transpiration (E_T) occurs from both the topsoil and subsoil horizons, based on the distribution of roots between the two soil horizons, soil water availability and the sensitivity of the vegetation to water stress.

After evaporation, rainfall, if any, occurs. Rainfall is first intercepted by the vegetation canopy or impervious surface, depending on the interception characteristics of the vegetation or impervious surface and a possible reduction in interception capacity due to any unevaporated intercepted rainfall from the previous day. The remaining rainfall, net rainfall, is then used in a modified version of the Soil Conservation Service (SCS) method (USDA, 1985; Schmidt and Schulze, 1987) to estimate stormflow. On days when rainfall occurs the initial abstraction is calculated using the simulated soil water deficit and a user specified coefficient of initial abstraction. As the SCS method is intended to be used for small catchments (less than 30 km^2) and *ACRU* is often used on larger catchments, the stormflow is multiplied by a response factor to give a quickflow portion that runs off the catchment on the same day it was generated and a delayed stormflow portion that runs off on subsequent days. Runoff, which is the sum of quickflow, delayed stormflow and baseflow for a day, is added to any streamflow from upstream subcatchments to determine the streamflow leaving the catchment. If there is a dam at the exit of the subcatchment then the streamflow will enter the dam and the streamflow leaving the subcatchment will depend on the remaining storage capacity of the dam, seepage from the dam and releases from the dam for use by downstream users. Streamflow then flows to the next subcatchment downstream.

The portion of net rainfall that is not stormflow is infiltrated into the topsoil horizon. Saturated flow occurs from the topsoil horizon to the subsoil horizon if the soil moisture in the topsoil horizon exceeds the drained upper limit (field capacity). Similarly saturated flow occurs from the subsoil horizon to the baseflow store. There is also an option to model unsaturated redistribution of water between the topsoil and subsoil horizons. Baseflow is estimated by applying a simple decay function based on a user specified baseflow coefficient.

Although *ACRU* has the facility to lag and attenuate hydrographs through river and dam reaches this is not often used as the necessary data inputs are generally not easily available and simulations at a sub-daily timestep take a long time to complete. If the catchment includes an irrigated area then the source of water may be from the dam within the same subcatchment, if there is one, or from run-of-river. There is a so-called 'loopback' option within *ACRU* that enables an irrigated area to have a dam in an upstream catchment as a water source, however, due to the execution sequence of the *ACRU* model shown in Figure 2-3, this is not computationally efficient and is thus very seldom used. Other non-irrigation engineered transfers of water into dams and out of rivers and dams are modelled in a simple manner using 12 averaged month-of-year flow values as input.

In some cases options are provided in the *ACRU* model enabling users to select from two or more methods of representing a specific hydrological process depending on the relative accuracy and detail required with which the process is to be represented. However, the accurate representation of processes depends on the availability and accuracy of model input data.

This section has provided a very brief description of the main processes represented by the *ACRU* model, but further details of these processes and the representation of processes occurring in the impervious area, irrigated area, riparian zone, wetland and dam subcomponents of a subcatchment can be found in Schulze (1995a). The representation of hydrological processes is not covered in detail in this document as the focus of the research is on the structure of the model, not the addition of, or improvements to, the representation of hydrological processes.

2.1.3 Model Input Files and Data

The physical nature of the model means that it has a high input data requirement relative to a parameter fitting type model, however, this also means that the model can be used in

situations where long records of measured data are not available for use in calibration, such as in ungauged catchments and for modelling climate change scenarios. Although *ACRU* is a daily timestep model, rainfall is the only input variable for which daily time series input is required, although daily time series of other climate variables may also be used as input to the model, if available. The more cyclic and less sensitive time varying variables such as reference potential evaporation, air temperature and vegetation characteristics may be input to the model as time series of 12 mean month-of-year values (Schulze and Smithers, 2004). These mean month-of-year time series are transformed within *ACRU* using Fourier Analysis to produce year-long times series of daily values (Schulze and Smithers, 2004). The *ACRU* 3 version also provides for a seldom used facility enabling the values of typically static input variables and mean month-of-year time series variables to change dynamically at user specified months and years during the simulation period.

Each configuration of the model the *ACRU* 3 version (Schulze, 1995a; Smithers and Schulze, 1995) uses a single flat text format input file (known as the "menu" file), containing subcatchment configuration information, static variables and mean month-of-year time series variables, and references to a set of text files, where each text file contains the daily time series inputs for a single subcatchment. The daily time series files may be in *ACRU* Single, Composite or CompositeY2K format, described by Smithers and Schulze (1995).

2.2 Rationale for the Restructuring and Proposed Further Development

In the late 1990s it was recognised that, to meet new modelling requirements and to benefit from advances in software engineering practices, a new phase of model development was required. Increasing pressure on water resources and the move towards implementing the National Water Act (NWA, 1998) required new modelling capabilities and tools to enable better understanding and more effective management of water resources (Kiker *et al.*, 2006; DWA, 2013). These requirements include: (i) improved input and output tools to facilitate understanding and interpretation of hydrological information, and (ii) improved model performance, functionality and extensibility to better represent water resources systems (Kiker *et al.*, 2006). In 1998 the developers of the *ACRU* model received funding from the Water Research Commission (WRC) for WRC Project 636 to further develop and provide user support for the model (Lynch and Kiker, 2001a). This funding provided the opportunity to restructure the *ACRU* model and resulted in the *ACRU* 2000 version, referred to in Table 2-1 and Table 8-1, and described by Clark *et al.* (2001), Kiker (2001), Kiker and Clark (2001b), Kiker and Clark (2001c) and Kiker *et al.* (2006). This restructuring of the *ACRU* model forms part of the development presented in this document. The rationale for

restructuring the model and other further development of the model, using the model characteristics listed in Table 1-1 as a guideline, are described in this section.

One of the main reasons for restructuring the *ACRU* model was to make it more extensible (Clark *et al.*, 2001; Kiker, 2001; Kiker and Clark, 2001b; Lynch and Kiker, 2001b; Kiker *et al.*, 2006). Since its inception numerous research staff and postgraduate students at the University of KwaZulu-Natal and collaborating researchers from other institutions have made numerous additions and enhancements to the model. The result of these many contributions to the model over the years was a code framework within which it was becoming increasingly more difficult to implement new additions to the model. At that time the use of the FORTRAN 77 programming language for the model provided some advantages with respect to computational efficiency but also resulted in disadvantages with respect to developing a modular, easily expandable model structure (Kiker and Clark, 2001b; Lynch and Kiker, 2001b). Clark *et al.* (2001) explain that a more modular structure was required for the model such that: (i) the physical hydrological components are more explicitly defined, (ii) hydrological processes may be easily added or changed without affecting the rest of the model code, and (iii) new model input variables and parameters may be easily added.

More powerful computers, more advanced computer programming tools and more advanced remote sensing technology have made it possible to better represent real-world complexity in computer models. The limited sub-units permitted within a subcatchment in *ACRU* 3, shown in Figure 2-2, limits flexibility to represent real-world complexity. Thus, Clark *et al.* (2001) state that another of the main reasons for restructuring the model was to represent the individual spatial elements of the hydrological system being modelled more explicitly, together with enabling more flexible configurations of these spatial elements and the order of execution of the hydrological processes. In addition, the *ACRU* 3 subcatchment numbering system does not enable subcatchments to be easily subdivided after initial configuration.

Engineered water flows, such as inter-catchment transfers, and related water supply and demand issues are increasing in importance as catchments are developed and pressure on available water resources increases. As described in Section 2.1.1 and shown in Figure 2-3, each subcatchment is simulated for the entire simulation period before water is cascaded to the downstream subcatchment. This sequence of execution is computationally efficient, but it has limitations when representing engineered flows between different subcatchments, as water user requirements and water source availability need to be evaluated within the same model timestep. Clark *et al.* (2001) explain that a parallel processing approach was required

such that each process on each subcatchment is executed each day before continuing to the next day, as shown in Figure 2-4. Related to this was a need to enable more flexible specification of operating rules for extraction of water from rivers and dams (Kiker and Clark, 2001b; Lynch and Kiker, 2001b).

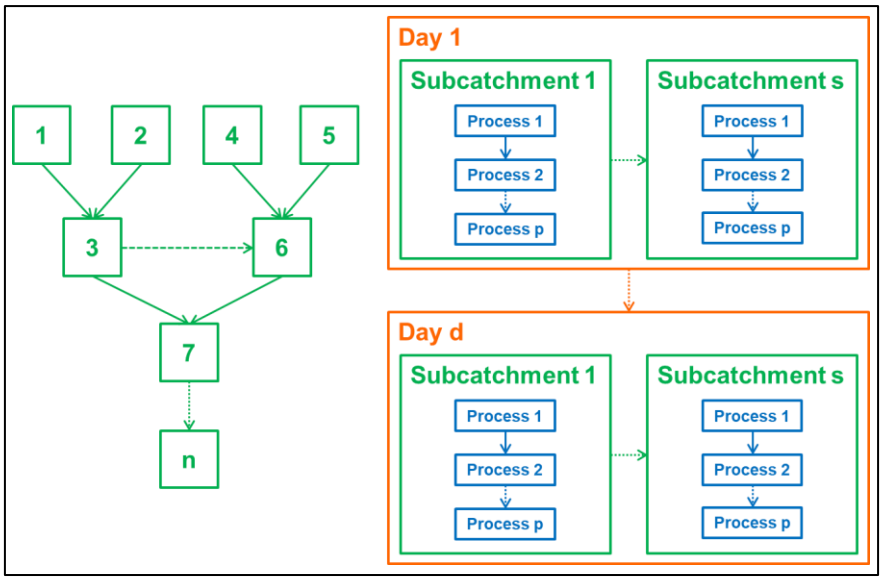

Figure 2-4 Calculation order of subcatchments, processes and days in a parallel processing approach

The *ACRU* model was developed as a daily timestep model and many of the subroutines representing hydrological processes are designed to operate at a daily timestep. The exceptions to this are the runoff hydrograph generation and flow routing subroutines that can be run at timesteps between 30 minutes and one day. A daily timestep makes sense, as many of the climate driven hydrological processes have a diurnal cycle which cannot be adequately represented using a monthly timestep and the data required to run the model at sub-daily timesteps is seldom available, especially at a large catchment scale. However, technologies such as remote sensing are improving the availability of daily and sub-daily data and so the model should ideally be able to make use of data at whatever time scale is available to suit the purpose of each individual modelling study. Some catchment characteristics such as land cover may change periodically during long simulation periods, and the *ACRU* 3 version of the model included a means to specify these dynamic changes, but this was not integrated into the usual model input files and was thus not easy to use. Kiker and Clark (2001b) and Lynch and Kiker (2001b) identified the need for the model structure to enable variable timesteps and data driven model input timesteps to be used. Clark *et al.* (2001) stated that the restructured model should not only enable simulation of the same hydrological processes at the same spatial and temporal scales as the *ACRU* 3

version, but that its structure should be designed to accommodate modelling at a wider range of spatial and temporal scales, if required in the future.

The *ACRU* model has traditionally been used for planning purposes. To be suitable for operational modelling, some modification and further development would be required, including hot-starting, persistence of state variables and better handling of time series data to accommodate the use of near real-time remotely sensed data and climate forecasts. The term “hot-starting” refers to the ability to initialise the model with the values of state variables determined from measurements or from previous simulations.

Research into suitable methods of restructuring the *ACRU* model began in 1998 and resulted in the decision to use object-oriented design and programming techniques. The object-oriented design technique has grown in popularity and usage since the early 1990s and was accepted by the *ACRU* developers as an intuitive way of modelling complex real-world systems, including water resources systems, in a conceptual manner (Clark *et al.*, 2001). Object-orientation also lends itself to the creation of models with a modular structure and more explicit representation of water resource system components and processes (Clark *et al.*, 2001). Some characteristics of object-oriented programming languages, such as inheritance and polymorphism, facilitate model designs that make models more extensible (Kiker *et al.*, 2006). Clark *et al.* (2001) concluded that object-orientation was an appropriate technique for designing a physical conceptual model such as *ACRU*. In addition, object-orientation would facilitate the development of a program structure to meet many of the identified development requirements identified for *ACRU*. The Java programming language was relatively new in 1998, but was selected for the implementation of the new object-oriented structure of the *ACRU* model, as Java was designed for object-oriented programming and offered the additional advantage that the code could be run on a variety of computer operating systems.

As discussed in Section 2.1.3, the *ACRU* model uses simple text-based model input files. One aspect of restructuring the *ACRU* model that was not clearly recognised at the time of the initial restructuring was that a new model input data structure would be required to complement the object-oriented model structure. This became evident after restructuring, and the Extensible Modelling Language (XML) was identified as a suitable way to structure the new model input data files.

As stated in Section 1.3 there is a need for multidisciplinary integrated modelling tools to support IWRM. There are two approaches to developing these integrated modelling tools,

either by extending the functionality of existing models, or by linking existing domain specific models. It would be impractical to develop *ACRU*, or any other hydrological model, to include all the domains present in complex water resource systems. However, it was recognised that there would be great value in implementing a suitable model coupling mechanism in the *ACRU* model, through which it could be flexibly coupled to other existing models representing different domains (Kiker and Clark, 2001b; Lynch and Kiker, 2001b).

This chapter has provided an overview of the concepts and structure of the *ACRU* model, especially the *ACRU* 3 version of the model, the rationale for restructuring the model and has also proposed some potential areas for further development. The *ACRU* 3 version is the reference point against which the development of the object-oriented *ACRU* 5 version, which was the main subject of this research, should be evaluated. The use of the object-oriented programming technique for developing water resources models and its use in restructuring the *ACRU* model are discussed in more detail in Chapter 3.

3 OBJECT-ORIENTED RESTRUCTURING OF *ACRU*

The concepts and structure on which the *ACRU* 3 version of the model was based and the rationale for restructuring the model are outlined in Chapter 2. This chapter starts with an introduction to the basic concepts of object-orientation and a brief review of literature related to the suitability of object-orientation for modelling water resources systems. The object-oriented restructuring of the *ACRU* model and the development of a complementary new model input structure using XML is then described.

3.1 Brief Overview of Object-Oriented Modelling

The world can, to a large extent, be perceived as being composed of discrete, physical, interrelated objects. Each of these objects has three main characteristics: identity, state and behaviour. Some objects may be simple, while others are complex and are composed of a hierarchy of smaller constituent objects. In the context of modelling, object-orientation may be described as an approach to thinking about problems using models with structures based on real-world concepts using hierarchical collections of discrete but interrelated objects that are characterised by both their attributes and behaviour (Rumbaugh *et al.*, 1991; Gärtner *et al.*, 2001; Bian, 2003). Object-oriented modelling is the abstract representation of real objects in computer code, as they are perceived by humans, rather than a linear sequence of calculations (Egenhofer and Frank, 1992; Silvert, 1993; Wang *et al.*, 2005b; Bian, 2007). The emphasis is on design and structure rather than the coding details (Cook and Daniels, 1994; Wang *et al.*, 2005b). Object-oriented programming is the result of an evolution in computer programming languages that started with assembly languages, the subsequent progression to process-oriented languages and later to object-oriented languages (Wegner, 1990). Object-orientation provides a new and versatile paradigm for thinking about and solving problems, and Wegner (1990) states that "*Its universality as a robust representation, modelling, and abstraction technique suggests that the object-oriented paradigm is conceptually and computationally fundamenta*l.".

In the literature the terms Object-Oriented Analysis (OOA), Design (OOD) and Programming (OOP) are used. These refer to three levels of abstraction, where: (i) OOA is the development of a conceptual model of real-world objects, relationships and events, (ii) OOD is the definition of a formal model of these real-world objects, relationships and events, and (iii) OOP is the code implementation of the OOD in a programming language (Bian, 2003; Bian, 2007). Thus, object-orientation can be applied to water resource modelling as: (i) a

means of representing the real-world system being modelled to promote understanding, and (ii) a programming technique for implementation in computer code (Rumbaugh *et al.*, 1991; Bian, 2003). Object-orientation provides an opportunity to explore new methods of representing complex hydrological phenomena (Wang *et al.*, 2005b).

3.1.1 Basic Concepts of Object-Orientation

A few of the basic concepts of object-orientation including objects, classes, inheritance, aggregation, association, polymorphism and encapsulation are explained briefly in this section. Detailed explanations of the concepts on which object-orientation is based, and the application of these concepts, can be found in books such as those by Rumbaugh *et al.* (1991) and Booch (1994).

An object is defined by Rumbaugh *et al.* (1991) as "*a concept, abstraction, or thing with crisp boundaries and meaning for the problem at hand*". Objects have identity, attributes which describe their state, and behaviour which can change the state of the object (Bian, 2003; Bian, 2007). Neither real-world nor software objects can be uniquely identified by their attributes or behaviour, and thus require a unique identity in order to be distinguishable and persistent (Wegner, 1990; Rumbaugh *et al.*, 1991). Object-orientation does not provide any strict principles of how objects and their attributes, behaviour and relationships should be defined, thus the conceptual interpretation and definition of objects will depend on the nature of the problem and the judgment of the modeller (Rumbaugh *et al.*, 1991; Bian, 2007). Bian (2007) suggests five criteria for identifying spatial entities: (i) spatial scale, (ii) the existence of a boundary, (iii) a common set of attributes, (iv) common behaviour or processes, and (v) type of mobility. Bian (2007) then goes on to differentiate between spatial objects, which have clearly defined physical boundaries, and spatial regions which are a portion of continuous space with definable but non-physical boundaries, where both spatial objects and spatial regions may be represented by software objects. However, Bian (2007) notes that there is a danger that object-orientation may be applied inappropriately to non-discrete phenomena.

Objects with the same attributes and behaviour can be represented by a common class, which is an abstraction of the objects it represents (Rumbaugh *et al.*, 1991; Bian, 2003; Bian, 2007). In object-orientation a class represents a blueprint, template or description of a collection of similar objects, where objects are described as being instances of a particular class, and the act of creating an object is referred to as instantiation (Wegner, 1990; Wang *et al.*, 2005a). However, some classes may be declared to be abstract, where an abstract

class can have subclasses but may not itself be instantiated to become an object. Silvert (1993) describes the concept of classes, which are important in object-oriented analysis, design and programming, as one of the most intuitive and useful characteristics of this approach. Closely related to classes is the concept of interfaces. An interface is an object-oriented design and programming entity that specifies a behaviour set. Any class that implements an interface is obligated to implement the behaviour specified in that interface.

Inheritance is the sharing of attributes and behaviour between classes based on a hierarchical relationship (Rumbaugh *et al.*, 1991). Inheritance enables classes to be organised into a hierarchy where subclasses inherit attributes and behaviour from their parent class, known as their superclass, helping to reduce code redundancy (Egenhofer and Frank, 1992; Bian, 2003; Bian, 2007; Molina *et al.*, 2010). Generalisation occurs as one moves up the class inheritance hierarchy, and specialisation as one moves down the class hierarchy (Egenhofer and Frank, 1992; Gärtner *et al.*, 2001). Single inheritance is a simplification of the real-world as it is strictly hierarchical, where each class has only one immediate superclass and belongs to only one hierarchy, however multiple inheritance is difficult to implement and use, as complex rules are required to resolve clashes in cases where attributes or behaviour with the same name are inherited from both superclasses (Egenhofer and Frank, 1992). Inheritance is a powerful means of generalising models, and the ability to reuse and modify code through inheritance is one of main advantages of object-oriented modelling (Silvert, 1993). Inheritance enables “type of”, “kind of” or “is a” relationships between objects to be specified.

The concept of composition refers to the description of the way in which objects are related to each other, or organised, through inheritance, aggregation and association (Bian, 2003; Bian, 2007). Aggregation, refers to the fact that an object can consist of other sub-objects, in other words, objects may be composite (Egenhofer and Frank, 1992; Bian, 2003; Bian, 2007). Aggregation results in “part of” and “consists of” relationships. Association refers to the existence of a relationship between two or more independent objects (Egenhofer and Frank, 1992). Association enables the specification of specialised relationships between objects that cannot be represented by inheritance or aggregation, they are “interacts with” relationships (Bian, 2003; Bian, 2007).

Polymorphism is the mechanism that enables the behaviour of an object to vary depending on the class of the object being acted on, the class of object performing the action or the information received by the object (Rumbaugh *et al.*, 1991; Silvert, 1993). Molina *et al.* (2010) describe polymorphism as enabling objects to have different natures.

The concept of encapsulation refers to the containment of identity, attributes and behaviour within objects (Bian, 2003; Bian, 2007). Encapsulation provides the ability to hide some internal details of an object, and expose only those details required for interfacing with other objects in a controlled manner (Silvert, 1993; Molina *et al.*, 2010).

The Unified Modelling Language (UML) is a notation developed to visually describe object-oriented models of real-world systems to facilitate communication and enable model designs to be documented in a manner that is independent of the computer programming language used to implement the model. A UML specification (OMG, 2017) is maintained by the Object Management Group (OMG). UML class diagrams provide a static view of a set of classes including attributes, behaviour, inheritance, aggregation and association. A brief overview of the UML notation for class diagrams used in this thesis document can be found in Appendix 8.2.

3.1.2 Suitability for Modelling Water Resources Systems

Object-orientation is a powerful and transparent approach to water resources modelling (Simonovic *et al.*, 1997; Ghashghaei *et al.*, 2013) and has increasingly been used for development of water resource management software (Leone and Chen, 2007). Object-orientation is a modelling tool that promotes understanding of the water resource system being represented (Gärtner *et al.*, 2001). Alfredsen and Saether (2000) state that not only is object-orientation well suited to representing hydrological systems, but given the increasing requirement for integration of models within hydro-informatic systems, object-oriented modelling should provide clear advantages in the field of hydrological modelling.

IWRM requires a systems approach, taking into account systems and subsystems consisting of collections of related components that interact with each other. Real-world systems, including natural hydrological and ecological systems, have a structure that can be represented in a natural, direct and intuitive manner using the object-oriented paradigm which has its origins in the field of system simulation (Simonovic *et al.*, 1997; Alfredsen and Saether, 2000; Wang *et al.*, 2005a; Wirth, 2006). Thus, object-oriented models, where real-world entities are represented by programming objects that combine attributes and behaviour, have a structure that strongly resembles the system being modelled (Silvert, 1993; Lafore, 2002; Wang *et al.*, 2005a). The object-oriented approach to modelling focuses on identifying the interrelated objects making up the application domain, and then fitting the functionality around them, making such models more robust as they evolve (Simonovic *et al.*, 1997). This is in contrast to traditional procedural approach where the focus is on data

and model functionality, typically implemented as a linear sequence of calculations, describing the manner in which variables values change with time (Wegner, 1990; Silvert, 1993; Wang *et al.*, 2005a; Wang *et al.*, 2005b). Thus, object-oriented modelling requires a paradigm shift in the manner hydrological processes or events are conceptualised (Wang *et al.*, 2005a).

Kiker *et al.* (2006) observe that hydrological systems are complex in themselves, are rendered more so by engineering and agricultural developments, and that further complexity is added when simplified abstract representations of these systems are created in computer code. Object-oriented design is one programming technique available to try to manage this complexity, and to model hydrological systems in a more intuitive manner (Band et al., 2000; Clark et al., 2001; Kiker and Clark, 2001a; Wang et al., 2005b). Object-orientation has provided researchers with an approach that enables them to model systems that would have previously been difficult to represent (Bian, 2007), and is a powerful conceptual tool for describing complex hydrological processes and systems (Wang *et al.*, 2005a; Wirth, 2006). Object-oriented model design enables complex natural systems to be more closely replicated due to the linking of objects and their actions (Wang *et al.*, 2005a). Wang *et al.* (2005a) state that using the object-oriented approach they were able to constrain complexity by increasing simulation flexibility, and conclude that object-oriented design and the concept of classes is a powerful conceptual tool for describing complex hydrological processes.

Conceptually object-orientation is based on the assumption that the real-world consists of discrete physical entities and discrete processes and this is replicated in the data model (Reitsma and Carron, 1997; Bian, 2003). However, natural phenomena are often conceptually continuous, and this conceptual mismatch may lead to difficulties when formalising representations of the environment using the object-oriented approach (Bian, 2003; Bian, 2007). The representation of continuous phenomena as conceptually discrete entities is also a challenge in Geographic Information Systems (GIS). A typical approach, in both GIS and object-oriented modelling, is to artificially partition spatially varied environments to create objects consisting of landscape features with similar form and function (Bian, 2003; Galton, 2004).

The object-oriented concepts of classes, inheritance, aggregation and association enable the development of water resource models that are modular, extensible and flexible (Martinez *et al.*, 2008). The application of object-oriented design principles results in models that are inherently modular, where a well-designed class and inheritance structure: (i) simplifies the implementation of different modules containing application or location specific

modelling functionality, and (ii) enables separate parallel development of different modules within a model (Silvert, 1993; Sydelko *et al.*, 1999; Kiker *et al.*, 2006). Inheritance and polymorphism make object-oriented models easier to extend, whether adding either improved algorithms or new functionality, without changes to existing classes, thus leaving existing functionality intact (Sydelko *et al.*, 2001; Bian, 2003; Wang *et al.*, 2005a; Kiker *et al.*, 2006; Martinez *et al.*, 2008; Kumar *et al.*, 2010). Inheritance also reduces code length by reducing duplication of code (Lafore, 2002; Wang *et al.*, 2005b). Model structures based on interrelated objects enable more flexible configuration of models which helps in constraining complexity and in making models more versatile (Shane *et al.*, 1996; Sydelko *et al.*, 2001; Lafore, 2002; Wang *et al.*, 2005a; Kang *et al.*, 2016). The object-oriented approach can simplify the design of models leading to more efficient programming and lead to improved maintainability of model code (Lafore, 2002; Wang *et al.*, 2005b; Kang *et al.*, 2016). There are thus many advantages to the use of the object-oriented approach for hydrological models, both in terms of representation of the system being modelled and as a method of programming.

Although there were some applications of object-oriented programming in hydrological modelling, there are no guidelines related to object-oriented design principles and how to implement these in hydrological models (Wang *et al.*, 2005a; Wang *et al.*, 2005b; Kiker *et al.*, 2006; Kang and Merwade, 2011; Kang *et al.*, 2016). In addition, literature describing object-oriented hydrological models does not include details of the designs. Some possible reasons for this are that: (i) many established legacy models may not have been restructured using object-oriented design principles, (ii) models are coded in an object-oriented programming language, but do not have an object-oriented design, and (iii) the object-oriented designs of the models have not been described in published literature as the design is proprietary or difficult to communicate in a concise manner. A few examples from the literature, of water resource related models that have used the object-oriented design approach, include the following:

- HEC-HMS - the Hydrologic Engineering Centre's Hydrologic Modelling System for rainfall-runoff simulation (Charley *et al.*, 1995).
- Regional Hydro-Ecological Simulation System (RHESSys) - a spatially distributed hydro-ecological model for simulating water, carbon, and nutrient cycling and transport in catchments at a hillslope level (Band *et al.*, 2000; Tague and Band, 2004).
- Identification of unit Hydrographs And Component flows from Rainfall, Evaporation and Streamflow (IHACRES) – a catchment-scale rainfall-runoff model that uses rainfall and temperature data to predict streamflow (Croke *et al.*, 2005).

- OBJect-oriented TOPographic-based (OBJTOP) - a hydrological model based on the concepts of TOPMODEL (Beven and Kirkby, 1979), which is described as a semi-distributed catchment-scale hydrologic model in which topography is assumed to be the main driver of water flow through upland catchments, together with heterogeneous rainfall and soil type (Wang *et al.*, 2005a; Wang *et al.*, 2005b).
- Regional Simulation Model (RSM) - an object-oriented, physically-based, hydrologic model designed for the South Florida region (Lal *et al.*, 2005).
- RiverWare - a generalised, flexible modelling tool for simulation and optimisation of river and reservoir systems (Zagona *et al.*, 2001; Frevert *et al.*, 2006; Valerio *et al.*, 2010).
- STORE DHM – a storage release, grid-based, modular, distributed hydrologic model for application in a GIS (Kang and Merwade, 2011; Kang *et al.*, 2016).

3.2 Design and Development of an Object-Oriented *ACRU* Model Structure

A list of desired characteristics for a hydrological model, for use as a tool in IWRM, were identified from literature and summarised in Table 1-1. The rationale for restructuring the *ACRU* 3 version and the decision to use object-oriented design and the Java programming language were discussed in Section 2.2. There were three broad objectives for the restructuring: (i) greater flexibility in model configurations, (ii) a more extensible code structure, and (iii) better data handling. A list of more specific design objectives for the restructured *ACRU* 5 version of the model are shown in Table 3-1. The development of the design objectives for the restructured *ACRU* model was to some extent an iterative process, with a better understanding of the advantages of object-orientation and the modelling requirements for IWRM being developed over time.

During the restructuring and further development of *ACRU* a key consideration was always to, as far as possible, maintain the integrity of the *ACRU* model by not changing the algorithms representing hydrological processes in which trust has been built and years of research have been invested. Another key consideration was that the design of the new model structure should not be constrained by the limitations of the old model data input file structure.

Table 3-1 Design objectives for the restructured *ACRU* 5 version of the model

Greater Flexibility in Model Configurations
More explicit representation of the physical components of water resources systems, including the concept of hydrological response units (HRUs) within subcatchments.
Better representation of real-world complexity through more flexible configuration of spatial elements and the order of execution of components and processes.
A parallel processing approach, such that each process on each subcatchment is executed each day before continuing to the next day to enable feedbacks between systems to be simulated.
Better representation of engineered water flows within and between different subcatchments, taking into account water user requirements and water source availability within the same model timestep.
More flexibility in the configuration operating rules for extraction of water from rivers and dams.
Inclusion of the concept of water ownership to enable representation water use allocations.
A model structure that facilitates modelling water quality processes in addition to water quantity processes.
Suitable for both water planning and operations type modelling.
A More Extensible Code Structure
More modular code structure with explicit representation of the hydrological processes enabling the addition of new process representations or refinements to existing process representations without affecting the rest of the model.
Easier addition of new model input variables and parameters (and metadata describing them) associated with new hydrological process representations.
Better Data Handling
Provision for a wider variety of time series data including breakpoint data so that periodic changes in catchment characteristics can be modelled within a simulation period.
Data aggregation and interpolation functionality to facilitate data driven model input timesteps.
The ability for the model to be hot-started, together with persistence of state variables, to facilitate better model initialisation, especially for use in operational modelling.
The facility for different modelling scenarios to be easily and efficiently setup and run.
The ability for water balances of individual physical components to be easily queried and checked to facilitate the compilation of water accounts.
Provide access to model variables and appropriate model timestepping execution control by third-party software to facilitate the implementation of a model coupling interface, such as OpenMI, to enable *ACRU* to be linked to other models and modelling tools for use in integrated modelling.

The design and implementation of an object-oriented structure for the *ACRU* model is described in the following sub-sections. The author's contribution to the conceptual design and implementation is explained in the preface to this document. There were three main phases to the development of the design:

- Initial conceptual design by Kiker and David (1998), which is briefly described in Section 3.2.1 to provide the background for the design described in Section 3.2.2.
- Refinement of the conceptual design and implementation as the *ACRU* 2000 version of the model. Additional minor refinements were made to the design in the *ACRU* 4

version in conjunction with implementing the XML-based model input file format. These interim designs and implementations are not described in this document.

- Further development of the conceptual design and implementation as the *ACRU* 5 version, which are described in Section 3.2.2, amending identified limitations of the *ACRU* 2000 and *ACRU* 4 versions.

3.2.1 Initial Design of the Model Structure

A simple way to represent real-world objects in an object-oriented model would be to create a class for each type of object. Each class would contain instance variables describing the class attributes and methods describing the class behaviour. For example, a class could be created to represent a catchment, this class could have instance variables with simple data types containing the area and other characteristics of the catchment, and one or more methods containing algorithms describing the hydrological processes, such as total evaporation, within the catchment. However, there are many possible ways to structure a model using object-oriented design. The initial conceptual design by Kiker and David (1998) is shown in its simplest form in the UML class diagram shown in Figure 3-1.

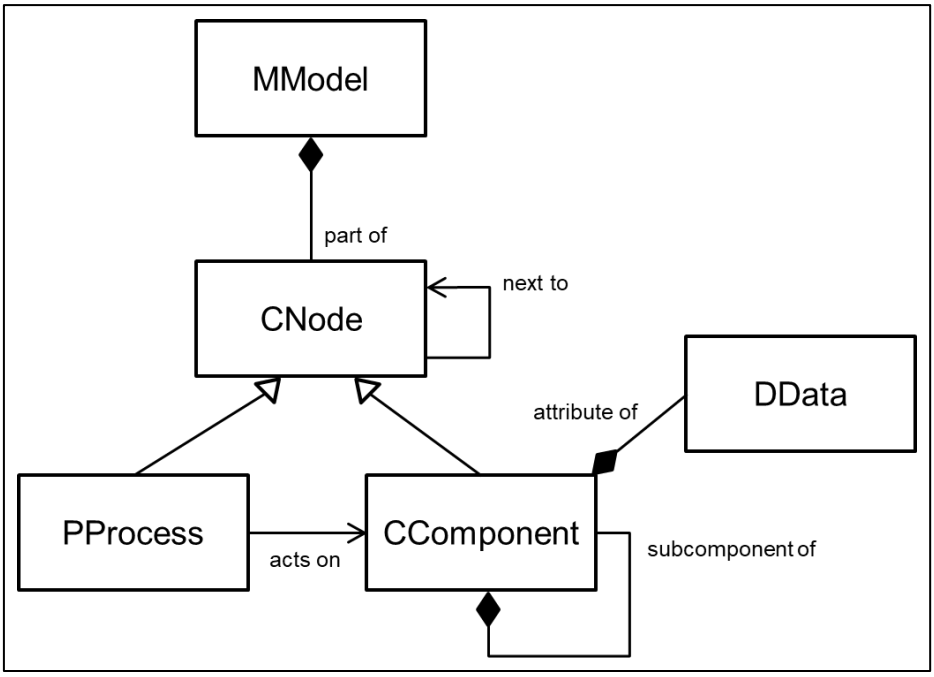

Figure 3-1 UML class diagram of the initial conceptual design for an object-oriented structure for the *ACRU* model (after Kiker and David, 1998)

* Refer to Section 8.2 for a guide to UML notation

In the initial conceptual design of a structure for *ACRU*, the use of object-orientation was taken a step further. The initial conceptual design by Kiker and David (1998) included the concept of *MModel*, *CComponent*, *PProcess* and *DData* classes, which are described below, and also the associations between these classes. Hydrological systems are conceived as being composed of a number of relatively discrete, but interrelated, physical components such as catchments, rivers, dams, vegetation and soil. In the new object-oriented model structure these physical components are represented by instances of the *CComponent* class. A *CComponent* class can have other *CComponent* classes as subcomponents, indicated by the "part of" relationship, for example, rivers and dams could be subcomponents of a catchment. The attributes of the physical components of the hydrological system are represented by instances of the *DData* class. This means that model variables and parameters are objects in the object-oriented model structure, and can thus be added to the model without needing to edit the code for the instance of CComponent that they describe. Other advantages of using data objects, instead of simple data types, are that the data objects can store metadata about the data values, such as units of measure, and enable variable specific range and error checking to be performed. Instances of *DData* would have a "part of" relationship with the instance of *CComponent* that they describe. Hydrological processes, for example, evaporation or runoff, are represented by subclasses of the *PProcess* class. An instance of *PProcess* may have an "association" relationship with one or more instances of *CComponent*, from which it either obtains input data values or sets simulated data values calculated within the process. In the initial design, both the *CComponent* class and *PProcess* class are defined as being subclasses of the *CNode* class, where the *CNode* class contains a set of instance variables and methods enabling relationships between instances of *CNode* to be specified and queried. Instances of the *MModel* class serve as the main container object for a model representing a specific hydrological system. Through the "part of" relationship between the *CNode* class and the *MModel* class, the *MModel* object consists of a collection of *CComponent* and *PProcess* objects. The class names each start with a letter indicating the type of class enabling them to be easily distinguished, thus the letters *M*, *C*, *D*, and *P* correspond to *MModel*, *CComponent*, *DData*, and *PProcess* classes respectively and their subclasses. The terms Model, Component, Data and Process are used to refer to the *MModel*, *CComponent*, *DData*, and *PProcess* classes and their subclasses in a generic manner. Although this structure was designed for the *ACRU* model it could conceivably be used to model other real world systems.

This initial conceptual design was then developed further as described by Kiker and Clark (1999). The main classes and interfaces that formed the foundation of the initial object-

oriented design of the *ACRU* model are described in more detail with the aid of UML class diagrams in Appendix 8.3. The initial conceptual design by Kiker and David (1998) and Kiker and Clark (1999) was largely a conceptual design using UML with very little implementation in Java code. This conceptual design was then fully implemented in Java code and almost the whole *ACRU* 3 version of the *ACRU* model was restructured from procedural FORTRAN 77 code to object-oriented Java code based on this initial design, resulting in the *ACRU* 2000 version of the model. Some aspects of the design of the *ACRU* 2000 model structure were published in Clark *et al.* (2001), Kiker (2001), Kiker and Clark (2001b), Kiker and Clark (2001c) and Kiker *et al.* (2006). The implementation of the initial conceptual design of the model structure resulting in the *ACRU* 2000 version of the model was a learning process and was, to some extent, a compromise between completely restructuring the model and constraining the model to certain concepts from the *ACRU* 3 version.

Application of the *ACRU* 2000 version built confidence in the restructured model and highlighted some limitations, which included:

- The model input file format limited use of the new object-oriented structure of the model to its full potential.
- A separate subclass of *DData* was created for each model variable, but most of these subclasses did not include additional functionality.
- A separate subclass of *CVegetation* was created for each vegetation type with no additional functionality.
- Representation of time series data was limited to daily and month-of-year data structures, as in *ACRU* 3.
- The need to enable time series data input using other file formats, as the *ACRU* 3 formats were difficult to edit and were limited to a specific list of variables they could contain.
- The need to handle state variables more explicitly and to be able to hot-start the model.
- The need to handle water and different types of constituents, such as Nitrogen, Phosphorus and sediment more explicitly.

Based on the identified limitations of the *ACRU* 2000 version two main areas of subsequent development took place as part of this study: (i) new XML format model input files were developed, as described in Section 3.3, with some changes to the model to read and use these files, resulting in the *ACRU* 4 version, and (ii) the model structure was refined, which

included some parts of the design being simplified and others being expanded, resulting in the *ACRU* 5 version described in Section 3.2.2.

3.2.2 Refined Design of the Model Structure

The main classes and interfaces that form the foundation of the *ACRU* 5 version of the model are shown in Figure 3-2. A Model may consist of one or more main Components, which may in turn consist of a set of nested sub-Components, each of which may have sub-Components of its own. Each Component knows which Model it belongs to. Both *CComponent* and *PProcess* are subclasses of the *CNode* class, sharing the functionality related to the *next* type associations between instances of *CNode*. These *next* associations each have a context that describes the type of association. For example, a river Component may be related to a downstream dam Component using a *next* association with a *downstream* context, and conversely the dam is related to the river in an *upstream* context. In addition to the *next* associations, instances of *CComponent* may also have a special association with an instance of *CComponent* that contains it. These *containerComponent* relationships are used to set up a flexible containment structure, for example, HRUs and river reaches within a subcatchment, or soil layers within a soil profile within an HRU. These containment structures could have been set up using conventional object-oriented aggregation relationships; however, these would not be flexible and would thus have been very *ACRU* specific.

Both *MModel* and *CComponent* implement the *IDataContainer* interface, thus, in addition to using Data objects to describe the characteristics of Component objects, a Model object may also have one or more Data objects assigned to it to represent general modelling options, parameters and variables. Conceptually, matter, such as water, nitrogen, phosphorus, sediment or biomass, fits somewhere between Components and Data. Thus, the *RResource* class was included in the design. Each Component may contain one or more modelled Resources, representing water and other water quality related constituents. The *RResource* class uses a special set of Data classes to describe a resource including quantity, location and ownership as states at any point in time during a simulation. The Resource classes have names starting with the letter R.

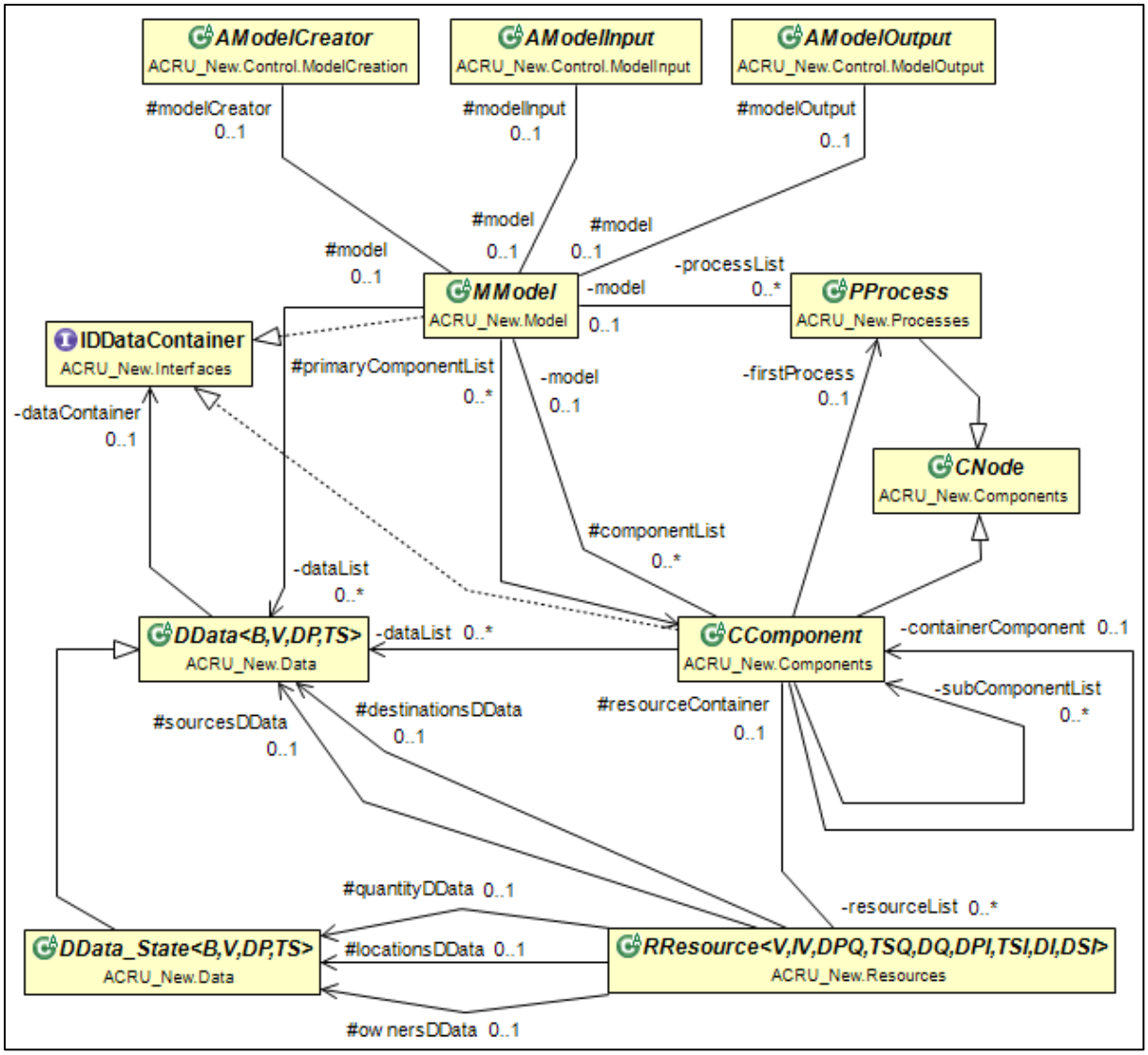

Figure 3-2 *ACRU* 5 design: main classes and interfaces

* Refer to Section 8.2 for a guide to UML notation

* Refer to Section 8.4.1 for a description of the Java generic types

Each Model has a list of Process objects and each Process object is aware of which Model object it belongs to. Components, usually Components representing spatial features such as catchments, HRUs and waterbodies, may each have a reference to a Process object, which is the first Process that should be run for the Component in each timestep. The first Process and subsequent Processes are related using *CNode next* associations, which in effect creates an ordered list of Processes to be run for a Component at each timestep.

An additional category of classes, known as the Control classes, was included in the design, and the names of these classes all start with the letter A. The Control classes are used to control the initialisation and running of an instance of *MModel*. Each instance of *MModel* is associated with: (i) an instance of *AModelCreator* to initialise an instance of MModel with its

constituent Component, Data and Process objects, (ii) an instance of *AModelInput* to coordinate reading in model input files and making the input available to an instance of *AModelCreator*, and (iii) an instance of *AModelOutput* which creates the model output files and coordinates the writing and saving of these files.

In the *ACRU* 5 version more explicit relationships between the classes were defined and extensive use was made of Java generic typing. The generic types used in Figure 3-2 are described in detail in Appendix 8.4.1. The root package name *ACRU_New* was used temporarily for the *ACRU* 5 version, but will be replaced by the package name *ACRU*. Each of the main class categories Model, Control, Component, Data, Resource and Process is described in more detail in Sections 3.2.2.1 to 3.2.2.6.

3.2.2.1 Model classes

The abstract *MModel* class is the generic superclass for one or more model specific implementations, thus, in addition to the *ACRU* model the abstract MModel class could be used as a foundation for the development of other time-stepping models. A model specific implementation of *MModel* acts as a container for a collection of Component, Resource, Data and Process objects that represent the real-world system being modelled. The *MModel* class has a Java *main* method, making it the starting point for model instantiation and execution. The main Model classes of the *ACRU* 5 version of the model are shown in Figure 3-3. The *MModel* class includes methods to: (i) initialise a model by reading in data and creating the appropriate Component, Data and Process objects, (ii) run the model, one timestep at a time, for the specified simulation period, and (iii) finalise the model by storing state data and closing model output files.

In real-world hydrological systems different processes occur simultaneously throughout a spatial and temporal continuum. In the *ACRU* model the spatial continuum is divided into discrete spatial units based on their hydrological characteristics. In an instance of *MAcruModel* the *computationOrder* instance variable contains a list of Component objects, representing spatial units, in the order in which they should be simulated for each timestep, and each Component has an ordered list of Processes. All the runoff generating land units are simulated first followed by the flow reach units in flow order starting with the upstream reaches and ending with the reach that exits the most downstream catchment.

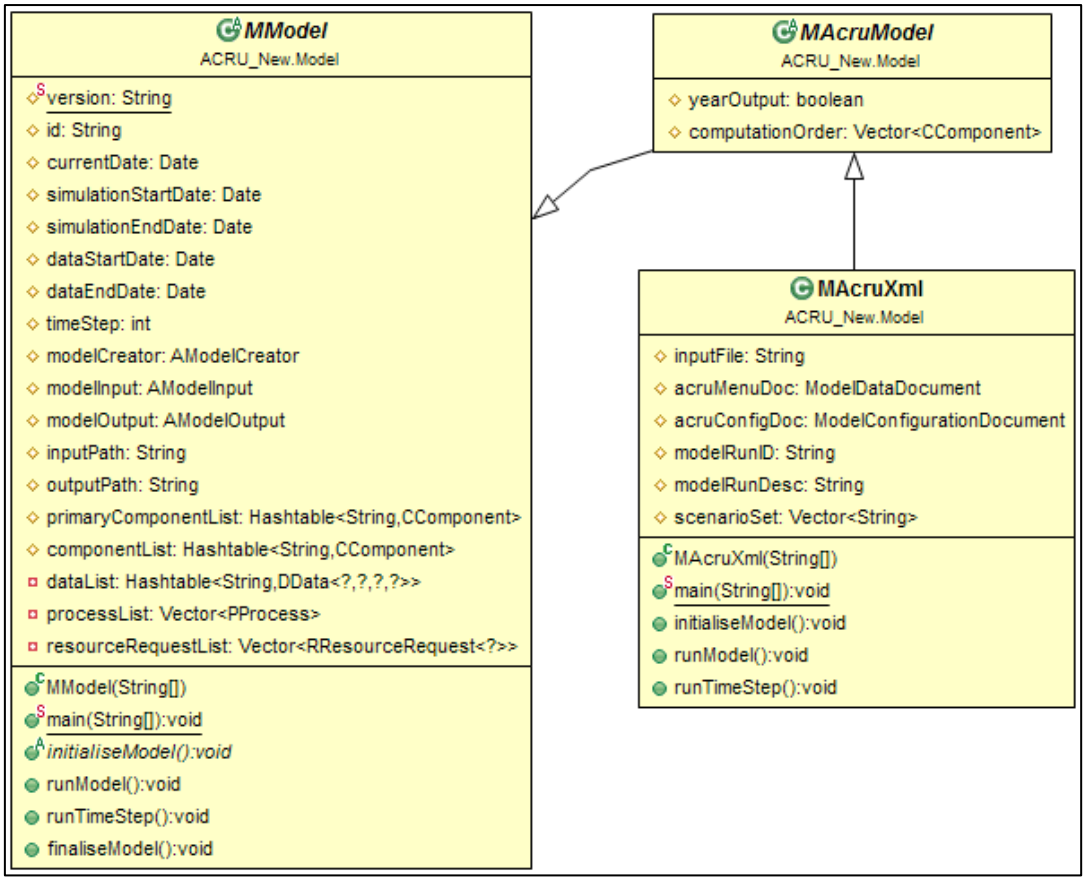

Figure 3-3 *ACRU* 5 design: Model classes

* Refer to Section 8.2 for a guide to UML notation

An important refinement was the inclusion of the *runTimeStep* method to explicitly run the model for a single time-step. The need for a method to explicitly run a single time-step was recognised when creating an OpenMI linkable component wrapper for the *ACRU* model, as this is one of the key requirements indicating suitability of a model to be made OpenMI compliant using the wrapper approach as described in Section 4.3.2. The abstract *MAcruModel* class was created to act as a superclass for one or more potential variations of the *ACRU* model. The *MAcruXml* class was created to represent a variation of the *ACRU* model that reads in model input files formatted using an XML schema developed specifically for the *ACRU* model (Section 3.3.1) and an XML *ACRU* configuration file (Section 3.3.2). The *MAcruXml* file has specific implementations of the *initialiseModel*, *runModel* and *runTimeStep* methods.

3.2.2.2 Control classes

The main Control classes are shown in Figure 3-4. For each of the Model classes there are associated model input, creation and output classes. The *AAcruXmlModelInput* class contains code used to open and read the XML formatted model input files. The

AAcruXmlModelCreation class contains code that uses information in the XML model input file (Section 3.3.1) and the model configuration file (Section 3.3.2) to configure an instance of the *MAcruXml* class with the appropriate Component, Data and Process objects for a study site. The *AAcruXmlProcesses* class specifies, together with options stated in the model input files, which Process objects are to be created for each Component and the order in which they are to be run. The *AAcruXmlModelOutput* class contains code to create the model output files and control the writing of model output values to these files.

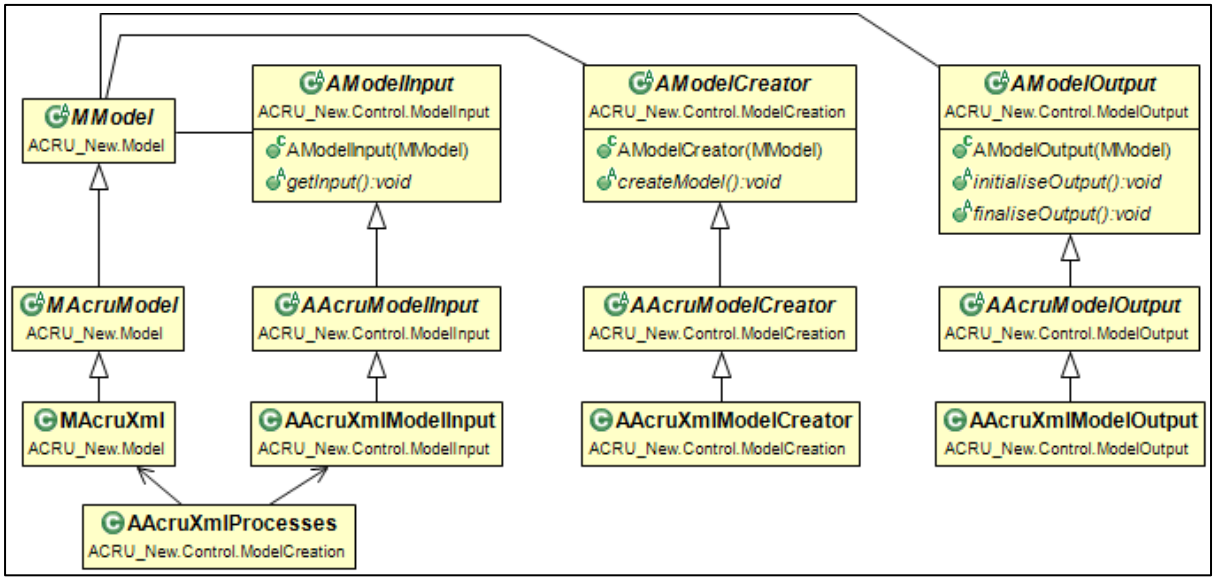

Figure 3-4 *ACRU* 5 design: main Control classes

3.2.2.3 Component classes

The design of the Component class structure was refined further for the *ACRU* 5 version of the model. The *CComponent* class is shown in Figure 3-5. Each instance of *CComponent* knows which container *CComponent* it is a part of, if any, and also contains a list of the immediate sub-Components of which the instance is comprised. The Component classes that represent the main spatial components of a hydrological system are shown in Figure 3-6. The abstract *CSpatialUnit* class is a superclass for all the Component classes that represent the spatial entities within a hydrological system; those that would typically be represented as a region, line or point on a map. In the *ACRU* 5 version of the model, the containment and association relationships between Components are specified in the model input file (Section 3.3.1) and the model configuration file (Section 3.3.2), and are not hardcoded in the Component classes. This input file based configuration enables the creation of different models or different model configurations without changing the

Component classes. However, this means that the containment and association relationships between Components are not shown in the UML class diagrams.

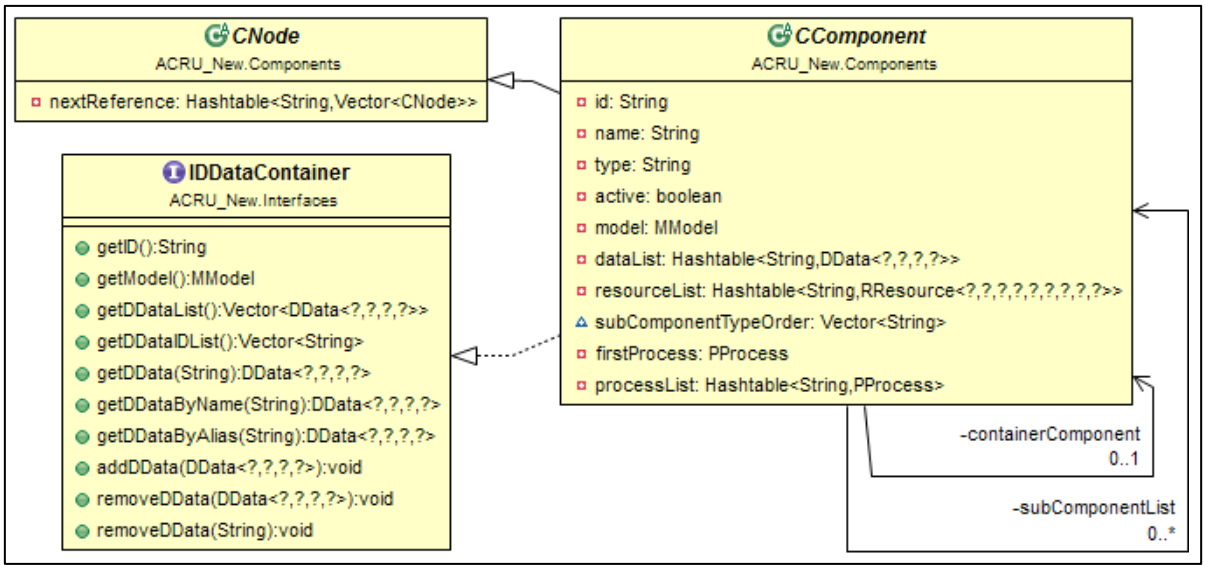

Figure 3-5 *ACRU* 5 design: main Component classes

The abstract *CLandSegment* class represents units of land on which runoff is generated, and the abstract *CReach* class represents reach segments of the flow network. The *CHRU* (hydrological response unit) class represents ordinary portions of land. The *CHRU* class typically represents portions of land with natural vegetation or dryland crops as a land cover. The *CSpecialRegion* subclass of *CLandSegment* is an abstract superclass for a set of classes that represent portions of land for which the land cover/use requires specialised Process classes to be used. These specialised regions include impervious areas, wetlands, riparian zones and irrigated areas. The concept of catchment Components was developed further in the *ACRU* 5 version. The *CSpatialUnit* class includes subclasses named *CCatchment* and *CSubCatchment*. An instance of the *CSubCatchment* class acts as a container for the instances of *CLandSegment* and *CReach* that together represent the spatial subcomponents of a subcatchment. An instance of the *CCatchment* class acts as a container for either: (i) a set of instances of *CSubCatchment* representing subcatchments, or (ii) a set instances of *CCatchment* representing smaller nested catchment areas within the catchment being represented. This system of catchments and subcatchments enables the conceptual representation of physical hydrological catchment areas contained within each other, and also enables the aggregation of modelled variables such as in catchment-scale water resource accounts.

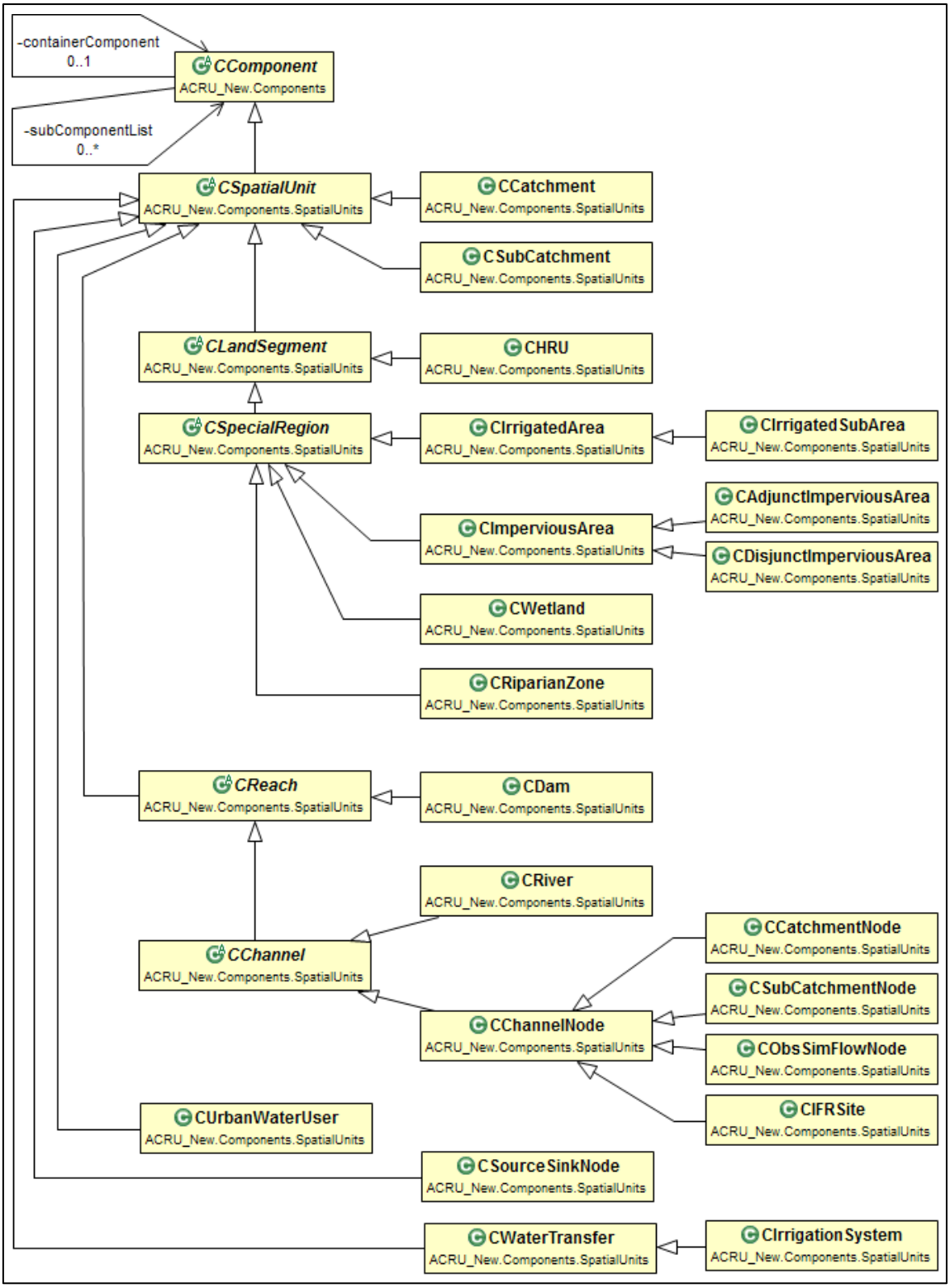

Figure 3-6 *ACRU* 5 design: main spatial Component classes

The *CReach* class has two subclasses: *CDam* representing impoundments that form part of the flow network, and the abstract *CChannel* class representing all other flow reaches. The *CChannel* class has subclasses; the *CRiver* class represents river reaches and the *CChannelNode* class represents the connecting points (nodes) between river and dam

reaches. The *CChannelNode* class is also used to represent points where: (i) runoff enters the flow network, and (ii) where engineered inflows to, and outflows from, the flow network occur. Subclasses of the *CChannelNode* class represent specialised nodes such as in-stream flow requirement (IFR) sites and nodes where observed flow values are used to replace simulated flow values. The *CCatchmentNode* and *CSubCatchmentNode* classes were included as specific subclasses of *CChannelNode* to represent the flow node at the downstream exit of catchments (*CCatchment*) and subcatchments (*CSubCatchment*) respectively. The *CSourceSinkNode* class represents external sources of water into the catchment area being modelled and external sinks of water out of a catchment, such as inter-catchment transfers into or out of a catchment. The *CWaterTransfer* class is used to represent engineered transfers of water between Components, such as the transfer of water from a dam to an irrigated area. The *CWaterTransfer* class has a specialised subclass *CIrrigationSystem* which is associated with a *CIrrigatedArea* class and holds water received from a water source until it is applied to the irrigated area. The *CUrbanWaterUser* class is used to represent urban water users.

An example is shown in Figure 3-7 of how a spatial unit of land, such as an HRU, is represented conceptually using Component classes both in a spatial sense and as a set of vertical subcomponent layers. Spatially an instance of the *CHRU* class would be part of an instance of the *CSubCatchment* class which in turn would be part of an instance of the *CCatchment* class. An instance of *CHRU* could typically be composed of several vertical subcomponent layers such as instances of *CClimate*, *CLandCover*, *CSoil* and *CGroundwater*. The subcomponents of the *CWetland*, *CRiparianZone* and *CIrrigatedArea* Components are represented in a similar way to *CHRU* with some small variations. The subcomponents of the *CImperviousArea* components are different to those of CHRU, as described briefly in Appendix 8.4.2.

An instance of the *CClimate* class is used to represent the climate or atmosphere above a unit of land, and it delivers precipitation and receives evapotranspired water. Climate variables such as rainfall, evaporation potential and temperature are important inputs to a hydrological model, however, the availability of climate data at a suitable spatial scale is often a problem and availability also varies between study catchments. Instruments such as rain gauges and evaporation pans generally have a sparse spatial distribution and it may be necessary to use data from the same instrument to represent the climate in several surrounding catchments. On the other hand the estimation of climate variables using remote sensing may enable better representation of the spatial variability of climate variables. Each instance of the *CSpatialUnit* class may optionally have an instance of the

CReferenceClimate class as a subcomponent, such that in some cases each individual instance of *CHRU* may have its own individual instance of *CReferenceClimate*, while in other cases all the instances of *CHRU* belonging to a subcatchment may be associated with a single instance of *CReferenceClimate* which is a subcomponent of the subcatchment. This arrangement of *CClimate* and *CReferenceClimate* also serves the purpose of reducing repetition of climate datasets in memory in cases where instances of *CReferenceClimate* are shared.

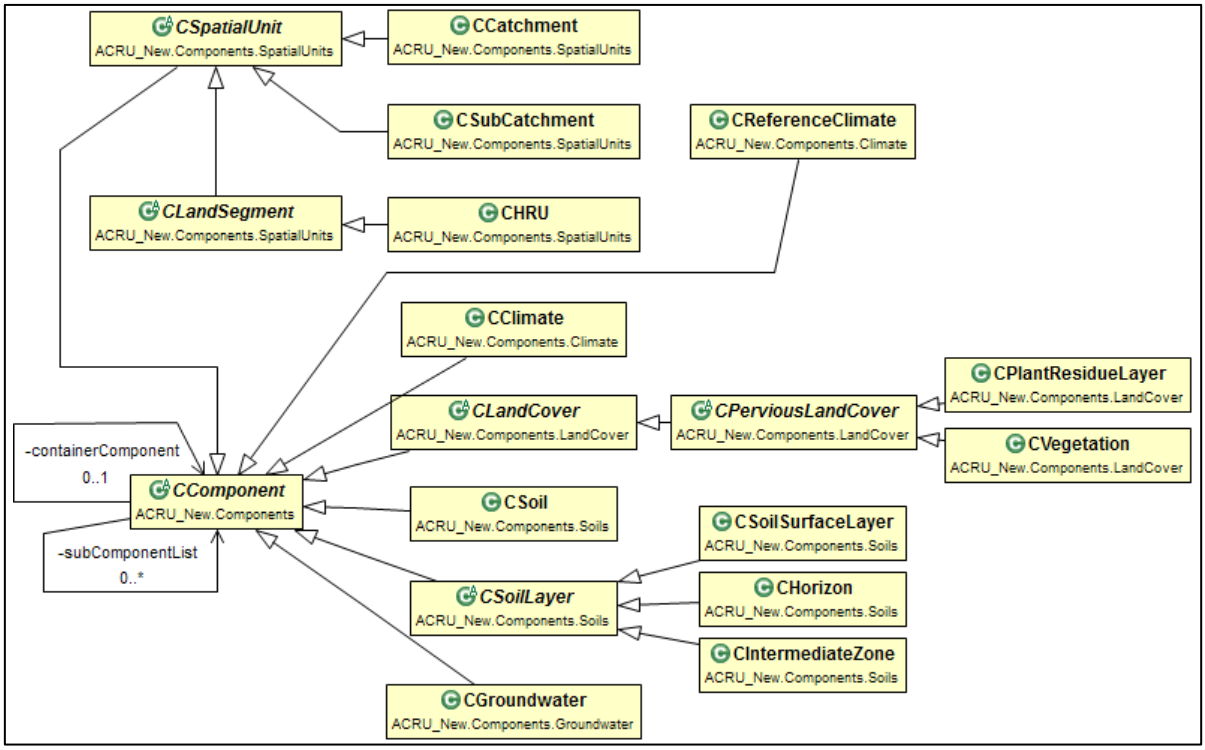

Figure 3-7 *ACRU* 5 design: *CHRU* subcomponent Component classes

The land cover on an HRU would typically be some sort of vegetation, represented by the generic *CVegetation* class and, in addition, there may be a plant residue layer on the soil surface, represented by the *CPlantResidue* class. In instances where it is necessary to be able to distinguish between different vegetation types, a Data object can be used within instances of *CVegetation* to identify these different vegetation types. Typically in *ACRU* each instance of *CHRU* would be associated with only one instance of *CVegetation*, representing one vegetation type, however, in the *ACRU-Veld* module (Kiker and Clark, 2001a) each instance of *CHRU* may be associated with more than one instance of *CVegetation*.

An instance of *CSoil* could be composed of one or more soil layers, where the abstract *CSoilLayer* class has subclasses representing more specific types of soil layer including: (i)

CSoilSurfaceLayer representing a very thin layer of soil near the surface of a land unit, (ii) *CHorizon* representing conceptually discrete soil layers with different hydrological properties, and (iii) *CIntermediateZone* representing a soil layer between the soil horizons and the groundwater store. Typically in the *ACRU* model just a topsoil horizon and a subsoil horizon are represented. The *CSoilSurfaceLayer* class was introduced for use in the new *ACRU-NP* nitrogen and phosphorus modelling module (Campbell *et al.*, 2001) where the modelling of chemical reactions and nutrient transport near the surface of a land unit are important.

3.2.2.4 Data classes

The Data classes in the *ACRU* 2000 version were developed to duplicate, to a large extent, the data structure used in the *ACRU* 3 version. However, further development of the model highlighted many shortcomings of the structure of the *ACRU* 2000 version's Data classes and the handling of data, including:

- the creation of a Data class for every parameter and variable in the *ACRU* model,
- only daily and month-of-year time series were possible,
- the storage of values for state variables was not explicitly provided for, and
- there was no flexibility for parameters and variables to be temporally dynamic.

In the *ACRU* 2000 version the *DData* class and it's more immediate subclasses provided most of the data handling functionality and the subclasses of these classes, representing specific *ACRU* model parameters and variables, provided very little if any additional functionality. The intention was that: (i) the class names of these specific Data subclasses would be used to identify parameters and variables in the model code, and (ii) that these classes would provide variable specific range checking, though this was not extensively implemented in the *ACRU* 2000 version. However, the creation of these numerous subclasses to represent new model parameters and variables was onerous and, in most cases, pointless as no additional functionality was provided.

For the above reasons the design of the Data classes was extensively restructured for the *ACRU* 5 version, as shown in Figure 3-8 and detailed in Clark and Smithers (2013). The new Data class structure was designed to provide a hierarchy of powerful and flexible non-abstract Data classes to which parameter or variable identities, data ranges and other attributes can be assigned when they are instantiated as objects during model setup. Parameters and variables are now identified using the *id*, *name* and *alias* attributes of the new *DData* class, instead of by the Data class name. The development of the new XML-based model input and configuration files, described in Section 3.3, was an important part of

implementing this new data class structure. This resulted in the elimination of several hundred Data classes from the *ACRU* model and made the model more extensible as it is less onerous to add new parameters and variables.

Each Data object is associated with a Data container, such as *CComponent* or *MModel*, which both implement the *IDDataContainer* interface. In addition to the identification attributes the *DData* class includes several attributes that are used to describe the model parameter or variable being represented. For example, the *maximumValue* and *minimumValue* attributes enable simple range checking for variables containing numerical values. The *parameterType* attribute specifies whether a model parameter or variable is: (i) a model input, (ii) a model output, or (iii) a state variable which is inherently both a model input and a model output variable. Better provision has been made for different data structures and data value types in the *ACRU* 5 version of the Data classes, which are described in more detail in Appendix 8.4.3. The *valueType* attribute specifies the data value type, for example: string, integer or double precision floating point value types. There are attributes describing the data structure, for example: a single value or a collection of values, where a collection of values could be an array or a lookup table.

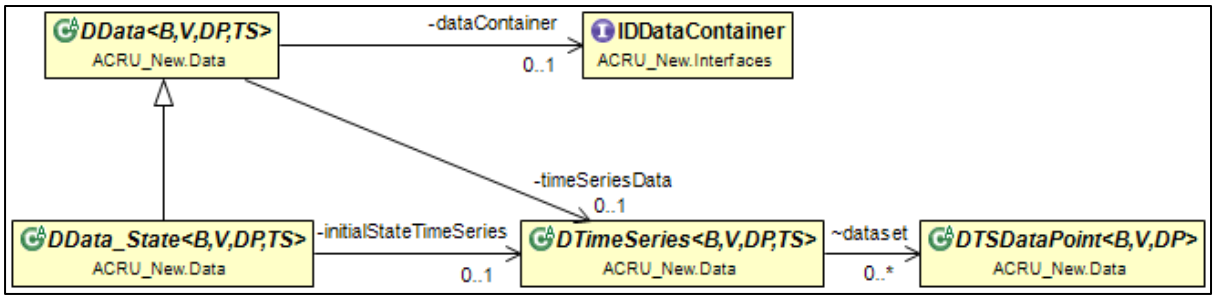

Figure 3-8 *ACRU* 5 design: main Data classes

The abstract *DData_State* class extends the *DData* class and acts as the superclass for all Data classes that store state type data, that is, data that persists from one model timestep to the next. Examples of state variables are variables representing water stored in a dam or within a soil layer. Previously the *ACRU* model required a suitable warm up period at the start of the time period being simulated so that certain state variables, such as the groundwater store, had time to stabilise. Initialising these state variables would be useful when using *ACRU* for planning purposes and essential if *ACRU* is to be used in future for assessing water operations scenarios using forecast climate data and then updating this with near real-time data as it becomes available. This initialisation of state variables to a specified point in time is termed “hot-starting”. The new model input file structure, together with the

DData_State class and improved handling of time series data within the model, especially breakpoint time series, has made it possible to hot-start the model.

The new Data structure of the *ACRU* 5 version also provides better handling of time series data and more flexibility with regard to the types of time series that can be represented. Each instance of *DData* may store either a constant value (time invariant in the context of the modelled scenario) or a time series of values. The *DTimeSeries* class was developed to provide functionality for handling time series data. Each instance of *DTimeSeries* contains one or more data points, where each data point is an instance of the *DTSDataPoint* class. A *DTSDataPoint* object stores the following: (i) the date-time of the data point, (ii) a data value, and (iii) an optional data quality flag. The details of these classes are described in more detail in Appendix 8.4.3.

Often there are attributes of a hydrological system that are typically assumed to be constant for the duration of a modelled scenario, for example, land cover. However, in some instances it may be important to model intermittent changes in such attributes. The *ACRU* 3 version made provision for modelling certain model variables dynamically through the use of a dynamic input file specifying breakpoint values, but this feature was not included in the *ACRU* 2000 version. One disadvantage of the dynamic input files in the *ACRU* 3 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 file structure and better handling of time series data within the model, especially breakpoint time series, has made provision for these dynamic variables to be represented in the model input files and internally in the model, as either constant values or as a time series of values, in the same manner as all other variables.

In the *ACRU* 5 version extensive use has been made of generic typing in the Data classes. Generic typing, which was not available in Java when the *ACRU* 2000 version was created, has helped simplify and improve the type safety of the Data classes. The generic types are described in Appendix 8.4.1.

3.2.2.5 Resource classes

Conceptually the *ACRU* model's Component objects represent the physical components of a water resource system and Data objects represent the attributes of the physical components. Matter such as water could potentially be represented in *ACRU* either: (i) as a physical entity using Component objects within container Component objects, or (ii) as a

Component attribute using Data objects as was done with *DFluxRecord* in the *ACRU* 2000 version. In the *ACRU* 5 version water, sediment and nutrients are collectively referred to as "resources" and a separate set of Resource classes have been added to the model. These *Resource* classes are conceptually similar to Component classes in that they represent a physical entity with Data attributes, but contain specialised functionality to record resource storage, influxes, outfluxes and ownership. The main Resource classes and their relationships to *CComponent* and *DData* classes are shown in Figure 3-9.

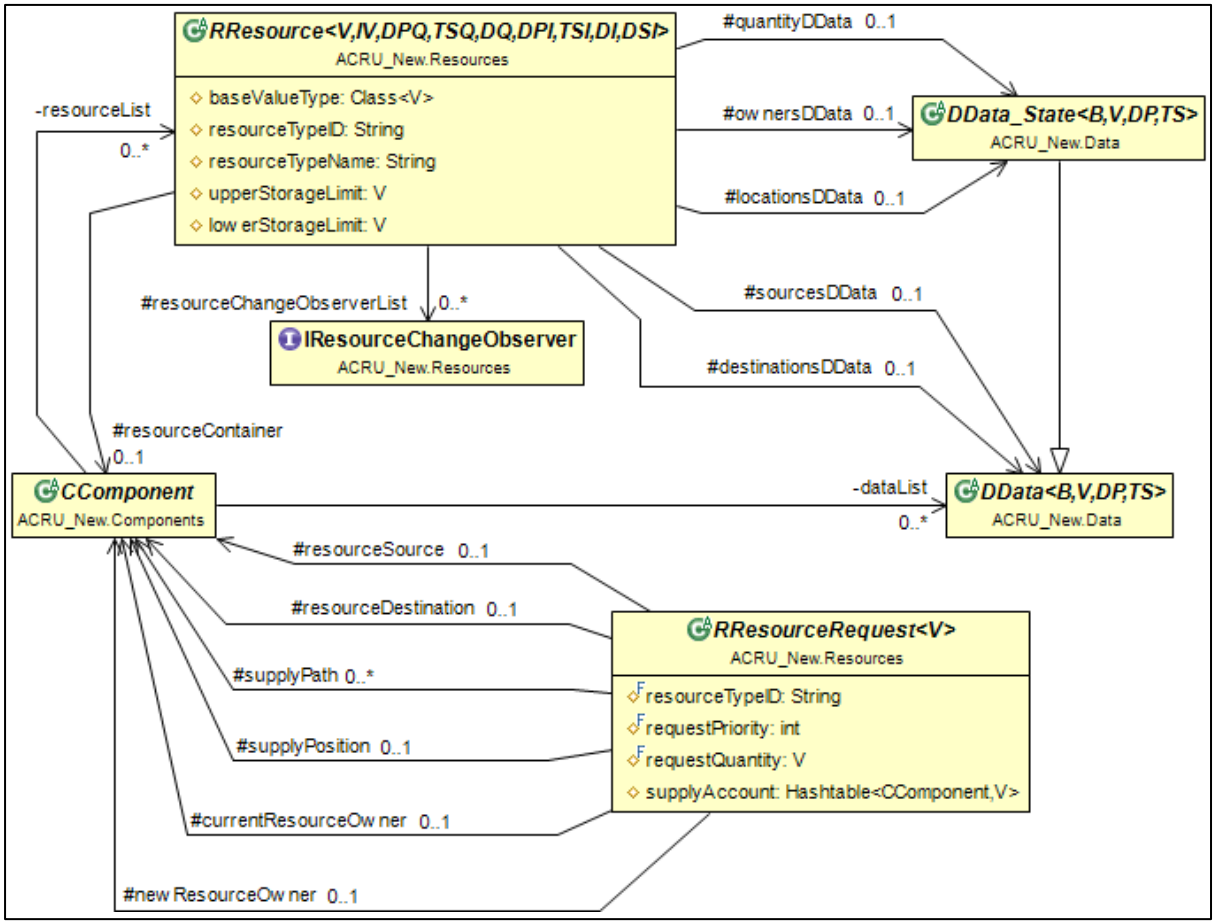

Figure 3-9 *ACRU* 5 design: main Resource classes

The *RResource* class has a *resourceTypeID* attribute that is used to identify the type of resource, for example "*WATER*", "*SEDIMENT*" or "*NITRATE*". Each instance of *RResource* belongs to an instance of *CComponent*, which is the container of the resource. Each instance of RResource is associated with five Data objects:

- an instance of *DData_State* which stores the resource quantity that exists within the container Component at any point in time during the simulation period,
- an instance of *DData* which stores the individual influxes of the resource to the container Component during a timestep and the source Components of these influxes,

- an instance of *DData* which stores the individual outfluxes of the resource from the container Component during a timestep and the destination Components of these outfluxes,
- an instance of *DData_State* which stores the ownership of portions of the resource within the container Component by owner Components, and
- an instance of *DData_State* which stores the quantities of the same resource type that the container component has ownership of, but which is stored in other instances of CComponent.

The *IResourceChangeObserver* is an interface that can be implemented by Process classes to initiate an action when the quantity of a specified resource type in a specified resource container Component object changes. The *RResourceRequest* class is used in conjunction with the *RResource* class to enable Component objects to order quantities of a resource from another Component object. For example, an instance of *CIrrigatedArea* may send a request for a specified quantity of water to an instance of *CDam*, and the requested quantity may be supplied from unallocated water in the dam or from water in the dam that has previously been allocated to a specified owner Component object. A *RResourceRequest* class contains several attributes to store the following information:

- the type of the resource being requested, for example "*WATER*";
- the quantity of resource requested (the amount actually supplied may be less than the requested amount, depending on availability at the source);
- the priority of the request (a request with *requestPriority*=1 has the highest priority, followed by a request with *requestPriority*=2, and requests with the same priority share resources equally);
- the source Component object which will provide the requested resource quantity if possible;
- the destination Component object to which the requested resource quantity will be supplied;
- a list of Component objects specifying the supply path through which the requested resource will move from source to destination;
- the Component object within the supply path through which the requested resource is currently travelling;
- the Component object which owns the requested resource at the source Component object; and
- the Component object to which ownership of the requested resource will be transferred when supply commences.

These new Resource classes are an improvement over the previous *ACRU* 2000 system of *DFluxRecord* classes as they: (i) are conceptually more intuitive, (ii) enable the state of resources to be more easily persisted in the XML-based model input files as the existing functionality for *DData_State* objects is used, and (iii) are more extensible as new resources can be represented by simply specifying a new value in the *resourceTypeID*, instead of having to create a new subclass for each resource type. Resources consisting of discrete entities such as livestock could also be modelled.

3.2.2.6 Process classes

The *PProcess* class, shown in Figure 3-10, has two important methods: (i) the *initialise* method and (ii) the *runProcess* method. The *initialise* method is empty in *PProcess*, but may be overridden as required in subclasses. The *initialise* method is called for every instance of PProcess, after model initialisation and just before the simulation starts, to perform any Process initialisation that may be necessary. The abstract *runProcess* method must be implemented in subclasses. The *runProcess* method is called for every instance of *PProcess*, for every simulation timestep, to execute the Process specific algorithms. The Process classes in the *ACRU* 5 version required some minor changes to implement the use of the new Data and Resource classes. Two additional, and more significant, changes that were made to the Process classes were: (i) an improved specification of the Data objects required by each Process in the *setRequiredData* method, and (ii) the addition of a similar specification of the Resource objects required within a new *setRequiredResources* method. As shown in Figure 3-10, two classes were created for this purpose, *PProcessDataItem* and *PProcessResourceItem*. The *PProcessDataItem* class stores: (i) a reference to the Data container (an instance of *MModel* or *CComponent*), (ii) the Data ID, and (iii) whether the Data object is an input parameter or variable in the context of the associated process, or an output, or both as is often the case with state variables. The *PProcessResourceItem* class stores: (i) a reference to the Component object containing the Resource object, (ii) the resource type ID and (iii) whether the Resource object is an input, an output or both. The *setRequiredData* and *setRequiredResources* methods are declared in the *PProcess* class and overridden in individual subclasses, which enables these methods to be called from the *AAcruXmlModelCreator* class to determine the data requirements of each Process object that forms part of the model.

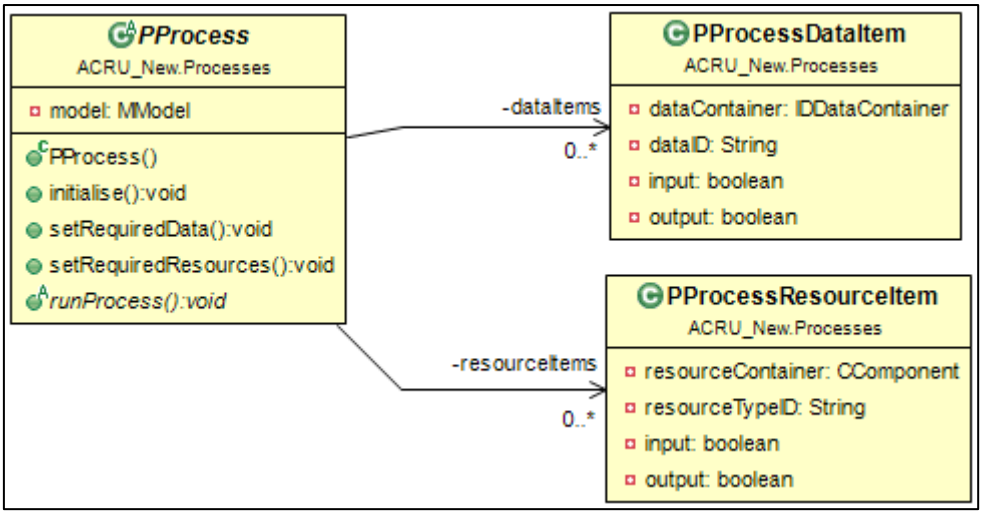

Figure 3-10 *ACRU* 5 design: main Process classes

All Process classes that represent the flow of water, including rainfall and evaporation, implement the *IWaterFlow* interface and thus have a *flowWater* method. Typically the *runProcess* method will call the *flowWater* method, which contains the main process algorithm, though in some cases the *flowWater* method may in turn call one or more other methods. The main abstract subclasses of the *PProcess* class, which relate to the flow of water, are shown in Figure 3-11.

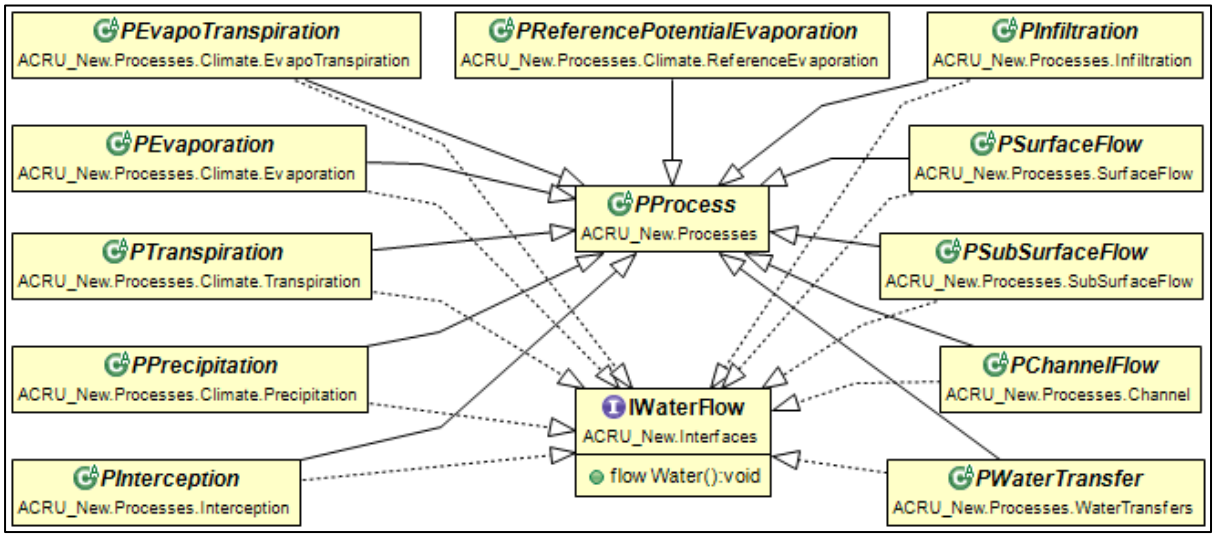

Figure 3-11 *ACRU* 5 design: *PProcess* class and main abstract subclasses related to the flow of water

An example of how Process, Component and Data classes are related is shown in Figure 3-12. A Component object may have an ordered list of associated Process objects. The *PSatDownwardFlow* class models the saturated downward movement of water between soil

layers. The *PSatDownwardFlow* class implements the *IWaterFlow* interface and the *flowWater* method is called by the *runProcess* method. The *PSatDownwardFlow* class acts on all the *CSoilLayer* objects that form part of a *CSoil* object and the *CGroundwater* object below the *CSoil* object. The *flowWater* method retrieves input values from each *CSoilLayer* object for: (i) Data attributes such as the depth, drained upper limit, wilting point, response factor, and (ii) Resource attributes such as current soil moisture storage. The method then calculates whether there is any movement of water, and if so, adjusts the soil moisture storage in the *RResource* (WATER) objects in both the source and destination *CSoilLayer* objects.

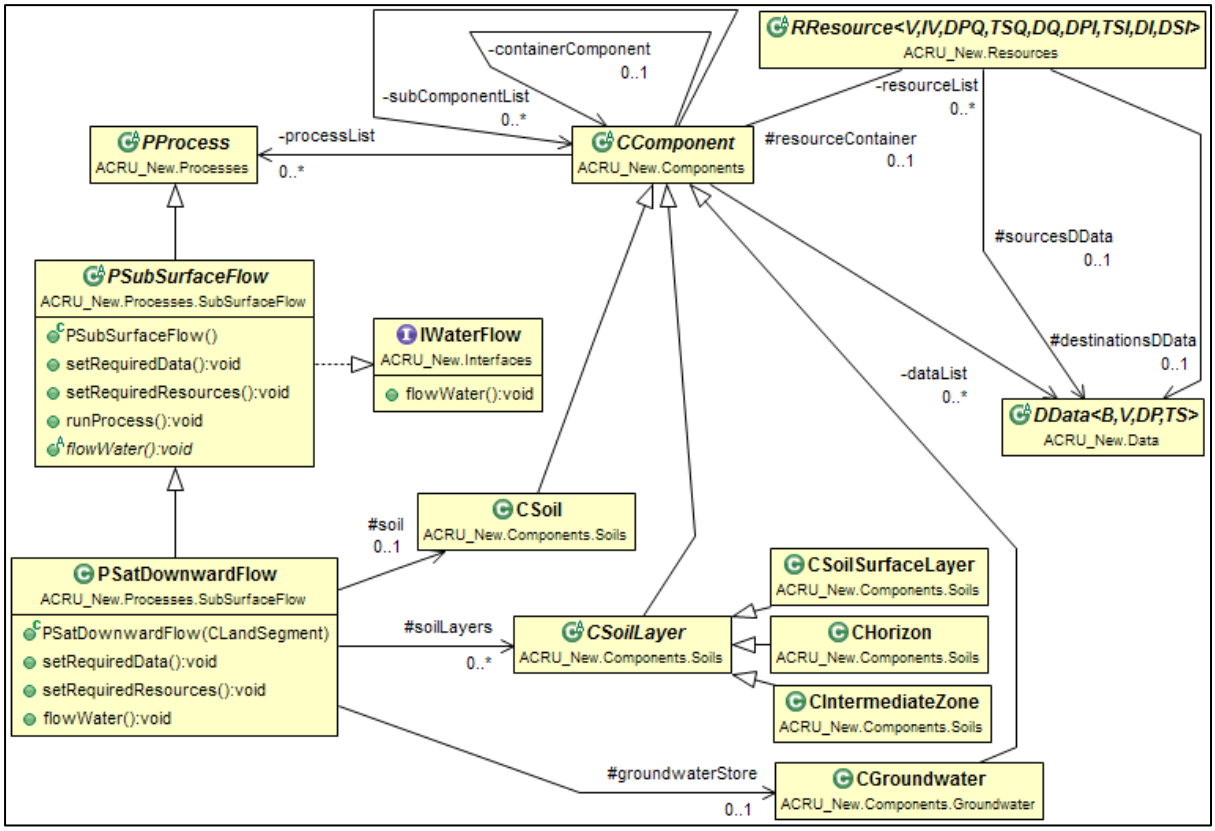

Figure 3-12 *ACRU* 5 design: example of Component, Data, Process class relationships

During model initialisation the typical sequence of events is as follows: (i) Component objects are created, (ii) Process objects are created, and the *setRequiredData* and *setRequiredResources* methods are called for each process, (iii) the required Data objects are created and, (iv) the *initialise* method is called for each Process object. During the model simulation phase the typical sequence of events is as follows: (i) for each Component object in the *computationOrder* list, for each Process object in a Component object's *processes* list, the Process object's *runProcess* method is called, which may in turn call other methods within the Process object or associated Process objects, (ii) timestep data for

model output variables is saved to the model output files, (iii) the simulation advances to the next timestep.

In real-world hydrological systems different processes occur simultaneously throughout a spatial and temporal continuum. In *ACRU*, hydrological systems are reducted to discrete Component and Process objects which are assigned to a computation order, first by component and then by process. Mostly this simplification is acceptable, though while restructuring *ACRU* it was found that there were two situations where this simplification created difficulties for modelling. The first situation is where there are feedbacks between different Processes in different Components. An example of this occurs when calculating an irrigation requirement for an irrigated area Component, then calculating irrigation supply at a dam Component, then finally irrigation application on the irrigated area Component for which computation for the simulation timestep has already been complete. The second situation is where there are feedbacks between different Processes for different Resources. This occurs when, for example, water flows and nutrient flows are calculated in different Processes, but the nutrient flow depends on the water flow in a specific water flow Process. Solutions to both examples were created by adding specialised code to the *ACRU* model. An alternative solution, recommended for further investigation, would be to develop an algorithm to determine the computation order of Process objects belonging to different components, based on data requirements, as a more flexible alternative to the hardcoded Process ordering rules.

3.3 Design and Development of an XML *ACRU* Model Input Structure

The *ACRU* 3 and *ACRU* 2000 versions of the model use relatively simple text-based model input files to store: (i) a few global modelling options, (ii) the model input parameter and variable values for each subcatchment, and (iii) the flow network connectivity between subcatchments. For the *ACRU* 3 version the main model input file was a single text file. For the *ACRU* 2000 version there was a single "control" file containing global options, and references to a set of subcatchment input files, with one file for each subcatchment, containing model input parameter and variable values. Both versions used additional separate text-based files to store daily time series information. Some of the disadvantages of these model input files for use with the restructured object-oriented design of the *ACRU* model, especially the *ACRU* 5 version, were that the flat text structure:

- was difficult to map to *ACRU* 5 Components and Data objects,

- was suitable for the simpler subcatchment view (Figure 2-2) of the *ACRU* 3 version, but did not allow for the more flexible and complicated subcatchment internal configurations possible with the *ACRU* 5 version, and
- included values for all parameters and variables for each subcatchment even if not required.

As discussed in Section 2.2, subsequent to the initial object-oriented restructuring of the *ACRU* model it was recognised that a new model input file data structure would be required to complement the object-oriented model structure, enabling the full potential of the new object-oriented model structure to be used. XML was identified as a suitable way to structure these files. XML files are text-based, platform independent files, that enable model input information and data to be structured in a manner that reflects the structure of the model. XML files are extensible and can be easily serialised into memory to populate the model. Another advantage of XML files is that their structure may be declared in, and checked against, an XML Schema file which acts as a form of XML file template. The initial designs of these XML-based model input files for *ACRU* were first described in Clark *et al.* (2009) and Clark *et al.* (2012b), and some refinements related to the *ACRU* 5 version of the model were described in Clark (2013). The design of the XML-based model input file structure sought to address four main requirements:

- 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, the *ACRU* model engine or to the software utilities that read from or write to the data model;
- store actual data values or references to where data values are stored, such as in a separate database; and
- enable storage of additional information that describes the model parameters or variables for use in graphical user interfaces (GUI) and other software tools.

Early in the design process it was recognised that two different XML-based files were required: (i) one file to store specific Component configuration information and Data values for a study catchment, and (ii) one file to store general metadata type information about the Component, Resource and Data types. The structure of the first of these files is specified in an XML schema referred to as the *ModelData* schema, which is described briefly in Section 3.3.1. All the XML-based model input files used with the *ACRU* 5 version must conform to

this *ModelData* schema. The structure of the second type of file is specified in the *ModelConfiguration* XML schema, which is described briefly in Section 3.3.2.

Although the XML-based model input files are readable and editable in a simple text editor, they are less readable than the older *ACRU* flat text file formats and thus a software library named the *XmlModelFiles* library, described in Appendix 8.5.3, was developed to assist in developing software tools to create and edit model input files. In addition a software library known as the *ModelDataAccess* library, described in Appendix 8.5.4, was developed to provide a uniform set of data readers and writers for data, especially time series data, stored outside, but referenced by, the model input files. The Configuration Editor tool, described by Clark *et al.* (2009) and Clark *et al.* (2012b), was developed to provide model users with a graphical user interface enabling them to visually create and edit *ACRU* model input files. However, the development of the Configuration Editor tool was not part of this study. The various file types and software tools related to the *ACRU* 5 version of the model are shown in Figure 8-14 in Appendix 8.5.

Although the *ModelData* schema, *ModelConfiguration* schema, *XmlModelFiles* library, *ModelDataAccess* library and Configuration Editor were designed and developed primarily for the *ACRU* model, they were designed in such a way that they could easily be applied to other models based on the same design. An advantage of using XML files is that they are programming language and platform independent.

3.3.1 *ModelData* Schema

The *ModelData* schema provides a data model describing the structure of an XML-based model input file that complements the object-oriented design of the *ACRU* model. 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. A different implementation of the *ModelData* schema would be used for each configuration of the *ACRU* model, that is, one XML model data file for each study catchment. The *ModelData* schema must be used in conjunction with the *ModelConfiguration* schema (Section 3.3.2). The root element of the *ModelData* schema, *Model*, and its main sub-elements are shown in a simplified schema diagram in Figure 3-13. There are many similarities between the main elements in the *ModelData* schema and the main *ACRU* 5 classes shown in Figure 3-2. A *Model* element may contain several *Component* elements, each of which may contain zero or more child *Component* elements. Each *Component* element contains a list *Data* elements containing modelling parameter and variable values describing the *ACRU* Component. The

ModelInfo element contains a list *Data* elements containing general model parameter and variable values. The *Model* element also contains a list of *Relationship* elements describing relationships between *ACRU* Components. The *ModelData* schema is described in more detail in Appendix 8.5.1

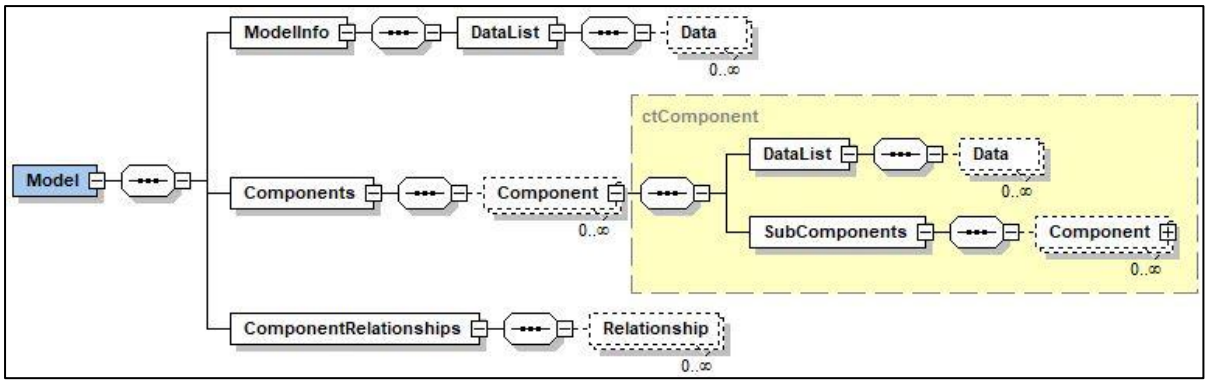

Figure 3-13 The *Model* element and main sub-elements of the *ModelData* schema

3.3.2 *ModelConfiguration* Schema

The *ModelConfiguration* schema complements the *ModelData* schema by providing a data model for storing information describing permitted component configurations and relationships and also metadata type information about model parameters and variables for use in the *ACRU* model and associated software utilities such as the Configuration Editor. A single implementation of the *ModelConfiguration* schema would be used for all configurations of the *ACRU* model. An implementation 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. The root element of the *ModelConfiguration* schema, *ModelConfiguration*, and its sub-elements are shown in Figure 3-14. The *ModelInfo* element may contain several *DataDef* elements containing metadata describing general model parameters and variables. The *ModelConfiguration* element may contain one or more *ComponentType* elements representing the different types of physical components making up the hydrological system being modelled (e.g. subcatchments, rivers, dams, vegetation or soil horizons). Each *ComponentType* elements contains: (i) a list *DataDef* elements containing metadata describing parameters and variables, and (ii) a list of *ResourceDef* elements describing resources that might be stored in a model Component of that type. The *ModelConfiguration* element also contains a list of *ResourceType* elements describing the types of resources that can be modelled. A list of *RelationshipType* elements describes the relationship types that may be configured between different types of Components. The *ComponentConfiguration* element contains information about which types

of Components may be configured as subcomponents of other types of Components and the relationships that can or must exist between different types of Component. The *ModelConfiguration* schema is described in more detail in Appendix 8.5.2.

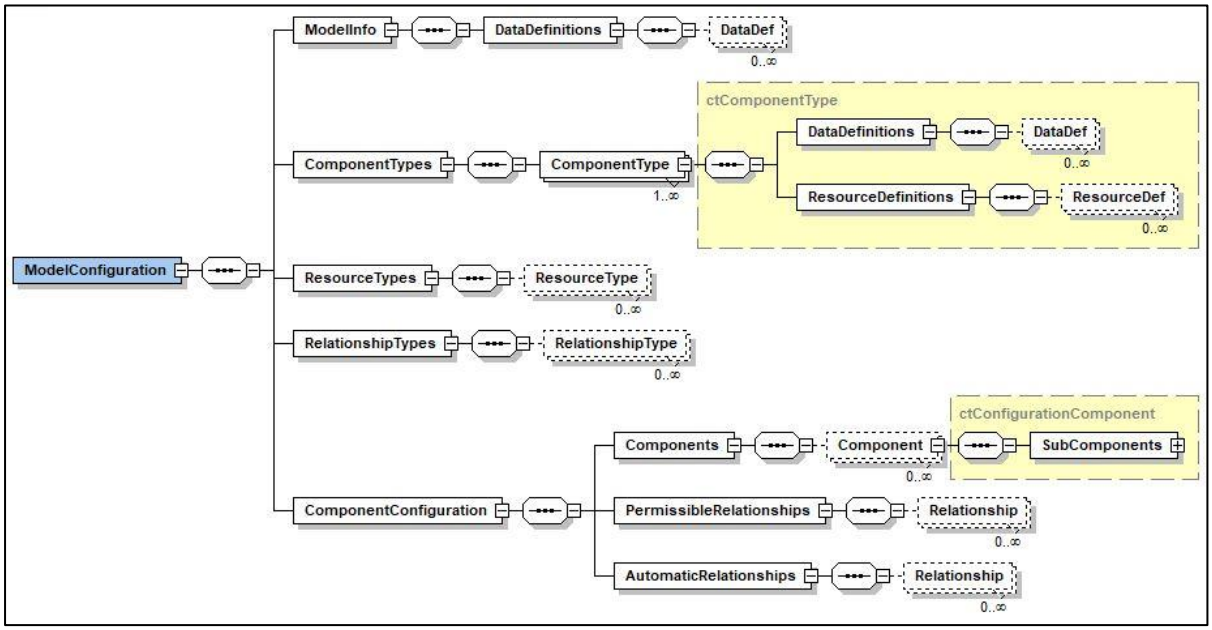

Figure 3-14 The *ModelConfiguration* element and main sub-elements of the *ModelConfiguration* schema

3.4 Development of a Water Accounting Module for *ACRU*

Clark (2015b) describes the development of a water use quantification and accounting methodology for South Africa using a hydrological modelling approach, as summarised in Appendix 8.8. As part of the methodology a new *Accounting* module was developed for the *ACRU* 5 version of the model to create catchment-scale water resource accounts. The purpose of water accounting is to quantify and communicate water stocks, inflows, depletions and outflows for an entity, which may be a factory, an urban water reticulation system or a catchment. The term 'water resource accounting' was used by Clark (2015a), in the context of water accounting at a catchment-scale, to include the water balance and all the significant water fluxes in the hydrological cycle within a catchment. Based on a review of water accounting frameworks by Clark *et al.* (2015), the Water Accounting Plus (WA+) water accounting framework, described by Karimi *et al.* (2013), was selected by Clark (2015b) as being most suitable for catchment-scale water resource accounting. A modified version of the WA+ Resource Base Sheet was included in the new *ACRU Accounting* module and there is scope for other accounting sheets to be included in the module. The

object-oriented design of the restructured model facilitated the development of the *Accounting* module in several ways, including:

- Modelling of several different land use classes per subcatchment as HRUs enabling sectoral water use to be estimated and represented.
- Representation of inter-catchment transfers, which required parallel processing, and daily flow volumes for these transfers instead of mean month-of-year flow values.
- Representation of nested catchments using objects and information about flows between catchments enabled easy aggregation of catchment accounts from sub-Quaternary Catchment level to Primary Catchment level.

3.5 Discussion

The algorithms representing hydrological processes in *ACRU* were developed based on research conducted over many years, primarily in South Africa, making the model a valuable repository of knowledge. However, the physical conceptual nature of *ACRU* has enabled it to also be applied in other countries and for a wide range of different purposes. The restructuring of the model sought to build on this strong foundation by applying object-oriented design techniques to make it easier to add new functionality to the model and to enable more flexible configuration of the model. The restructuring was more than a simple translation from one computer language to another; the object-oriented approach resulted in a different way of conceptually representing the water resource systems being modelled. All of the design objectives for the restructured *ACRU* 5 version of the model, listed in Section 3.2, have been achieved as a result of the object-oriented restructuring and the unique and innovative design. The first three objectives of the research study, (i) to (iii) listed in Section 1.4, have been achieved in the restructured *ACRU* model and are discussed in more detail in the following sections, followed by a section with more general reflection regarding the outcome of the restructuring.

3.5.1 Object-Oriented Restructuring for Greater Flexibility

The first objective for restructuring the model was greater model flexibility to enable more flexible configuration of catchments within the model to better represent real-world complexity, including multiple land cover/use types and the interconnectivity between components of the modelled water resources system. This increased flexibility was achieved through: (i) defining the physical hydrological components more explicitly as objects, (ii) use of the containment (“part of”) mechanism of object-oriented programming, where objects can consist of one or more sub-component objects, (iii) the association

mechanism of object-oriented programming which enable relationships between objects to be specified, (iv) the more modular representation of the hydrological processes as classes, enabling different representations of the same process to be more easily interchanged, and (v) structuring the execution order of processes to permit parallel processing, such that components of the modelled water resources system can exchange water on a timestep-by-timestep basis. However, there is still scope for improvement in the model's internal administration of the order of Process object execution. The order in which Process objects are executed is still hard-coded, to a large extent, and a more dynamic determination of the order, using already existing internal Process class metadata stating the data variables required by a Process object, should be investigated in the future.

3.5.2 Object-Oriented Restructuring for Greater Extensibility

The second objective for restructuring the model was to make it more extensible. The improved extensibility was achieved: (i) partly through class inheritance and polymorphism mechanisms which are key characteristics of object-oriented programming, (ii) by defining the physical hydrological components more explicitly using classes and their object instances, (iii) by defining the individual hydrological processes more explicitly using classes, and (iv) through making it easier to declare new model input variables and parameters, in the model as Data objects, in the model input files and in the model configuration file. The concept of Resource classes and objects in the restructured model has made it easier to: (i) model water quality constituents such as nutrients and sediment, (ii) ensure accurate balances of water and constituents within the model, (iii) track flows of water into and out of Component objects, which is especially useful for the model's application for water resource accounting, and (iv) track ownership or water. This extensibility has been confirmed through the addition of several new classes and modules to the *ACRU* 2000 version of the model, including: (i) new Process classes for better representation of river and dam operations (Butler, 2001), (ii) the *ACRU-Veld* module for modelling mixed vegetation land cover and utilisation by herbivores (Kiker and Clark, 2001a), (iii) the *ACRU-NP* module for nitrogen and phosphorus (Campbell *et al.*, 2001), (iv) the *ACRUSalinity* module for modelling salinity (Teweldebrhan, 2003), (v) the *ACRUCane* module for advanced sugarcane and irrigation modelling (Moult, 2005), and (vi) shallow water-table modelling of flatwood areas in Florida (Martinez *et al.*, 2008). The extensibility of the *ACRU* 5 version was demonstrated through the addition of the *Accounting* module (Clark, 2015b). These new classes and modules also demonstrated the flexibility of the object-oriented structure.

3.5.3 Object-Oriented Restructuring for Improved Data Handling

The third objective for restructuring the model was to provide improved data handling. The internal data structure of the restructured model provides more flexible handling of time series data. *ACRU* is a daily timestep model and many of the algorithms representing hydrological processes are designed to be run at a daily timestep. The more flexible handling of time series data makes it easier to use the best available data and either aggregate or disaggregate it to daily values. This flexibility, especially for breakpoint time series, also enables dynamic changes (not at a regular daily timestep), such as vegetation characteristics to be represented more easily than with the dynamic input files of the *ACRU* 3 version. The inclusion of metadata about model data variables and parameters enables range checking and translation between different units of measure. The provisions within the new data structure to more explicitly deal with state variables will make the model more suitable for use in an operational context, through hot-starting, by enabling the model to be initialised to a measured or previously simulated state. The inclusion of the concept of scenarios has made it possible to include different data values in a single model input file for different modelling scenarios.

The base classes of the *ModelDataAccess* library, together with the model's new internal data structure, have made it easier to include different data formats for model input and output data. This makes it easier to use data from different data sources, facilitates linking of *ACRU* to other models and has enabled *ACRU* to be included as a model in two integrated modelling frameworks: (i) the SPATSIM Hydrological Decision Support Framework (SPATSIM-HDSF), as described in Clark *et al.* (2009) and Clark *et al.* (2012a), and (ii) the Delft Flood Early Warning System (Delft-FEWS), as described in Appendix 8.5.5.

The new XML-based model input file structure complements the restructured object-oriented model. The XML model input files are not object-oriented, as they do not include key concepts such as inheritance and polymorphism, but they are structured to describe the containment and association relationships between the water resource components to be represented in the model. Without this complementary model input file structure it would be difficult to use the object-oriented structure of the model to its full potential. The Configuration Editor, which was not developed as part of this study, uses a graphical tree structure to represent the containment structure of the water resource components. Based on informal feedback from users this graphical representation of the containment structure of the water resource components has made the configuration of catchments and their subcomponents more intuitive.

3.5.4 Reflections

There are a few negative outcomes to the restructuring that need to be acknowledged. Informal benchmarking of the execution time of the *ACRU* 5 version against the *ACRU* 3 version, has shown that the execution time of the restructured model is slower. In part, the poorer execution speed may be due to Java being an interpreted language. There are also performance trade-offs for parallel processing as it is necessary to have a larger quantity of data loaded into memory at one time, the whole case study area compared to just one subcatchment at a time in the *ACRU* 3 version. There is scope for optimisation of the source code of the restructured model. The XML-based model input files are typically larger in size than the simpler text files used for the *ACRU* 3 version. The bigger file size is due to the XML tags used to structure the data in the XML files. However, there are some trade-offs as the model input files used for the *ACRU* 3 version can in some cases include a large number of redundant variable and parameter values for model variables that are not required for a particular configuration of the model. The XML files are also not as simple to edit using a text editor which has resulted in some frustration to experienced users of the model. However, forcing inexperienced users to use the Configuration Editor is useful as it does useful validation of model input and thus the time spent providing support to users who have made unintended errors in configuring the model is reduced. The change from a procedural FORTRAN 77/90 programming paradigm to object-oriented programming in Java has also, unfortunately, reduced the number of researchers actively involved in *ACRU* code development.

Despite the restructuring of the model and input files, most users of the model are unlikely to notice any real difference from the *ACRU* 3 version as the algorithms representing hydrological processes are the same. However, more experienced model users will be able to exploit the greater flexibility with which water resource systems can be represented in the model. For model developers, it is intended that the better conceptual foundation for the model will make the model code easier to understand and extend. The design of the *ACRU* 5 version of the model and its input files are likely to be stable from this point on and no substantial changes to the design are anticipated. The restructured *ACRU* model and XML model input files, together with the associated libraries and other tools are not an end in themselves, as the design and the base Component, Process, Data and Resource classes could be used to develop simpler sub-models of *ACRU* or indeed completely new models.

4 MODEL INTEGRATION

The fourth objective of this research study, as stated in Section 1.4, was to develop a means of linking *ACRU* with other models to facilitate integrated modelling studies. Integrated water resources management (IWRM) recognises that there are not only water quantity and quality aspects of water resources management but also ecological, economic, political, social and institutional aspects to water resources management. IWRM requires integrated assessment of complex water resource systems, which in turn requires the application of detailed process-oriented models (Barthel *et al.*, 2006). It is unlikely that a single model will be able to adequately represent all facets 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). Thus, the solution of real-world problems through modelling most often requires integrated analyses, which in turn requires linking of models (Kokkonen *et al.*, 2003). Kralisch *et al.* (2005) state that sustainable water resource management requires integrated, flexible hydrological models to simulate both water quantity and quality aspects of the hydrological cycle with a suitable degree of certainty. Existing models are often developed for, or have strengths, in specific domains within the water resource system and integration of models is a popular solution in attempting to model complexity (Moore and Tindall, 2005; Barthel *et al.*, 2006). However, many existing models were developed for specific scales and purposes, are often coded in different programming languages, run on different operating systems, and may not be easily adapted for integration with other models (Hoheisel, 2002; Kralisch *et al.*, 2005).

When linking models, one of the main challenges is to understand and define the dependencies between models (Hoheisel, 2002). There are also both technical and conceptual constraints to be overcome when attempting to integrate models from different disciplines. Technically, existing environmental models are not generally designed to communicate with other models within the same discipline, let alone communicate with models from other disciplines, and conceptually, different models are often based on different ontologies due to different disciplines having different views of the natural environment, and differences in the way concepts are expressed in computer code (Argent, 2004). 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 the individual models and the integrated collection of models are necessary (Barthel *et al.*, 2006).

As described in Clark and Smithers (2013) it is necessary to develop and integrate modelling tools to support water resources planning and operations decisions by water resource managers in South Africa. The project undertaken by Clark and Smithers (2013) specifically sought to enhance the capabilities of the *ACRU* agrohydrological model by linking it to a river network model. A review of model linkage systems by Clark *et al.* (2013), summarised in Section 4.2, led to the selection of the OpenMI model linking interface. A review and evaluation of river network models by Thornton-Dibb *et al.* (2013) led to the selection of the MIKE BASIN model, largely due to its ease of use, strong GIS support through ArcGIS and availability of local user support and training. None of the reviewed models were found to be OpenMI compliant at that time. For the *ACRU* model, OpenMI Version 1.4 (OpenMI 1.4) linkable components were created for both Java and .Net programming languages (Clark and Lutchminarain, 2013). For MIKE BASIN, linkable components were created for both OpenMI 1.4 and OpenMI Version 2.0 (OpenMI 2.0) for .Net (Clark and Lutchminarain, 2013). The *ACRU* and MIKE BASIN models were configured and linked using OpenMI for the Kaap River Catchment in Mpumalanga, South Africa, demonstrating several use cases for linking between the models.

The eWater Source model is described by Carr and Podger (2012) as the new Australian National Hydrological Modelling Platform. It is an integrated modelling system developed by the eWater Cooperative Research Centre (CRC) for the simulation of river flow networks, including water quality and environmental flows. A review by Thornton-Dibb *et al.* (2013) of a beta version of the eWater Source as a river network model indicated that it met the requirements for linking to the *ACRU* model (Clark and Smithers, 2013), but it was not evaluated further as it was still under development. This PhD study presented an opportunity to evaluate eWater Source further as potential modelling tool for use in South Africa, and to simultaneously demonstrate the new ability to link the *ACRU* model to models representing other domains to better represent reality.

This chapter on model integration includes the following:

- a discussion of different approaches to model integration,
- a summarised review of model linkage systems which led to the selection of OpenMI,
- an overview of the OpenMI interface standard,
- a description of the development of an OpenMI 2.0 linkable component for *ACRU*,

- a description of the development of an OpenMI 2.0 linkable component for eWater Source.
- a discussion related to differences between models and OpenMI in the way time is represented, and
- a description of how the *ACRU* and eWater Source models were linked using OpenMI.

4.1 Approaches to Model Integration

Some typical approaches to model integration are shown in Figure 4-1 and explained in this section. Integration of models and modelling approaches for IWRM have led to the development of integrated modelling environments, model interfacing specifications and modular modelling systems (Krause *et al.*, 2005). Gregersen *et al.* (2007) noted that some existing hydrological decision support systems were 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: (i) direct integration of whole models through implementation of a model interface specification, and (ii) modular modelling systems where modules representing individual processes are combined to create custom models, with both approaches having advantages and disadvantages.

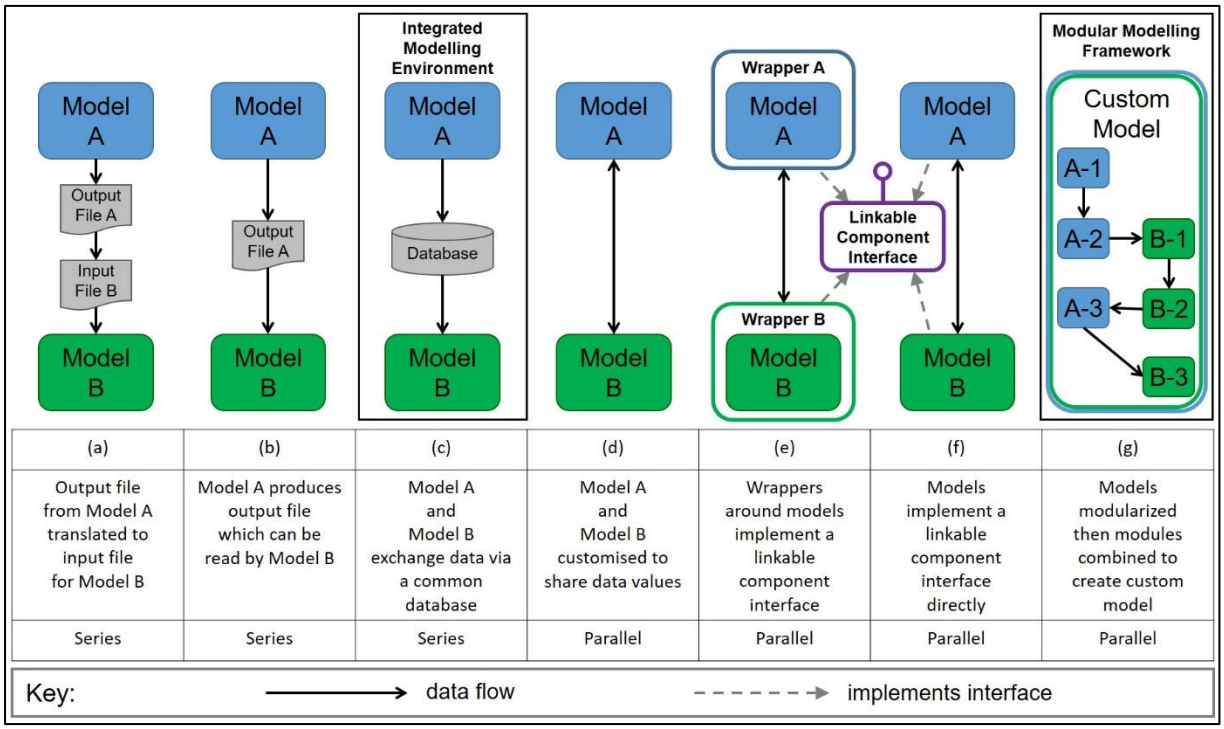

Figure 4-1 Approaches to model integration

4.1.1 Simple Model Integration

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 (Figure 4-1a), where 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 year period. 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 model. 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: (i) the effort required to reformat the output from one model to be suitable for use as input for the other model, and (ii) 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 (Figure 4-1b).

4.1.2 Model Integration Using Modelling Environments

IWRM has led to the development of integrated water resource modelling environments (Figure 4-1c) or decision support systems such as LIANA (Hofman, 2005), SPATSIM-HDSF (Clark *et al.*, 2009), DeltaShell (Donchyts and Jagers, 2010), Delft-FEWS (Werner *et al.*, 2013) and GeoJModelBuilder (Zhang *et al.*, 2017). Integrated modelling environments typically include common data storage and formats, common data editing tools, and common spatial and temporal data visualisation and analysis tools. These modelling environments provide a modelling environment within which model users can operate without having to learn new user interfaces or data editing and analysis tools for each model. They enable model developers to concentrate on the science behind their models instead of having to reinvent the common functionality that is part of these modelling environments. Individual models would need to be modified to read from and write to the environment's data format, and would then benefit from being able to use the common tools within the framework. 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 dynamic feedbacks between the models would still exist, although in some cases, for example DeltaShell, these environments may include some means of directly coupling models (Krause *et al.*, 2005).

4.1.3 Custom Coupling of Specific Models

Running one or more models in parallel requires each model to be run one timestep at a time, with values of modelled variables being exchanged between models at each timestep. 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 (Figure 4-1d) 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 timestep and there needs to be some sort of controller that commands each model, or a component of each model, to run for the next timestep. Alternatively, for the models to communicate with each other directly, they each need to provide some sort of publicly accessible software interface, for example the Component Object Model (COM) interface protocol. The interface protocol 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 protocol, which may not be possible if the models are proprietary software. 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.

4.1.4 Model Interface Specifications

As noted by Krause *et al.* (2005), one of the model integration development paths for integrated modelling has been the development of model interface specifications (Figure 4-1e,f) such as OpenMI (Blind and Gregersen, 2005; Gregersen *et al.*, 2005; Moore and Tindall, 2005; Gregersen *et al.*, 2007; Knapen *et al.*, 2009), the Common Component Architecture (CCA) (Bramley *et al.*, 2000; Armstrong *et al.*, 2006) 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 (Section 4.1.5), 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 (Figure 4-1f) in the model code or creating a wrapper around the model (Figure 4-1e). 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 to make it suitable for wrapping. 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 graphical user interfaces. Links would then be created between appropriate variables in each model. It is important to note that it is specific configurations of each model that are linked, not the model engines themselves.

There are two approaches to indirectly controlling the flow of a simulation by linked models, pull mechanisms and push mechanisms. Pull mechanisms start at the point where a result is required, with requests for data values being filtered up through links and, where necessary, processes are called until the required result has been calculated. Push mechanisms start at the point where a piece of information is available, with data values being filtered down through the links and then each process being run when all its input variables are available. The linked model run is initiated by an external trigger on one of the models. 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 that conform to the same interface specification. Krause *et al.* (2005) conclude that, at that time, the OpenMI approach to model coupling was the most sophisticated. Hoheisel (2002) state that tight coupling of models usually uses shared memory to communicate between models usually coded in the same programming language and requiring a lot of effort to achieve integration, while loose coupling is more flexible, often using asynchronous communication between models.

4.1.5 Modular Modelling Frameworks

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 (Figure 4-1g) such as MMS (Leavesley *et al.*, 2002), OMS (Ahuja *et al.*, 2005; Kralisch *et al.*, 2005), TIME (Rahman *et al.*, 2003; Argent and Rizzoli, 2004; Rahman *et al.*, 2004; Rahman *et al.*, 2005; Murray *et al.*, 2007) 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 models 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 developer of a legacy model which is to be modularised must choose to adopt 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. Jones *et al.* (2001) conclude that a modular modelling approach, where new model components can be easily added, maintained and modified, facilitates integration of knowledge from different disciplines.

4.2 Selection of a Model Linkage System

The requirement for integrated assessment of water resources and advances in computer programming technologies have resulted in numerous innovative endeavours to provide software tools for integrated modelling. A review of several model linkage systems was conducted by Clark et al. (2013). The reviewed systems are listed and briefly described in Table 4-1. Integrated modelling environments were not included in the 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.

Table 4-1 Model linkage systems reviewed in Clark *et al.* (2013)

System	Description
Open Modelling Interface (OpenMI)	Standard interface to facilitate linking models, operating at various spatial and temporal scales, such that feedbacks between modelled processes can be represented (Blind and Gregersen, 2005; Moore and Tindall, 2005; Gregersen *et al.*, 2007).
Object Modelling System (OMS)	Modular framework for developing environmental models, including facilities for data provision, testing, validation, and deployment (David *et al.*, 2004; Ahuja *et al.*, 2005; Kralisch *et al.*, 2005; David *et al.*, 2010).
Jena Adaptable Modelling System (JAMS)	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)
The Invisible Modelling Environment (TIME)	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)
LIQUID®	Modelling framework for modelling hydrological processes enabling integrated models composed of reusable modules to be built and run (Branger et al., 2010a; Branger et al., 2010b).
High Level Architecture (HLA)	A computer architecture for constructing distributed simulations, facilitating interoperability between different simulations and simulation types and promoting reuse of simulation software modules (Dahmann *et al.*, 1997; Lindenschmidt *et al.*, 2005; Jagers, 2010).
Common Component Architecture (CCA)	A component architecture for scientific high-performance computing, providing platform independent inter-component communication mechanisms, enabling parallel computing across components and configuration of components before and during execution (Armstrong *et al.*, 2006; McCartney and Arranz, 2009; Jagers, 2010).

Clark *et al.* (2013) suggested that the systems reviewed could be categorised into two main groups: CCA, HLA and OpenMI which are purely interface specifications, whereas OMS, JAMS, TIME and LIQUID® are modular modelling frameworks which include a system for linking models or process modules. OpenMI, OMS, JAMS, TIME and LIQUID® are designed primarily for use in the water and environmental modelling domain. HLA was designed for use in the defence domain. CCA is a general purpose linking system and is suitable for use in a high performance computing operating environment. The coupling interfaces defined by CCA, OMS, OpenMI and TIME are similar in that that they all include similar initialise, run, finalize, get data and set data concepts, but differ in the amount of code needed to implement the interface, and in run time performance (Lloyd *et al.*, 2009; Jagers, 2010).

When linking either whole models or process specific 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 suitably 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 likely to be 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 built from modules 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, which makes a model linkage system hard to implement, and too little flexibility, which will reduce the number of situations in which the system can be applied.

The objective of the review was to select a suitable model linkage system for use in linking the *ACRU* model to other models to provide a holistic water resources modelling system for use in water resources planning and operations. An initial goal was to link the *ACRU* model with a river network model. The requirements were that:

- the linkage system needed to be suitable for use with the *ACRU* model,
- the linkage system should require minimal changes to the code of the *ACRU* model,
- ideally the linkage system should be regarded as an international standard,
- the linkage system should not be proprietary so as not to place a financial burden on users of the linked models,
- the linkage system should have been implemented for a range of other models so that other compatible models can be linked to *ACRU* in the future,
- the linkage system should enable the models to be linked in parallel so that dynamic feedbacks between models can be adequately represented,
- the linkage system should enable the linked models to be run at different spatial and temporal scales, and
- the linkage system should enable execution of the linked models in a time that is not substantially longer than the total execution time of the individual models.

It was recommended by Clark *et al.* (2013) that, in order of preference, the OpenMI, TIME and OMS systems were the most suitable for linking the *ACRU* model to a river network model. All three systems are primarily intended for use in the water and environmental modelling fields. The advantages of OpenMI are: (i) that it is generally accepted as a *de facto* international standard, (ii) is strongly supported by the OpenMI Association, (iii) has been widely adopted by key research and commercial players providing a useful set of compliant water related models, and (iv) has been well documented. The advantages of TIME are: (i) its lightweight architecture, (ii) it is strongly supported by the eWater CRC, (iii) 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 (iv) has been well documented. The advantages of OMS are: (i) its lightweight architecture, and (ii) that it has been moderately implemented by the developers, providing a small set of compliant water and environmental models, though these models are tailored to United States Geological Survey (USGS) and United States Department of Agriculture (USDA) requirements. Clark *et al.* (2013) selected OpenMI for implementation with the *ACRU* model as it met all the requirements and was judged to be the most appropriate solution.

4.3 Overview of the OpenMI Model Interface Standard

OpenMI is 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*". The OpenMI Standard is made freely available in both a .Net and a Java version. In 2014 Version 2.0 of the OpenMI Standard was approved by the Open Geospatial Consortium (OGC®) as an OGC standard (Open Geospatial Consortium, 2014).

The purpose of this section is to provide a brief overview of the OpenMI model interface standard and associated Software Development Kit (SDK), to provide the necessary background to its implementation in the *ACRU* and eWater Source models. A more comprehensive review of OpenMI can be found in Clark *et al.* (2013) and the OpenMI Association's website [www.openmi.org] provides links to the official documentation and

other information. Documentation available for OpenMI 2.0 includes the following documents:

- *Scope for the OpenMI* (OpenMI Association, 2010e),
- *OpenMI Standard 2 Specification* (OpenMI Association, 2010d),
- *OpenMI Standard 2 Reference* (OpenMI Association, 2010c),
- *The OpenMI 'in a Nutshell'* (OpenMI Association, 2010b),
- *What's New in OpenMI 2.0* (OpenMI Association, 2010f),
- *Migrating Models* (OpenMI Association, 2010a), and
- *OGC® Open Modelling Interface: Interface Standard* (Open Geospatial Consortium, 2014)

4.3.1 OpenMI Terminology

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

- *engine* - generic representation of a process or processes, consisting of the algorithms or calculations used to model the process or processes;
- *model* – when the *engine* is run it reads the data for a 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 it 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;
- *composition* – a set of two or more connected *linkable components*; and
- *migration* – the process of implementing the OpenMI interface standard in an *engine*.

4.3.2 Requirements for Models to be Suitable to Implement OpenMI

For a model engine to be suitable for migration to become an OpenMI linkable component, the following main requirements need to be met (OpenMI Association, 2010a):

- The model engine needs to be structured so that model initialisation is performed separately from the computation of the model algorithms.
- The values of model input and output variables (quantities) representing boundary conditions should be retrieved by the model algorithms, on a timestep-by-timestep basis.
- The model engine needs to be structured so that the call to compute the model algorithms for each timestep can be accessed by third-party software for the purpose of controlling the linked simulation.
- The model engine needs to expose information about the model input and output variables (quantities), which can be accessed by other models or by third-party software for the purpose of configuring the model linkages.
- The model engine needs to be able to provide the values of modelled quantities and, if relevant, their associated spatial and/or temporal attributes to other models.

4.3.3 The *IBaseLinkableComponent* Interface and Model Execution Phases

The OpenMI Standard consists of numerous interfaces. The key interface is the *IBaseLinkableComponent*, shown in Figure 4-2, which must be implemented by all OpenMI linkable components. In addition, if the linkable component is a timestepping model then the interfaces of the *OpenMI.Standard2.TimeSpace* extension, primarily the *ITimeSpaceComponent* interface, would also need to be implemented. Linkable components may also implement the optional *IManageState* and *IByteStateConverter* interfaces if required.

An OpenMI linkable component provides different services during different phases of its execution (OpenMI Association, 2010d). The status of the linkable component changes as it progresses through the different phases. The different states are shown in the *LinkableComponentStatus* enumeration in Figure 4-2 These phases are briefly explained as follows (OpenMI Association, 2010d):

- Initialisation – Instantiation of a linkable component as an object and initialisation of the model using model specific configuration and input files. The linkable component

should then be able to expose the input and output exchange items through which it can be linked to other linkable components.

- Inspection and Configuration – The input and output exchange items that have been made available are inspected and then the required connections between these pairs of input and output exchange items are configured and validated, including any exchange adapters that may be required.
- Preparation – Any additional model preparation, such as creation of model output files, is done.
- Computation – The computation phase on the linked model simulation starts with a call to the *Update* method of the last linkable component in the computation chain. This initiates a request-reply service between the linkable components which is repeated for every timestep until the end of the simulation period.
- Completion – During this phase model output files may be written and closed and other clean-up operations are performed before the models are closed.

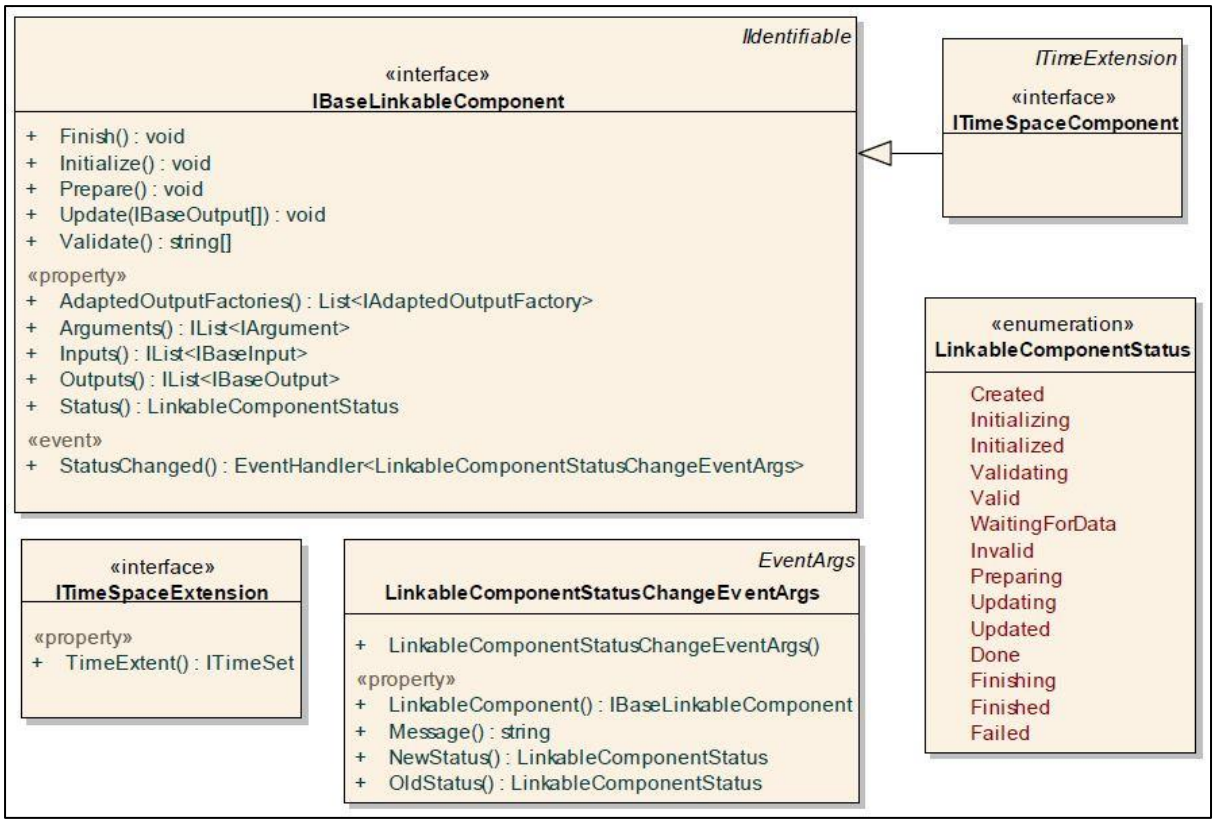

Figure 4-2 The OpenMI *IBaseLinkableComponent* interface (OpenMI Association, 2010c)

4.3.4 Timestepping and Flow of Data in OpenMI

The sequence of model execution during the computational phase of a linked model simulation is described by Blind and Gregersen (2005) as a pull driven mechanism, which means that one linkable component makes a request to another component for a set of quantity values for a given time for one or more specified elements (spatial locations). Gregersen *et al.* (2007) describe this as a 'request and reply' mechanism, and explain 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.

The linked model simulation starts with a call to the *Update* method of the last linkable component. For this last linkable component, the consumer, to be able to complete its next timestep it calls the *GetValues* method on the next linkable component up the chain, the provider. The provider linkable component will return the requested quantity values if they are available for that timestep, and if not available, the *Update* method on this provider component will be called. As shown in Figure 4-3, this sequence of *GetValues* and *Update* calls is repeated up a chain of linked components until each component in the chain can complete the current timestep. A new call to the *Update* method of the last linkable component for its next timestep starts the whole cycle again, until the end of the complete time period to be simulated. If the linkable components have different timesteps then some linkable components may have to be run for several timesteps before the input required for the timestep of the consumer component model can be provided. As shown in Figure 4-3, not all chain computation layouts are simple linear unidirectional chains, and in some cases it may be necessary for a provider component to provide an estimate to the consumer or to redirect the consumer to another provider component.

It is clear that in a model linking system such as OpenMI the information describing each link between two or more models is important. A provider linkable component makes one or more output exchange items available and for each one specifies information such as the data type of the values, the units of measure and the spatial location of the modelled spatial entities for which data is available. Similarly a consumer component declares one or more input exchange items and for each one specifies information about the data it requires. In OpenMI 2.0 a connection is made between two linkable components by specifying one or more pairs of connected input and output exchange items, that is provider-consumer pairs. However, a provider and a consumer may not be completely compatible and an output adapter may need to be inserted between the provider and the consumer to handle unit of measure transformations and differences in temporal and spatial resolutions. These

adapters are both consumers and providers and provide the person linking two models with a versatile mechanism to ensure that the information passed between models is compatible. Some examples of connections between linkable components using adapters are provided in Figure 4-4.

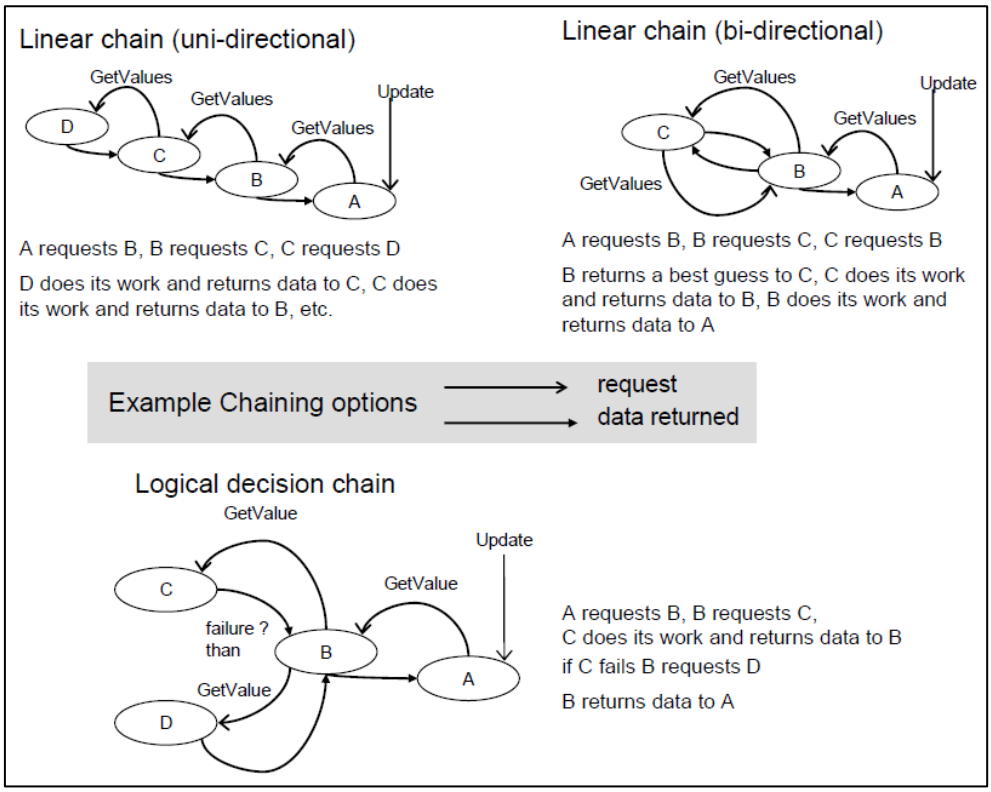

Figure 4-3 Different chain computation layouts (OpenMI Association, 2010d)

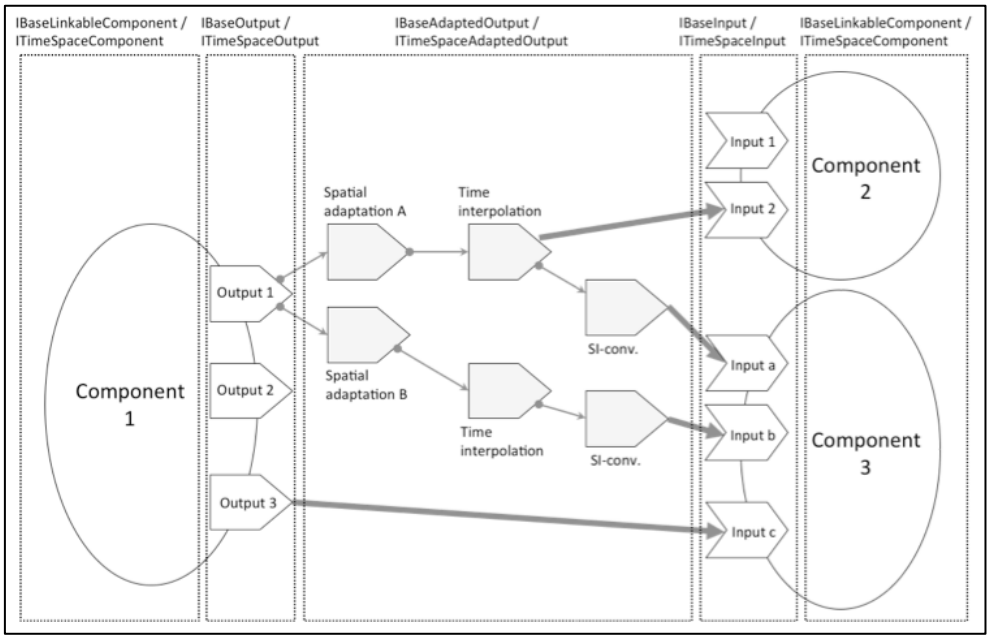

Figure 4-4 Example of OpenMI Interfaces and the flow of data between components (Open Geospatial Consortium, 2014)

4.3.5 Implementations of the OpenMI Standard

The OpenMI Standard is a collection of either .Net or Java interfaces that a model developer needs to implement for their model to be OpenMI compliant and thus able to link to other OpenMI compliant models. These interfaces are just a blueprint specifying a standard of "behaviour" that a model needs to conform to and the OpenMI Standard does not include any actual code implementation.

Separate to the development of the OpenMI Standard, the OpenMI Association Technical Committee (OATC) developed an implementation of the Standard in C# for the .Net platform and included tools that it used to implement and verify the Standard (OpenMI Association, 2010b). The OATC's Software Development Kit (SDK) is a library of C# classes that provide an implementation of the Standard and includes examples and unit tests. The OATC's Configuration Editor is a simple graphical user interface that enables users to configure compositions of linked OpenMI compliant models. The SDK and the Configuration Editor were made available under a public open source licence to assist users in implementing the Standard in their models and then applying linked models. However, OpenMI Association (2010d) states that the OpenMI Association does not formally support them. OpenMI Association (2017) states that this was due to: (i) a lack of funding to do this themselves, and (ii) the development of an open source GUI and SDK as part of the FluidEarth initiative, led by HR Wallingford, which aims to support integrated modelling through the development and support of software tools to assist in the implementation of OpenMI. The FluidEarth SDK and other tools, described by Harpham *et al.* (2014) and (Harpham *et al.*, 2016), are available on the SourceForge webpage for FluidEarth [http://sourceforge.net/projects/fluidearth]. One of the tools developed in the FluidEarth initiative was Pipistrelle, which is a graphical user interface that can be used to create and then run compositions of linked models. Tutorials describing the application of the FluidEarth SDK and other tools can be found on the FluidEarth eLearning webpage [http://eLearning.fluidearth.net/]. OpenMI Association (2017) indicates that there is an initiative by a third party to develop a Java SDK for OpenMI 2.0. However, Knapen (2015) stated that due to a lack of funding for this work the Java SDK had not been completed.

To make either an existing or a new model engine OpenMI compliant there are two main options: (i) to directly implement the OpenMI Standard interfaces in the code of the model engine, or (ii) to create an OpenMI compliant linkable component wrapper around the model engine. The second option is usually preferable for existing models, and is the only option for proprietary models where the developer of the wrapper does not have access to the

source code for the model engine. Both the OATC and the FluidEarth SDKs provide ready-built classes that can be used to created linkable component wrappers for models.

4.3.6 Application of OpenMI

As a standard for linking models, OpenMI is only really useful if it is widely adopted by model developers so as to provide a range of OpenMI compliant models representing different modelling domains within the water resource system. Gijsbers *et al.* (2010) and OpenMI Association (2017) list numerous OpenMI compliant models from a range of model developers internationally. However, many of these models appear to have been made compliant to Version 1.4 of the Standard and it is not clear how many of these models would also have been made compliant to Version 2.0 of the Standard. OpenMI Association (2017) indicates that it is mainly the models from Deltares, DHI and HR Wallingford that have been officially recognised by the OpenMI Association as being compliant to Version 2.0 of the Standard. However, based on the publications relating to OpenMI listed by OpenMI Association (2017), OpenMI has been applied for a wide range of purposes, consequently there may be many other OpenMI compliant models. One interesting example is the OpenMI compliant AquaCrop crop water productivity model described by Foster *et al.* (2017).

The application of OpenMI in several multi-disciplinary projects requiring integrated modelling was evaluated by Knapen *et al.* (2013). One finding was that though both model developers and model users recognised the value of OpenMI compliant models, the actual work of making a model compliant was not the primary goal of either group. Knapen *et al.* (2013) concluded that OpenMI was useful for integrating both existing and new models and that it was suitable for linking models outside the hydrological domain for which they were originally developed. Bulatewicz *et al.* (2010) commented on the importance of validation of both the individual and the integrated models, especially in instances where feedbacks between the modelled systems occur. They concluded that the software development effort was reduced through reuse of existing models using the OpenMI linking system, and that the use of a standard such as OpenMI enabled the compliant models to be used in other studies.

4.4 Development of an OpenMI Linkable Component for *ACRU*

As a result of the restructuring of the *ACRU* model, described in Chapter 3, the *ACRU* 5 version of the model engine meets all the requirements, stated in Section 4.3.2, for a model

engine to be suitable for migration to become an OpenMI linkable component. The restructuring of the *ACRU* model engine enabled it to be initialised and run in timestepping mode by third party software, with the provision for exchange of both model input and output variable values at each timestep. Object-oriented design and implementation using an object-oriented programming language are not requirements for OpenMI migration. However, the object-oriented structure and programming of the *ACRU* 5 version facilitated the development of an OpenMI linkable component by providing access to model input and output variable values through the Component and Data object structure, including metadata about the variables.

The development of OpenMI 1.4 linkable components for the *ACRU* model for both Java and .Net was described in Clark and Lutchminarain (2013). At that time it was decided that an OpenMI 1.4 linkable component would be developed for *ACRU* as: (i) the OpenMI 2.0 SDK for Java was still under development, (ii) some problems experienced with the initial release of the OpenMI 2.0 OATC SDK for .Net, and (iii) few OpenMI 2.0 compliant models were available. The development of the OpenMI 2.0 Standard and the OpenMI 2.0 FluidEarth SDK, have since been completed and thus an OpenMI 2.0 linkable component for .Net was developed as a wrapper around the *ACRU* model in this study, as will be explained in this section. The development of an OpenMI 2.0 linkable component for Java would be difficult without a Java version of the OpenMI 2.0 SDK. The lack of a Java version of the OpenMI 2.0 SDK also means that there are likely to be few, if any, OpenMI 2.0 compliant Java models available.

As explained in Clark and Lutchminarain (2013) 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 (packaged as a Java archive file named *ACRU.jar*) to a .Net assembly named *ACRU.dll*.

This study had access to the *ACRU* source code, which would have enabled direct implementation of the OpenMI *IBaseLinkableComponent* interface. However, a decision was made to rather use the model engine wrapper classes provided by the FluidEarth OpenMI 2.0 SDK. The reasons for selecting this implementation option were: (i) this option

enables OpenMI compliance to be separated from the model itself, leaving the model untouched, and (ii) the wrapper classes from the SDK already include functionality that would have to be duplicated.

The developers of OpenMI recognised that model engines that do timestep-based computations would have common requirements with regard to being made OpenMI compliant, which lead to the development of the *LinkableEngine* wrapper class in the OATC's SDK (OpenMI Association, 2010a). The FluidEarth OpenMI 2.0 SDK includes similar classes for wrapping model engines. In this study the *BaseEngine* and *BaseComponentTimeWithEngine* classes from the FluidEarth OpenMI 2.0 SDK were used to develop a linkable component for the *ACRU* model using the C# programming language. These classes, together with their superclasses, are shown in the UML diagram in Figure 4-5. The separate *BaseEngine* and *BaseComponentTimeWithEngine* classes enable one implementation of the *BaseEngine* class to be used by more than one implementation of the *BaseComponentTimeWithEngine* class. This would enable a model engine to expose itself to OpenMI in different ways for use in different situations.

The *Acru_Engine* class, which extends the abstract *BaseEngine* class, is the wrapper around the *ACRU* model engine itself. The role of the *Acru_Engine* class is to: (i) create an instance of the *ACRU* model engine, (ii) initialise the engine by reading in the model input files and configuring the model, (iii) provide the model data values requested by another model, (iv) set model data values provided by another model, (v) run the next simulation timestep when instructed to do so, (vi) keep track of the current *ACRU* simulation time, (vii) translate between Java and .Net date-time representations, and (viii) close down the instance of the *ACRU* model engine at the end of the linked simulation.

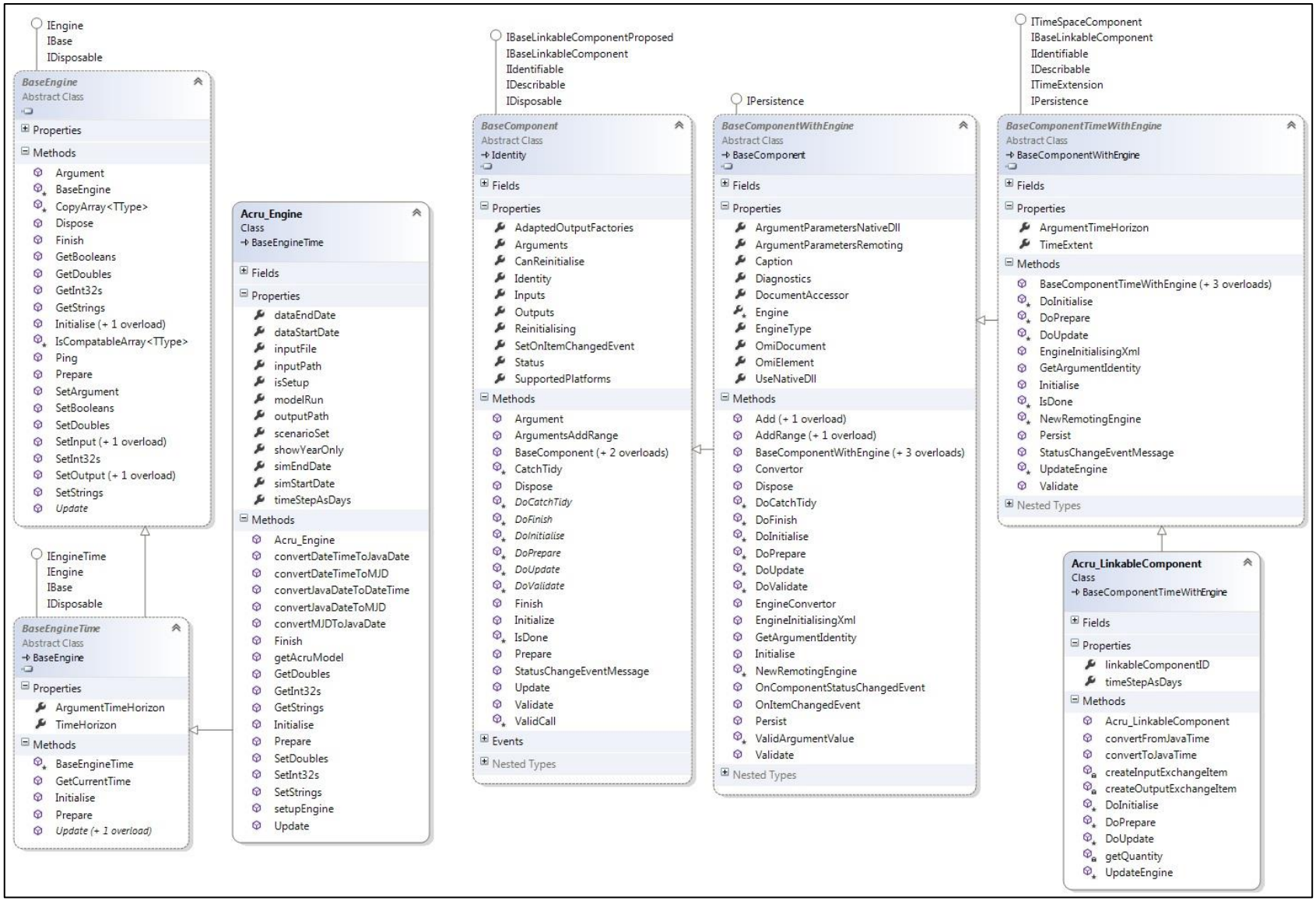

Figure 4-5 UML class diagram of the *ACRU* OpenMI 2.0 linkable component classes

The *Acru_LinkableComponent* class extends the abstract *BaseComponentTimeWithEngine* class, and is effectively a wrapper around the *Acru_Engine* class. The *Acru_LinkableComponent* class exposes selected parts of the *ACRU* model to the "outside world" and acts as a translator between the model and OpenMI. The role of the *Acru_LinkableComponent* is to: (i) specify the arguments that must be provided to the *ACRU* model engine for it to be able to initialise itself (for example, the path to the model input file), (ii) specify the input exchange items, the input variables into which *ACRU* could potentially receive data values from other models, (iii) specify the output exchange items, the *ACRU* output variables from which *ACRU* could provide data values to other models, (iv) determine the time horizon, that is the simulation period, for which the *ACRU* model can be run, (v) in the *UpdateEngine* method initiate a call to the *Update* method of the associated instance of *Acru_Engine*, and (vi) translate between Java and .Net date-time representations. The *Acru_LinkableComponent* internally creates input and output exchange items for a wide range of *ACRU* input, output and state variables.

As explained in Clark and Lutchminarain (2013) one 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 timestep model and so all fluxes are implicitly per day. This was resolved by providing duplicate input and output exchange items for each flux variable, one as a quantity and one as a rate.

Another problem which was overcome was the translation between Java and .Net date-time representations. OpenMI uses a modified Julian Day to represent time and these are calculated internally from instances of the .Net *System.DataTime* class. However, though a .Net compiled version of *ACRU* had been created using the IKVM.NET tool, this version of *ACRU* returned date-time values as instances of a .Net compiled version of the *java.Util.Date* class which represents dates differently to the .Net *System.DataTime* class. This problem was overcome with some simple methods to translate between the .Net and the Java representations of date-time.

The wrapper classes provided in the FluidEarth OpenMI 2.0 SDK made it possible to create a working linkable component for the *ACRU* model in a short space of time, and with relatively few lines of code. No additional code or changes to code were required in the *ACRU* model to achieve this, although to some extent the *ACRU* 5 version of the model had been developed with the OpenMI migration requirements in mind. As stated previously, the development of a .Net linkable component wrapper for the *ACRU* 5 version of the model was

greatly facilitated by the object-oriented structure and also the configuration information which fully describes the data type of each variable including the units of measure. The open source nature of the FluidEarth OpenMI 2.0 SDK also made it easier to investigate and understand how the wrapper classes worked.

The linking of the *ACRU* and eWater Source models, for the purpose of demonstrating the application of the OpenMI linkable component, is described in Section 5.5. Initially it was intended that the daily timestep *ACRU* model would be linked to the eWater Source model which would be set to run at a daily timestep. However, due to the relatively small size of the upper uMngeni Catchment used for the case study, it was realised that it would be necessary to run eWater Source at an hourly timestep to demonstrate the lag and attenuation of flows routed down the river reaches in eWater Source. Thus, an OpenMI adapted output was required to be able to disaggregate daily runoff volumes, simulated by *ACRU,* to hourly runoff volumes that could be used by eWater Source as inflows to the modelled river network. This presented an opportunity to investigate and apply OpenMI adapted outputs as part of the connections between two models as shown in Figure 4-4. An adapted output class was created by extending the abstract *AdaptedOutputTimeBase* class from the FluidEarth OpenMI 2.0 SDK. This new adapted output class named *Adapter_DisaggregateRunoff_SCS_UH_DailyToHourly* disaggregates daily surface runoff volumes to hourly volumes in the *AdaptRecords* method by applying the SCS unit hydrograph (UH) approach (Schmidt and Schulze, 1987) and a synthetic rainfall distribution (Weddepohl, 1988) used in the flow routing module of *ACRU* 3 and *ACRU* 2000 (Smithers and Caldecott, 1995). This adapter requires two arguments: (i) the catchment lag time, and (ii) the rainfall intensity zone. An adapted output factory class was required to dynamically create instances of the new output adapter. The *ACRU_OpenMITools_AdapterFactory* class implements the *IAdapterOutputFactory* interface from the OpenMI 2.0 Standard. A UML class diagram showing these two new classes is shown in Figure 4-6.

Though the primary purpose of OpenMI is to link models, the OpenMI 2.0 Standard and SDK makes better provision than OpenMI 1.4 for data files and databases to be wrapped as linkable components, which are able to provide linked models with direct access to stored data at run time (Donchyts *et al.*, 2010). This was tested by creating a linkable component for the AcruCSV format file used for input and output of time series data in *ACRU* 5, as shown in Figure 4-7. This linkable component was created using the same approach that was used for the *ACRU* linkable component, where the *AcruCSV_Engine* class is a wrapper directly around the Java *ModelDataAccess.DataReaderWriter_AcruCSV* class compiled to .Net. The *AcruCSV_LinkableComponent* class does the translation between the wrapped

AcruCSV file and OpenMI. In principle it should be possible to create a single wrapper class that extends the abstract *BaseComponentTime* class provided by the FluidEarth OpenMI 2.0 SDK, but an initial attempt to do this was not successful and would need to be investigated further.

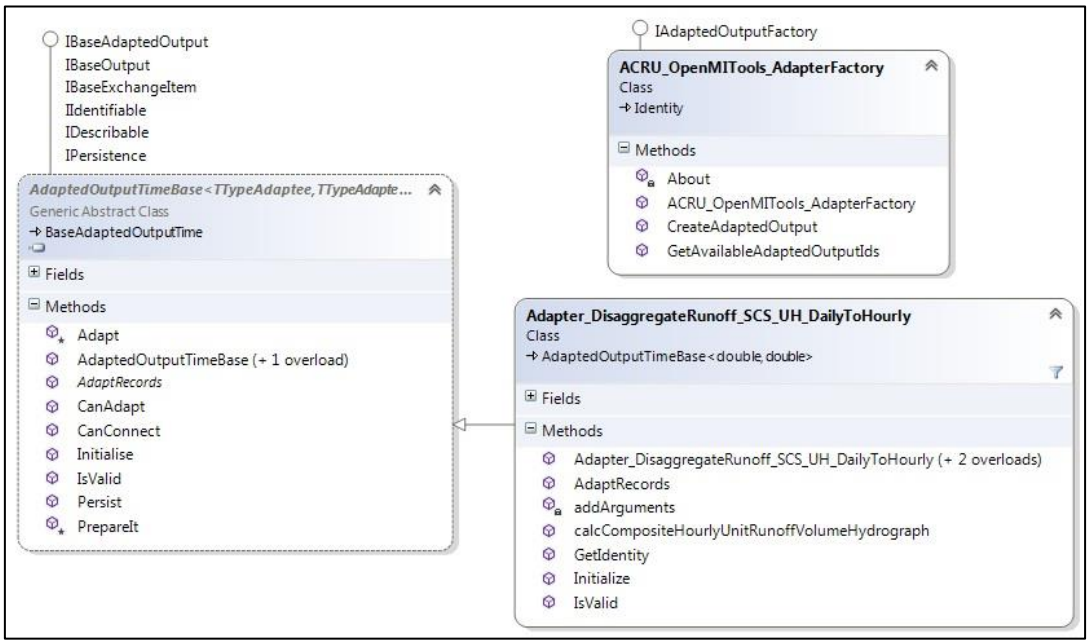

Figure 4-6 UML class diagram of the OpenMI 2.0 adapter class used to estimate hourly runoff from daily runoff

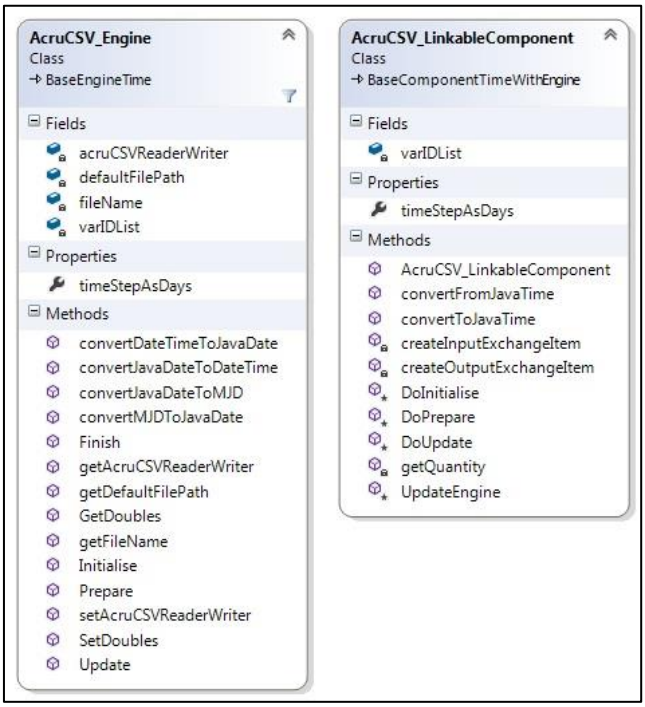

Figure 4-7 UML class diagram of the OpenMI 2.0 linkable component class for the AcruCSV file format

4.5 Development of an OpenMI Linkable Component for eWater Source

As is motivated in the introduction to Chapter 4, the eWater Source model was selected for evaluation in this study for potential use as a river network model linked to the *ACRU* model and to demonstrate linking *ACRU* to another model using OpenMI. An inspection of the installed files for eWater Source indicated that eWater Source might be OpenMI compliant. Davis (2015) confirmed that eWater Source was OpenMI compliant, but commented that this had not yet been included in the documentation. Davis (2015) explained that the RiverSystem.OpenMI.XMLCreator.exe tool, which is part of the eWater Source installation, allows an eWater Source project file to be selected and for an OpenMI OMI-file (*.omi) to be generated for the project. This tool worked as expected but for unknown reasons it was not possible to use the project as a linkable component within the FluidEarth configuration tool Pipistrelle, mentioned in Section 4.3.5. This led to the decision to create a separate new wrapper for eWater Source using the FluidEarth OpenMI 2.0 SDK, to test the feasibility of creating a linkable component for a third party model without any access to the source code for the model.

As shown in Figure 4-8 the same FluidEarth OpenMI 2.0 SDK wrapping approach was used as for the *ACRU* model. The FluidEarth OpenMI 2.0 SDK were used to develop a linkable component for the eWater Source model (public version 4.1.0.4337) using the C# programming language The *eWaterSource_Engine* class extends the abstract *BaseEngine* class as the wrapper around the eWater Source model engine, using the *RiverSystem.ApplicationLayer.Simulation.SimulationHandler* class as the model engine. To open and initialise a eWater Source project within a linkable component the following information must be specified as arguments by the user: (i) the path to an existing eWater Source project file (*.rsproj), (ii) the name of an existing project scenario, (iii) the simulation timestep to be used, (iv) the simulation start and end date, and (v) the folder path to which output files are to be written. The *eWaterSource_Engine* class includes code to determine the eWater Source input and output variables using the *GetInputMetaParameters* and *GetOutputMetaParameters* methods of the *SimulationHandler* class. Additional code was added to the *eWaterSource_Engine* class to write model output variables to CSV files in a specified output path. The *eWaterSource_LinkableComponent* class extends the abstract *BaseComponentTimeWithEngine* class. The *eWaterSource_LinkableComponent* class internally creates input and output exchange items for all the available eWater Source project input and output *metaparameters*. It was possible to determine the units of measure for each *metaparameter* using functionality within the *SimulationHandler* class. In eWater Source extensive use of was made of eWater Source *Functions* to act as input variables to

receive data from *ACRU* via OpenMI connections. Similarly eWater Source *Functions* and *ModelledVariables* were used as output variables to provide data for use by *ACRU* via OpenMI connections. Setting the correct *Result Units* and *Time of Evaluation* settings on these *Functions* was important. The *Result Units* field in the *Functions* was used to correctly handle the units of measure of the data passed between the models using OpenMI, rather than using OpenMI adapted outputs for unit of measure conversions.

As was the case with *ACRU*, it was possible to create a working linkable component for the eWater Source model in a short space of time, and with relatively few lines of code. However, not having access to the source code for the model, meant that the Application Programming Interface (API) for the eWater Source classes was the only source of information available and thus development of the linkable component required some trial and error.

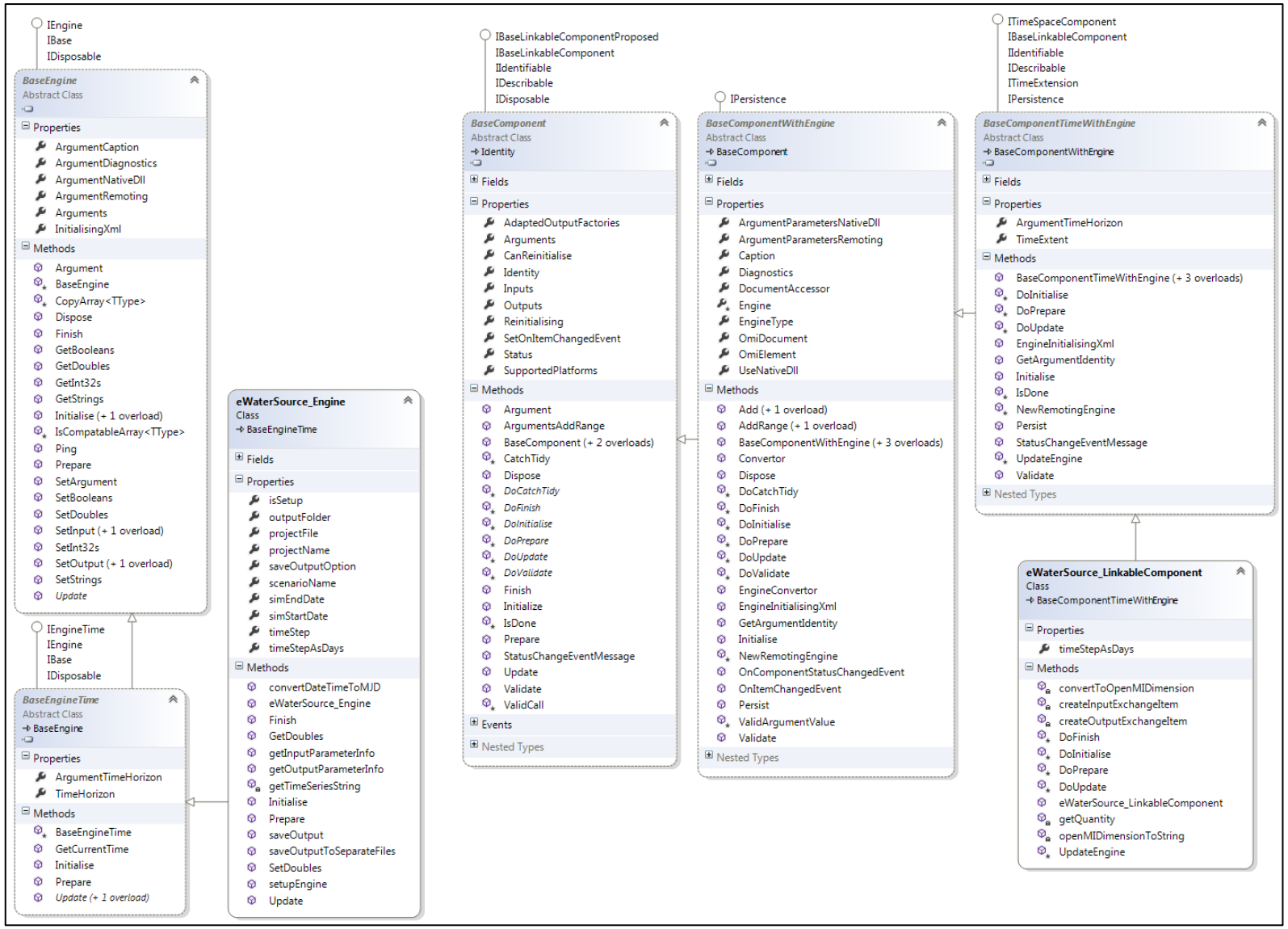

Figure 4-8 UML class diagram of the eWater Source OpenMI 2.0 linkable component

4.6 Time Differences in Linked Models and OpenMI

The greatest challenge in wrapping and linking the two models lay in understanding how *ACRU*, eWater Source and OpenMI each conceptualised simulation timesteps, simulation start and end dates, and the timestamps used to define these. OpenMI requires a time horizon (the simulation period) to be returned by each linkable component, and if these are different then a time horizon corresponding to the overlap between the individual linkable component time horizons is used. However, this can cause problems in the models so ideally the linked models need to be configured for the same simulation period. However, even more important is the timestamp associated with the current timestep of each model as a simulation progresses. For the purpose of this discussion, consider a linked simulation with a duration of 5 days starting on 2017/01/01 and ending on 2017/01/05. The different model engine and OpenMI linkable component times and also the timing of data requests and flows are shown for two timesteps in Figure 4-9.

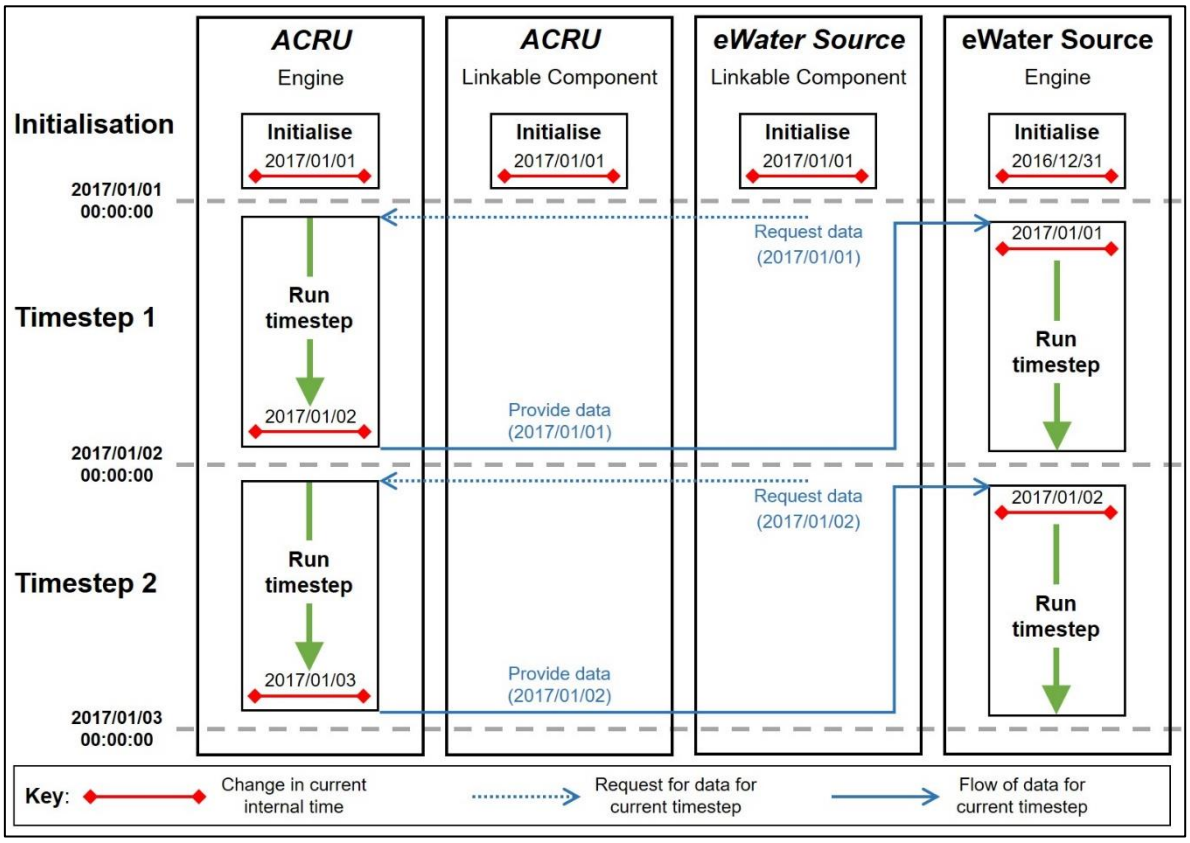

Figure 4-9 Time in the *ACRU* and eWater Source models linked using OpenMI

ACRU is a daily model (timestep of one day), it starts with an initial state at the beginning of the first day, e.g. 2017/01/01 (2017/01/01 00:00:00), and runs to the end date, e.g. 2017/01/05 (2017/01/06 00:00). During the first timestep the model has the internal current

date of 2017/01/01 which changes to 2017/01/02 at the end of the first timestep, so the state values are effectively the state at 2017/01/02 00:00:00, which makes sense. The *ACRU* output time series files store daily non-state flux data (average flow, evaporation) as values for the day on which they occurred. But these same files store state data (dam storage, soil moisture) as the value at the end of the day (e.g. the state at 2017/01/02 00:00:00 is stored against date 2017/01/01), because the state is queried and saved to output before the model's current date is changed to 2017/01/02.

In eWater Source the timesteps, varying from annual to six minutes, can be selected so the time component of the timestamp is important. In eWater Source, if a start date-time of 2017/01/01 00:00:00 is specified then at initialisation eWater Source seems to set the internal current date-time to the start date-time minus one timestep (2016/12/31 00:00:00 for a daily timestep, 2016/12/31 23:00:00 for an hourly timestep). When eWater Source starts the first timestep it first advances its internal current time by one timestep (back to 2017/01/01 00:00:00) to model the first day from 2017/01/01 00:00:00 to 2017/01/02 00:00:00), though the internal current time will only change to 2017/01/02 00:00:00 at the start of the second timestep. In eWater Source the end date-time should be specified as the date-time at which the last timestep starts (2017/01/05 00:00:00 for a daily timestep, 2017/01/05 23:00 for an hourly timestep.

Irrespective of the internal current time in each of the linked models, OpenMI seems to be solely governed by the common horizon defined by a start date-time and end date-time for the models. Hence, to model five days means that the start date-time needs to be 2017/01/01 00:00:00 and the end date-time needs to be 2017/01/06 00:00:00, which may not correspond to the start date, current date and end date values retrieved from the models, so it is important for the linkable component classes for the individual models to take this into account.

4.7 Linking the *ACRU* and eWater Source Models Using OpenMI

Using OpenMI, two or more linkable components can be linked to create an OpenMI 'composition'. However, the configuration of an OpenMI composition of two or more models requires connecting individual corresponding variables for each hydrological component, which requires a detailed knowledge of the models to be linked and is not necessarily a simple task. The concept of having a rainfall runoff model (*ACRU*) provide simulated runoff flows to a river network model (eWater Source) seems relatively simple until feedbacks between parts of the two modelled domains are considered. A typical example of such

feedbacks are irrigated areas, where: (i) the climate-crop-soil water balance is modelled in *ACRU*, (ii) the irrigation demand is determined in *ACRU* and sent to eWater Source, (iii) the actual irrigation water quantity supplied, limited by availability, is determined in eWater Source, (iv) the actual irrigation is applied in *ACRU*, and (v) potentially some surface runoff and baseflow from excess irrigation may be generated in *ACRU* and sent to eWater Source. To demonstrate linking the *ACRU* and eWater Source models using OpenMI for the upper uMngeni case study catchment (Section 5.5), such that feedbacks between the two modelled domains are represented, requires several different types of connections between the models to be configured, as shown in Figure 4-10. The lighter blue solid-line connections represent surface runoff and baseflow from the various response units in *ACRU* which were passed to an *Inflow* node in eWater Source. The surface runoff and baseflow quantities were passed in separate connections, so that the daily surface runoff from *ACRU* could be disaggregated to hourly values using an adapted output. The daily baseflow quantities are simply converted by eWater Source from a daily rate to an hourly rate. The grey dotted-line connections represent cases where information was passed between the models, these were usually water requests sent from *ACRU* to *Time Series Demand* nodes in eWater Source. The darker blue solid-line connections represent connections where water supply quantities calculated in eWater Source were passed from *Time Series Demand* nodes in eWater Source to *ACRU*. In the case of wetlands, to be consistent with the manner in which *ACRU* models wetlands, the soil moisture deficit in the topsoil horizon of the wetland was calculated in *ACRU* and sent to eWater Source from which, based on the flow calculated in Source, water was supplied to *ACRU* where it entered the topsoil horizon. In the case of irrigated areas and urban areas in *ACRU*, water requests were sent to *Time Series Demand* nodes in eWater Source, which were then supplied to *ACRU* if sufficient water was available in the eWater Source water source. For urban areas, return flows calculated in *ACRU* were passed to an *Inflow* node in eWater Source.

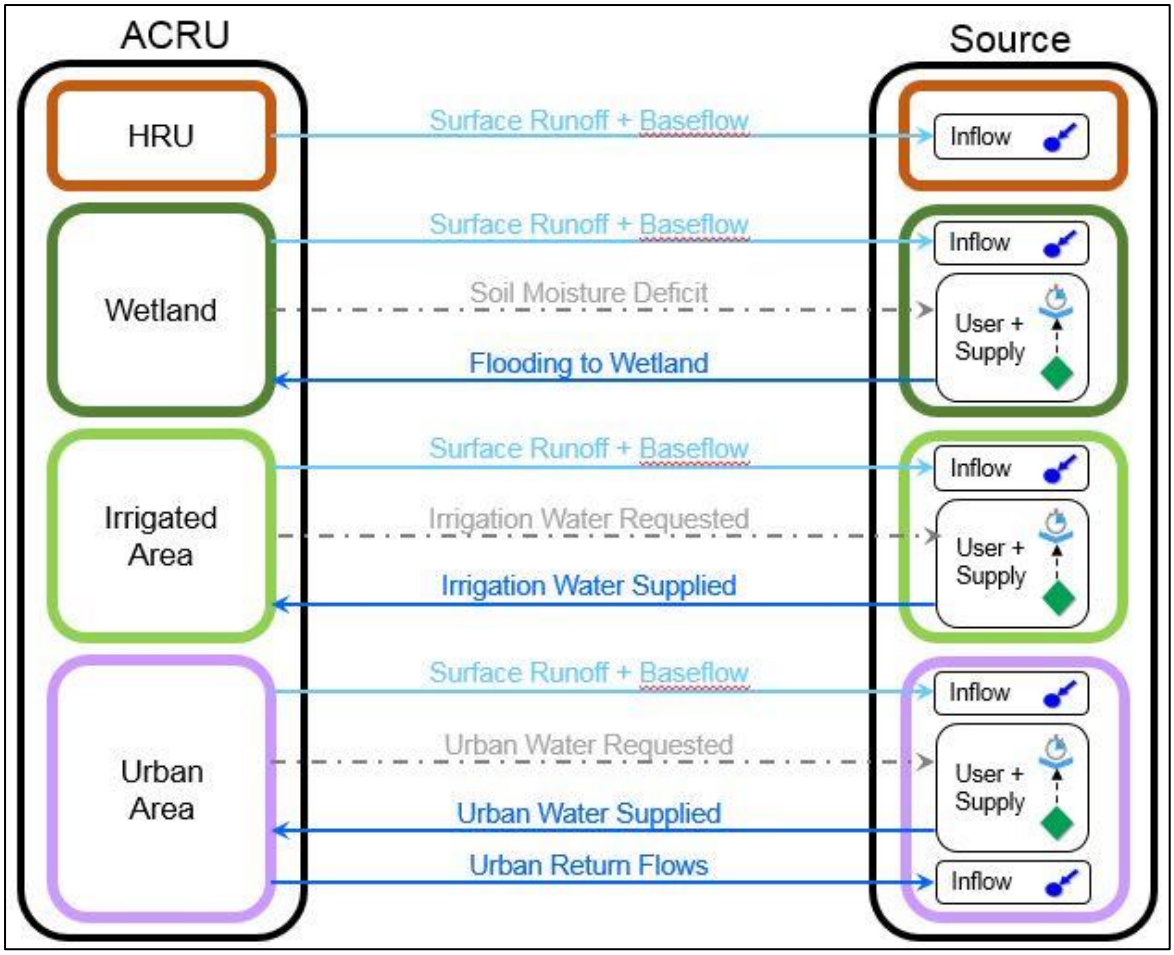

Figure 4-10 The different types of connections between *ACRU* and eWater Source

The FluidEarth Pipistrelle graphical user interface tool, mentioned in Section 4.3.5, is useful for creating and editing compositions with a relatively small number of connections and adapters between the models. However, this would be a tedious exercise if there were a large number of connections and adapters to be configured, as was the case with the case study in the upper uMngeni Catchment (Chapter 5) for the linked *ACRU* and eWater Source models. Thus, some new tools were developed to make it easier to create and run OpenMI compositions:

(i) A general tool consisting of an XML file format to store information about a composition, and a corresponding library of C# classes to read and write these XML files and to run an OpenMI composition. The XML file format and the library of C# classes are described briefly in Appendix 8.6.

(ii) A plugin to the graphical user interface for eWater Source, described briefly in Appendix 8.7, which uses information from a user specified *ACRU* model input file to create a corresponding eWater Source project and OpenMI configuration.

(iii) A plugin to the graphical user interface for eWater Source, described briefly in Appendix 8.7, which enables the user to select and run an OpenMI composition.

4.8 Discussion

The newly developed facility to link *ACRU* with models representing other domains, using OpenMI, is an innovation that is expected to enhance *ACRU*'s application as a tool for IWRM. The development of OpenMI compliant wrappers for both *ACRU* 5 and eWater Source using the FluidEarth .Net implementation of the OpenMI 2.0 Standard was relatively straightforward. The object-oriented structure of the restructured *ACRU* model facilitated the easy development of an OpenMI 2.0 wrapper for the model. The OpenMI 2.0 wrapper for *ACRU* will enable *ACRU* to be linked to other OpenMI 2.0 compliant models and thus facilitate the use of *ACRU* as part of integrated water resource modelling systems which can utilise the strengths of different domain models. One of the objectives of the OpenMI 2.0 Standard was to facilitate the linking of models to data sources, in addition to linking models, and this will also facilitate the use of *ACRU* within integrated modelling frameworks that have implemented OpenMI 2.0 without any changes or additions to the *ACRU* code. It should be noted that it would not have been possible to develop an OpenMI wrapper for the *ACRU* 3 version of the model operating in distributed mode, due to each subcatchment being modelled individually for the full simulation period (Figure 2-3), compared to the *ACRU* 5 version where all subcatchments are modelled within each timestep (Figure 2-4).

It is unfortunate that a Java implementation of the OpenMI 2.0 Standard has not been completed. However, in spite of this, the compilation of the *ACRU* Java code to .Net using the IKVM software worked well and enables *ACRU* to be linked with OpenMI compliant models developed in .Net, which are likely to be in the majority. Towards the end of the development of the OpenMI 2.0 wrapper for *ACRU* a statement by Frijters (2017) indicated that development of the IKVM software had been discontinued. Thus, an alternative solution to compiling Java code to .Net may need to be investigated in the future.

Although the source code for the eWater Source model was not available it was still possible to develop an OpenMI 2.0 wrapper as the model fulfilled the necessary criteria for wrapping. However, the development of the wrapper for eWater Source did highlight the need for a good knowledge of the model being wrapped.

When linking models it needs to remembered that there will now be two or more models being loaded into memory simultaneously. Thus, the technical specifications of the computer on which the linked models are to be run need to be considered. Buahin and Horsburgh (2015) state that the increase in simulation time for loosely coupled models is mainly due to: (i) initialisation and disposal of model components - increasing with the

number of exchange items and their complexity, and (ii) the data transformations and transfers between components - increasing with the number of model components, the complexity of the model components and their exchange items, the number of OpenMI connections and the length of the model timesteps. The benefits resulting from linking models need to be balanced against costs such as increased configuration and simulation times (Buahin and Horsburgh, 2015).

Although the Pipistrelle tool makes is easy for model users to link two OpenMI compliant models, the user will required a detailed understanding of both the models being linked and also some knowledge of how OpenMI works. Validation of two or more models linked using OpenMI will also be important to ensure that the links have been correctly configured. While Pipistrelle is a useful GUI tool for configuring model linkages, it would be tedious to use in instances where there are numerous links to be created between various components of the linked models. In the case of these more complex model linkages it would be advantageous to write code to configure the model linkages. The composition information classes, described in Appendix 8.6, would make writing this code easier, and are a useful contribution resulting from this study.

The *ACRU* and eWater Source models are compatible in the way in which they conceptualise water resource systems. The water flow and management functionality of eWater Source complements the *ACRU* surface land cover/use modelling functionality well. While the two models can be applied in linked mode using OpenMI, as demonstrated in Section 5.5, this is not straightforward.

In conclusion, the fourth objective of the research study, stated in Section 1.4, to develop a means of linking *ACRU* with other models, has been achieved through the development of an OpenMI linkable component for *ACRU*. In addition, the innovative linking of the *ACRU* and eWater Source models, representing feedbacks between the two modelled systems, has provided a non-trivial demonstration of linking two models on a timestep-by-timestep basis. The suitability of the *ACRU* model engine for migration to OpenMI was largely a result of the restructuring described in Chapter 3. The investigation and implementation of OpenMI as a model linkage system described in this chapter demonstrated that it was possible to:

- Develop an OpenMI 2.0 linkable component for *ACRU*, for which there was a good understanding of the model and access to the source code.
- Develop an OpenMI 2.0 linkable component for a model input data file in AcruCSV format.

- Develop an OpenMI 2.0 linkable component for eWater Source, for which there was initially little understanding of the model and no access to the source code.
- Link two models using OpenMI such that feedbacks between the two modelled systems were represented.
- Link two models using OpenMI where the models were run at different timesteps, *ACRU* daily and eWater Source hourly.
- Develop an OpenMI adapted output to do a non-trivial adaption from daily runoff in *ACRU* to hourly runoff for use in eWater Source.

5 CASE STUDY - UPPER UMNGENI CATCHMENT

The purpose of this chapter is to demonstrate and discuss, using a case study catchment, how the objectives of this research study have been achieved. In the context of using the *ACRU* model to produce simulated estimates of the hydrological variables required to compile catchment-scale water resource accounts, it will be shown that the restructured *ACRU* model is: (i) more flexible enabling more realistic representation of the physical components of complex water resource systems, (ii) extensible through the addition of the new *Accounting* module, and (iii) includes improved handling of time series data. The new ability to link *ACRU* to other models using OpenMI is demonstrated by linking *ACRU* to the eWater Source model, using eWater Source to do hydrologic routing of streamflow from *ACRU* through river reaches and dams, taking abstractions of water for urban and irrigation use into account.

5.1 Overview of the upper uMngeni Catchment

The uMngeni Catchment is situated in the summer rainfall region, within the KwaZulu-Natal province of South Africa. The catchment has an area of 4455 km^2 and the altitude ranges from 2064 m in the West to sea level in the East. The Mean Annual Precipitation (MAP) varies from 1550 mm in the West to 700 mm in the drier middle part of the catchment (Warburton, 2011). The uMngeni River is the main source of water for the city of Pietermaritzburg with a population of 618 536 (StatsSA, 2012), the city of Durban and the greater eThekwini metropolitan area with a population of 3 442 361 (StatsSA, 2012), and several smaller towns including Hilton, Howick, New Hanover and Wartburg. The uMngeni River is regulated by four large dams (Midmar, Albert Falls, Nagle and Inanda). The catchment is categorised as being fully developed (DWA, 2013) and is currently augmented with transfers from the Spring Grove Dam and the Mearns Weir on the Mooi River in the neighbouring uThukela Catchment. The bulk water utility Umgeni Water is responsible for providing potable water within the catchment. The eThekwini metropolitan area is the second largest commercial and industrial area in South Africa (DWAF, 2004) and significant future population growth is anticipated in the eThekwini/Pietermaritzburg area due to urbanisation and economic growth. Rural areas include subsistence and commercial farming, with extensive irrigated agriculture, cultivation of sugarcane and commercial forestry plantations (DWAF, 2004). Streamflow in the catchment is largely perennial and there is relatively little extraction of groundwater (DWAF, 2004).

The uMngeni Catchment configuration of the *ACRU* model developed by Clark (2015d), to demonstrate the application of an integrated water resources accounting methodology, formed the foundation for the case study presented in this study. However, for the purpose of this study it was decided to focus just on the upper portion of the uMngeni Catchment, that is, the portion of the catchment upstream of and including Albert Falls Dam. Due to the position of the dams and urban areas in the uMngeni Catchment, the upper uMngeni catchment is a critical part of the water supply system due to the higher rainfall and that, together Midmar Dam and Albert Falls Dam (via releases to Nagle Dam), provide water to the city of Pietermaritzburg, to the surrounding small towns and to a large portion of the eThekwini metropolitan area. Although the Inanda Dam is also an important part of the system, its relatively low altitude means that it is best suited to providing water to the lower altitude portions of the city of Durban and neighbouring coastal towns due to the cost of pumping. The location of the upper uMngeni Catchment is shown in Figure 5-1. A detailed map of the upper Umgeni Catchment is shown in Figure 5-2, including Quaternary Catchments, subcatchment boundaries, rivers, major dams, urban areas and water transfers into and out of the catchment.

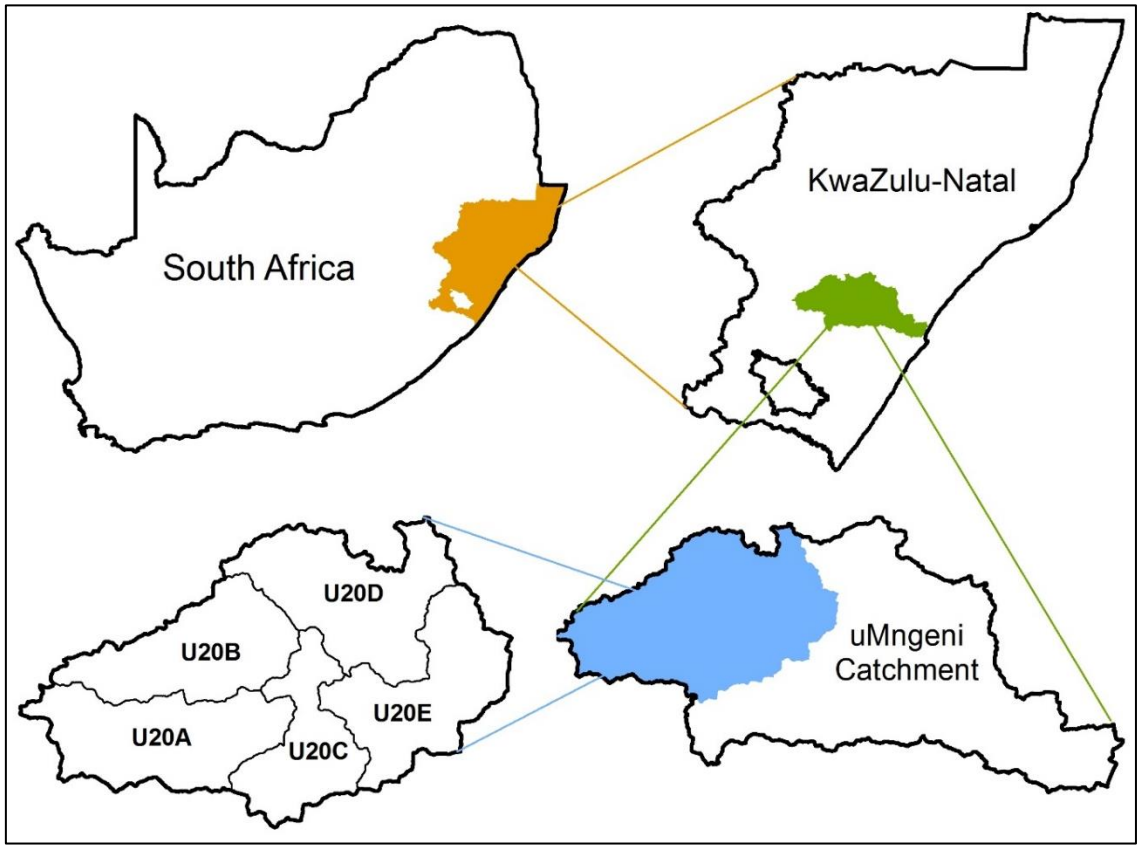

Figure 5-1 Locality map and Quaternary Catchments for the upper uMngeni Catchment

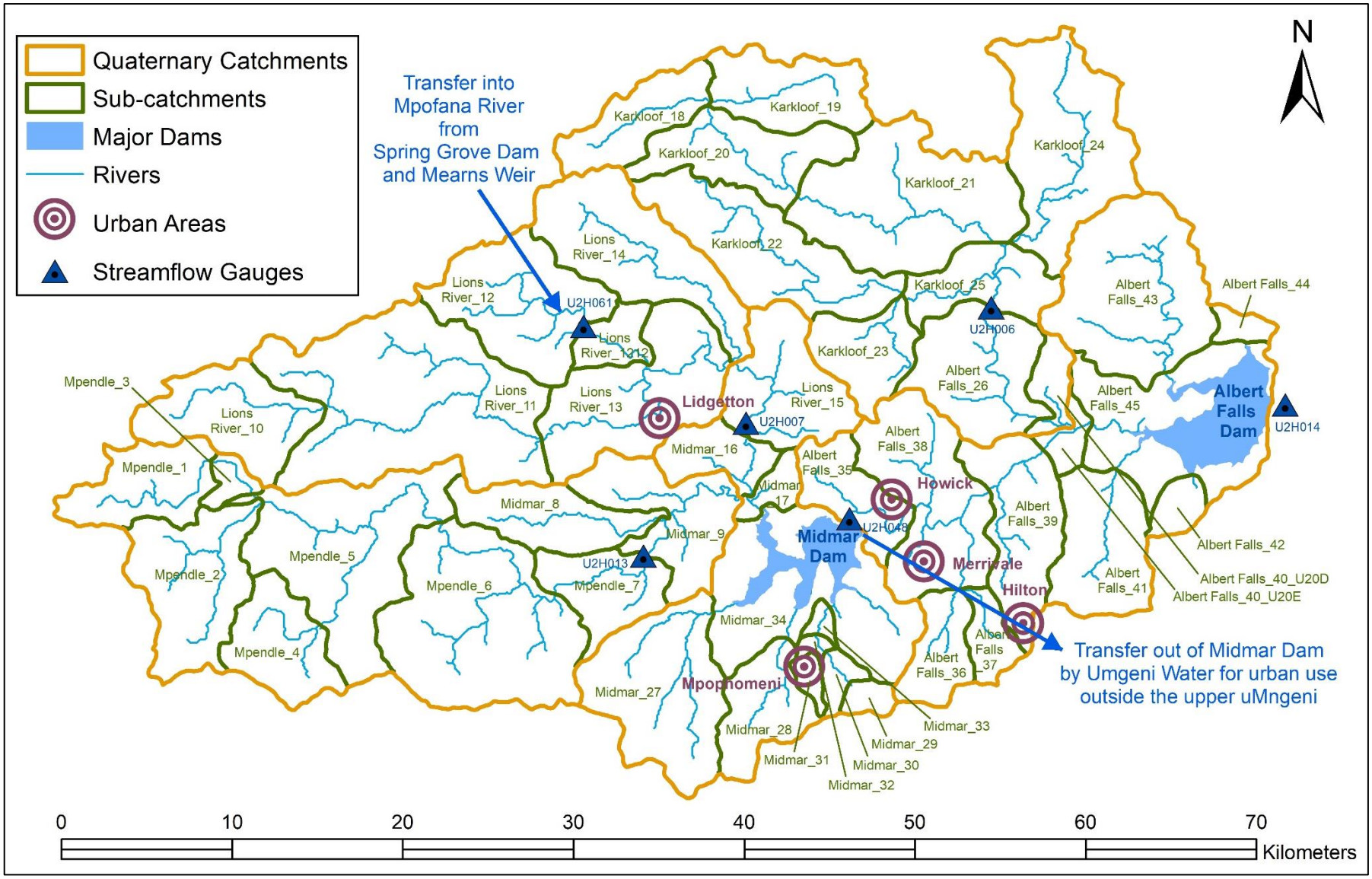

Figure 5-2 Catchments, rivers, major dams, urban areas, water transfers in the upper uMngeni Catchment

5.2 Configuration of the *ACRU* and eWater Source Models

In a research project titled "*Development And Assessment Of An Integrated Water Resources Accounting Methodology For South Africa*" (Clark, 2015a) a methodology was developed to produce annual catchment-scale water resource accounts, as described in Clark (2015b) and summarised in Appendix 8.8. A hydrological modelling approach was adopted to provide the various water balance components required for the water resource accounts. Much of the effort required to produce the accounts was related to identifying suitable datasets, processing these datasets and then using these datasets to configure the *ACRU* model. The methodology and Python scripts used to process the datasets and configure *ACRU* are described in Clark (2015b). The same methodology, with a few enhancements, was applied to configure *ACRU* for the upper uMngeni Catchment. The methodology and datasets used to configure *ACRU* and eWater Source for the case study are described in more detail in Clark (2015d) and in Appendix 8.9.

5.3 Verification of the Simulations

The *ACRU* model has been applied and verified extensively in South Africa (Schulze, 1995d) and has also been applied in several other countries. Verification studies specifically relating to the uMngeni Catchment include: Tarboton and Schulze (1991), Tarboton and Schulze (1992), Kienzle *et al.* (1997), Schulze *et al.* (2004), Kiker *et al.* (2006) and Warburton *et al.* (2010). The *ACRU* 4 version of the model is currently being applied in research and consulting contexts, and is used in teaching hydrology and engineering undergraduate students at UKZN. The *ACRU* 5 version of the model has thus far been applied for research purposes in the upper uMngeni Catchment (Clark, 2015d), the Sabie-Sand Catchment (Clark, 2015c), the upper uThukela Catchment (Clark, 2017a) and the upper and central Breede Catchment (Clark, 2017b), as part of a methodology for compiling annual catchment-scale water resource accounts. The representation of hydrological processes in the *ACRU* 5 version of the *ACRU* model, discussed in this chapter, is almost identical to the *ACRU* 3 version. Therefore, it was not intended for this case study to provide a detailed verification of the *ACRU* model. However, the *ACRU* model was run for the period 1 October 2007 to 30 September 2016, a total of nine hydrological years, using the first year as a warmup period to initialise small dam, soil and groundwater storages. The simulated streamflow and dam storage volumes were compared to measured streamflow and dam storage volumes. The detailed results are discussed in Appendix 8.10. In summary, the seasonal trends in the measured streamflow were generally represented well, but with substantial over or underestimations in the magnitude of the flow in some events and seasons. The trends in

the simulated flows appeared to be associated with the trends in the estimated rainfall. In the relatively high rainfall upper uMngeni Catchment, with a strong seasonal variation in rainfall and with rainfall frequently occurring in the form of high intensity storms, rainfall is the primary driver of hydrological responses. It was concluded that the poor degree of association between the simulated and the measured streamflow was most likely to be due to: (i) differences in actual and estimated rainfall volumes, and (ii) mismatches in the timing of peak flows. It was also concluded that the mismatches in the timing of peak flows was possibly partly due the *ACRU* model not lagging and attenuating flows as they proceed down river reaches and through dams.

As stated at the beginning of this section, the *ACRU* model has previously been applied successfully in the uMngeni Catchment. Thus, although there is always scope to improve understanding and model representation of hydrological process, the model is at a mature stage of development and a certain degree of trust in the model has been established. However, this study has highlighted two possible areas for further investigation related to processes, stormflow generation and flow routing. The suitability of the SCS equation to estimate stormflow, used in the *ACRU* model (Schulze, 1995c), when using spatially averaged rainfall values from remotely sensed rainfall products, as opposed to point rain gauge measurements, needs to be investigated further, but is beyond the scope of this study. In addition a simple method of at least lagging flows should be considered for inclusion in the *ACRU* model to improve the timing of flows, especially down long river systems, for use in instances where detailed flow routing is not required.

5.4 Application of the Restructured Object-Oriented *ACRU* Model

The rationale for restructuring the *ACRU* model is discussed in Section 2.2 and the design objectives to be achieved in the restructured model are listed in Section 3.2. The purpose of this section is to use the case study catchment to demonstrate how the achievement of these design objectives has resulted in more versatile and flexible *ACRU* model. Although this section focusses primarily on the *ACRU* model engine, many aspects of the enhanced functionality of the object-oriented model engine depend on the newly developed XML model input file structure which complements the object-oriented structure of the engine.

5.4.1 Nested Catchment Structure

The object-oriented model structure in which the physical components of the hydrological system are more explicitly defined, the object-oriented concept of objects being composed of

other sub-component objects and the ability to define relationships between objects, has resulted in a model that has a more representative and more flexible structure. One example of how this has improved the *ACRU* model is the nested structure of subcatchments within catchments within bigger catchments, as shown in Figure 5-3. As described in Section 2.1.1, the *ACRU* 3 version of the model conceptually represents hydrological systems as a set of subcatchments (shown as green squares in Figure 5-3) numbered in sequential flow order, but not explicitly belonging to a parent catchment. The streamflow network is created by specifying a downstream subcatchment for each subcatchment. In the *ACRU* 5 version, subcatchment Component objects can be configured as residing within a parent (container) catchment Component object (orange outlines), which in turn may reside within a bigger catchment component object (purple outline), resulting in a structure of nested subcatchments and catchments. The nested system of subcatchments and catchments has two advantages: (i) it makes it possible to determine the simulated flows at different catchment scales by querying the simulated flows at the subcatchment or catchment outflow nodes, and (ii) it makes it possible to aggregate other spatial variables, such as rainfall or total evaporation at different catchment scales. The latter facility, to be able to aggregate spatial variables up through a series of nested catchments, made it possible in the new *ACRU Accounting* module to create sets of nested water resource accounts at different catchment scales.

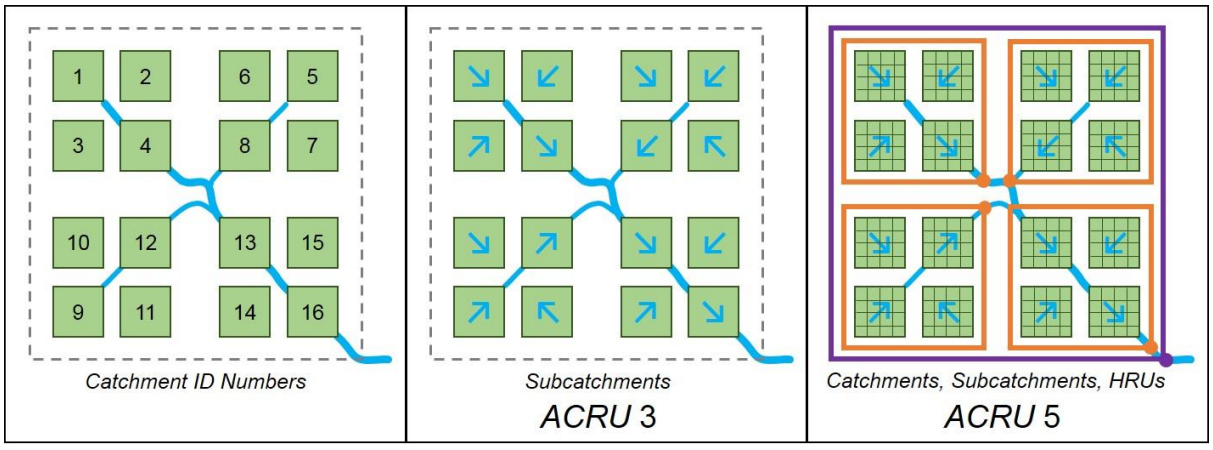

Figure 5-3 Representation of HRUs within nested subcatchments and catchments

5.4.2 Flexible Configuration of Subcatchments

The availability of more detailed satellite remotely sensed land cover/use datasets has led to the requirement for more flexibility to include a greater number of land cover/use HRUs within a subcatchment. As discussed in Section 2.1.1, and shown in Figure 2-2, the *ACRU* 3 version represented a subcatchment as having a single dominant land cover/use, with the

option to also include a dam, an irrigated area, an adjunct impervious area and a disjunct impervious area. Common workarounds for this inflexible subcatchment configuration were to: (i) calculate area weighted averages of land cover characteristics, or (ii) to use subcatchments to represent individual HRUs. One of the main reasons for restructuring *ACRU* was to enable more flexible configuration of spatial components within subcatchments. The object-oriented model structure, complemented by the XML model input file structure has made this possible. In the *ACRU* 5 version a subcatchment may have any number of HRUs within a subcatchment (shown as subdivided green squares in Figure 5-3), usually based on land cover/use classes, but are not usually spatially explicit within a catchment. For example each subcatchment may contain: (i) several HRUs representing different natural vegetation types, (ii) several HRUs representing different dryland crops, (iii) several HRUs representing different irrigated crops, (iv) impervious areas, (v) wetlands, (vi) more than one dam, and (vii) one or more river reaches. However, it should be noted that this flexibility to represent a large number different HRUs within each subcatchment will not necessarily result in better simulations as many HRUs may have similar hydrological responses.

Many of the subcatchments modelled in the upper uMngeni include several different land cover/use classes, as shown in Figure 8-45. The increased flexibility in configuring subcatchments was used in this case study. One of the purposes of the water resource accounts was to represent sectoral water use, and modelling each land cover/use class as a HRU made it possible to account for the water use by different sectors.

In this case study the increased flexibility in configuring subcatchments was also used in the representation of dams within a subcatchment. In the upper uMngeni catchment there are a large number of farm dams, as shown in Figure 8-44, many of which are not on the main river reach running through a subcatchment. It is not practical to individually model the subcatchment providing runoff to each of these small dams. A typical approach used in configuring the *ACRU* 3 version is to lump all the dams into a single dam which is assumed to be either: (i) situated within the subcatchment, but off the main river channel, or (ii) an in-channel dam situated at the downstream exit of the subcatchment. In this case study, as described in Appendix 8.9.5, within each subcatchment: (i) all the small unregistered farm dams were lumped and modelled as a single dam, off the main river reach, with no irrigation abstractions, (ii) all the registered farm dams, for which more information was available, were lumped and modelled as a single dam, on the main river reach, with irrigation abstractions, and (iii) Midmar Dam and Albert Falls Dam were modelled as individual dams on the main river reach. In each subcatchment, where relevant, two runoff regions were

determined, the regions upstream and downstream of unregistered and registered farm dams, as shown in Figure 8-44. An example of how a subcatchment may be configured internally in *ACRU* 5, as was done in this case study, is shown in Figure 5-4, comparing it to a typical configuration in *ACRU* 3. In the region upstream of farm dams, runoff is received by the lumped unregistered dam, which then flows into the lumped registered dam, which then flows through the region downstream of dams to the exit of the subcatchment. The irrigated areas in both the regions are assumed to be supplied with water from the lumped registered dam. In subcatchments without dams, the runoff would flow directly to the exit of the subcatchment and irrigation would be from run-of-river.

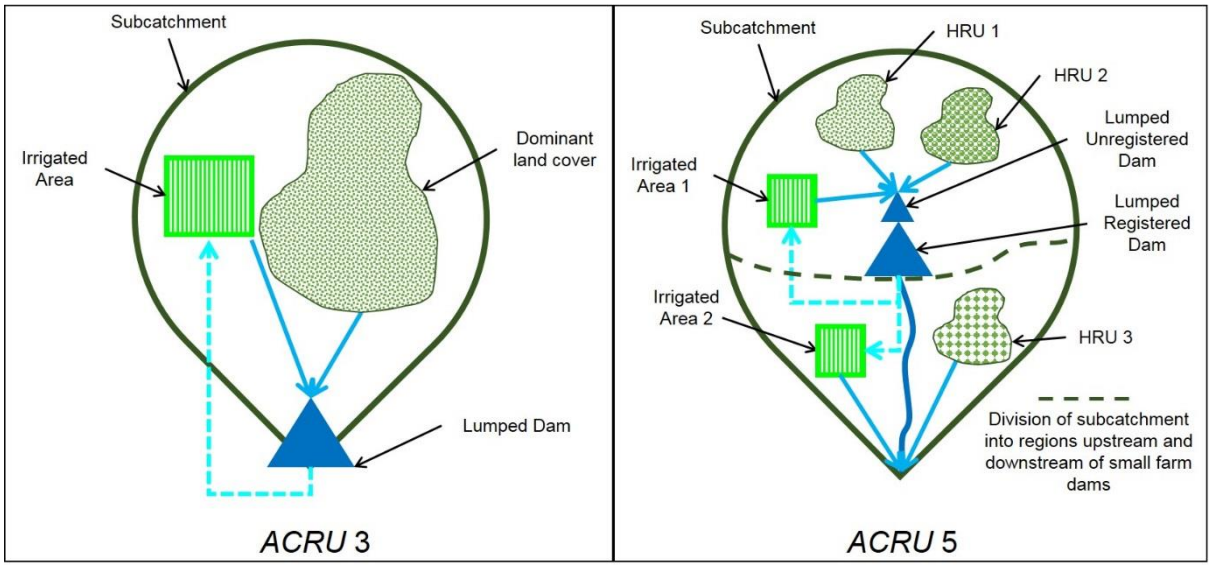

Figure 5-4 Example of more flexible configuration within subcatchments

5.4.3 Flexible Configuration of the Flow Network

The more flexible configuration of the spatial components contained within subcatchments to more realistically represent heterogeneous land use, as discussed in Section 5.4.2, requires the configuration of the flow connectivity between these spatial components. The object-oriented model structure in which the physical components of the hydrological system are more explicitly defined also enables the flow network to be configured more flexibly. Each subcatchment contains a subcatchment node at its downstream exit. Within each subcatchment a flow network of nodes, river reaches and dams may be configured ending with the subcatchment node. Similarly each catchment contains a catchment node at its downstream exit. The smaller scale subcatchment flow networks are connected to create larger scale catchment flow networks. The flow network is configured by specifying upstream and downstream relationships for each node, river reach and dam.

5.4.4 Flexible Configuration of Engineered Flows

Engineered flows of water for irrigation and urban use can add significant complexity when configuring a hydrological model and creating catchment water accounts. This is especially the case where water is abstracted from one subcatchment and used, possibly with return flows, in a different subcatchment. Typically in *ACRU* 3 water for irrigation would be from a river or dam in the same subcatchment, although the "loopback" option (Smithers and Schulze, 1995) does enable water for irrigation to be supplied from an upstream subcatchment. The *ACRU* 3 version does not make direct provision for water abstractions and return flows for urban water use, though there are options to represent transfers of water out of river reaches, and into or out of dams. In this case study catchment the assumption that irrigation will be from a river or dam in the same subcatchment is reasonable as there are no large-scale water schemes. However, there are significant abstractions of water from Midmar Dam for urban use in Mpophomeni and Lidgetton upstream, and Howick, Merrivale and Hilton downstream, located as shown in Figure 5-2, in addition to the supply of water for urban use outside the upper uMngeni Catchment. To meet the need to represent these engineered water flows for urban use in the water resource accounts, new functionality was added to the *ACRU* 5 version to model daily urban water requirements by *CUrbanWaterUser* Components in a simple manner. For both irrigation and urban water users the *ACRU* 5 version requires that a water source Component be specified, usually a dam or a river node, where the water source component may be in another subcatchment which may be upstream or downstream or in a completely different catchment.

The ability to model engineered water flows between catchments creates some difficulties when determining the order in which the different components and their processes are to be executed. In *ACRU* 3 each subcatchment is simulated for the full simulation period, for example 10 years, before moving on to the next subcatchment which is then simulated for the full simulation period, as shown in Figure 2-3. This means that only water transfers to downstream irrigation water users can be modelled, and only if the downstream requirement is already known. It was for this reason that one of the objectives of restructuring *ACRU* was to enable a parallel processing approach, such that each hydrological process on each subcatchment is executed each day before continuing to the next day, as shown in Figure 2-4. Representing Components and Processes as objects in *ACRU* 5, together with some code to determine the computation order for Components and Processes objects, made it possible to implement such a parallel processing approach. This made it possible in this case study to simulate daily water urban water requirements and the supply of water from Midmar Dam to meet these requirements for urban areas both upstream and downstream.

5.4.5 Flexible Handling of Time Series Data

Umgeni Water extracts water from Midmar Dam to supply bulk water to municipalities, both within and outside the upper uMngeni Catchment. The flow rate data for DWS gauging station U2H049, a flow meter measuring abstractions from Midmar Dam, was downloaded from the DWS website [http://www.dwa.gov.za/Hydrology/Verified/hymain.aspx]. This data showed that for the hydrological years 2007/2008 to 2014/2015, on average, the annual abstraction volume from Midmar increased by about 5.4% but, due to drought induced restrictions put in place by Umgeni Water, abstractions reduced from 130 Million m^3/annum in 2014/2015 to 120 million m^3/annum in 2015/2016. The inter-catchment transfers from Spring Grove Dam and Mearns Weir on the Mooi River, in the neighbouring uThukela Catchment, into the uMngeni Catchment are also significant and vary from year to year depending on availability in the Mooi River system and needs in the uMngeni system. For example, in the 2015/2016 hydrological year, a drought year, the inter-catchment transfer in contributed 112 million m^3/annum, which is nearly half the 235 million m^3 full capacity of Midmar Dam. Thus, it is important to be able to represent these variations over time in the *ACRU* model inputs.

In the *ACRU* 3 version, transfers into and out of catchments would typically be represented using: (i) the DOMABS variable for abstractions out of a river, (ii) the XDRAFT variable for abstractions out of a dam, and (iii) the PUMPIN variable for transfers into a dam. These variables contain 12 month-of-year flow rate values, and thus vary from month to month within a year but not from year to year. With some additional effort it would be possible to use the *ACRU* 3 dynamic file option to specify transfer rates that change monthly and annually. In this case study daily data on flow rates for the inter-catchment transfer and the Umgeni Water abstraction were available. The more flexible handling of time series data and file formats in the model and the XML model input file, enabled these daily time series of flow data to be used in the model to better represent these flows for use in the water resource accounts.

Similarly, in the *ACRU* 3 version, controlled flow releases from dams to downstream catchments would be represented using the QNORM variable containing 12 month-of-year flow rate values. Flow releases from Midmar Dam are mostly to provide for environmental requirements downstream. However, for Albert Falls Dam flow releases are made to supply Nagle Dam downstream, from which water is pumped by Umgeni Water to supply urban areas within the eThekwini metropolitan area. Although these flow releases do not typically vary greatly from day to day, it was still useful to be able to represent daily flow releases

estimated from measured daily flow rates at weirs almost immediately downstream of both of these dams.

5.4.6 Water Resource Accounts

As stated in the rationale for restructuring the *ACRU* model, discussed in Section 2.2, one of the main reasons for restructuring the *ACRU* model was to make it more extensible. The restructured model is more extensible due to the object-oriented design, especially the more explicit representation of physical components and hydrological processes as Component and Process classes. This extensibility was demonstrated through the addition of several new modules to the *ACRU* 2000 version, as discussed in Section 3.5.2. The extensibility of the *ACRU* 5 version was demonstrated through the development of the *Accounting* module used to compile catchment-scale water resource accounts, which is described briefly in Section 3.5.2 and in more detail in Clark (2015b).

During an *ACRU* simulation the *Accounting* module stores simulated values required for the accounts. At the end of the simulation these stored values are used to generate water resource accounts for each subcatchment for each month. Using information stored in the subcatchment and catchment Component objects it is possible to aggregate the subcatchment accounts, both spatially and temporally, to create annual water resource accounts for a range of different catchment scales. The new model structure, not only made it easier to develop the new *Accounting* module, but also made it possible to model the water resource systems in suitable detail to provide the data and information required to generate the water resource accounts.

Annual water accounts were generated for eight hydrological years from 2008-2009 to 2015-2016. Three of these accounts are discussed and compared in this section: (i) for the highest rainfall year 2010-2011, shown in Figure 5-5, (ii) for the lowest rainfall year 2011-2012, shown in Figure 5-6, and (iii) for the recent drought year 2015-2016 shown in Figure 5-7. It was interesting that for these eight simulated years the lowest rainfall year followed the highest rainfall year. These water accounts are in the form of the modified WA+ Resource Base Sheet described in Clark (2015b). A key to the individual items of the accounts, with a brief description of each item, can be found in Appendix 8.11. These water accounts should be regarded as being for illustrative purposes only, as the accuracy of the modelled flows was not good for the reasons discussed in the verification described in Appendix 8.10.

Inflows to the catchment are shown on the left-hand side of the Resource Base Sheet. In 2010-2011 the precipitation is approximately 37% greater than in 2011-2012 and 30% greater than in 2015-2016. There are no surface water inflows to the upper uMngeni as it is at the top of the uMngeni Catchment. Groundwater flows between catchments are not modelled in *ACRU*, as all baseflow is assumed to contribute to streamflow within the catchment in which it is generated. In 2010-2011, the inter-catchment transfer from the neighbouring Mooi River Catchment is almost twice that in 2011-2012, which may be due to availability of water at Mearns Weir. In 2015-2016 the inter-catchment transfer is approximately three times that in 2011-2012, with increased availability of stored water with the building of Spring Grove Dam and greater need as a result of increased population and thus water demand from Midmar Dam.

After inflows, changes in water storage within the catchment are accounted for. In 2010-2011 the change in surface water ($\Delta S_{f\ SW}$) is negative indicating that some of the gross inflow to the catchment was used to increase the water stored in dams in the catchment during the accounting period. In contrast, during the following year (2011-2012) the change in surface water is positive indicating that water stored in the dams made a net contribution to flows out of the catchment and thus stored water decreased. In 2015-2016 there was a small increase in surface water storage despite the drought, but this was only possible due to inflows from the inter-catchment transfer. In both 2010-2011 and 2011-2012 the change in the soil moisture store ($\Delta S_{f\ SoilM}$) is similar, and the negative values indicate an increase in soil moisture. The change in the soil moisture store is highly dependent on rainfall and total evaporation in the days preceding the start and end of an accounting period. As the soil moisture is calculated of a wide areal extent, over almost the whole area of a catchment, the changes in volumes of water in the catchment in soil moisture store can in some circumstances have a significant effect on the water account. In 2010-2011 the change in the groundwater store ($\Delta S_{f\ SW}$) is negative indicating that there was some recharge of the store, in contrast to the drier 2011-2012 and 2015-2016 where drawdown of the store occurred.

Total evaporation within a catchment is usually the main form of water consumption. Unsurprisingly the Landscape ET section shows total evaporation having a similar trend to the precipitation. The accounts do not indicate evaporative potential during the accounting period, but it is expected that even in wet years total evaporation is at times limited by water availability. Total evaporation from cultivated areas and areas with natural land cover is the greatest, though evaporation from water bodies is also significant. The values of Incremental ET for the cultivated areas are surprisingly small compared to the Landscape

ET values. The reasons for this are that: (i) although there is a substantial area of irrigated agriculture in the upper part of the catchment, it is still a relatively small portion of the whole catchment area, (ii) irrigation of the annual crops only takes place during the growing season, and (iii) the catchment has a relatively high rainfall and thus irrigation just supplements the rainfall. In the drier 2011-2012 and 2015-2016 years irrigation and thus Incremental ET is greater than in the wetter 2010-2011. Looking at the partitioning of total evaporation, in all the years modelled transpiration makes up the greatest proportion of total evaporation, followed by soil water evaporation and then evaporation of intercepted water. Evaporation from open water surfaces is the smallest proportion, but that is expected as the area of open water is small compared to the total catchment area.

Outflows from the catchment are shown on the lower right-hand side of the Resource Base Sheet. Outflows of surface water due to downstream releases from Albert Falls Dam were similar in 2010-2011 and 2011-2012, but less in 2015-2016. Transfers out of the catchment from Midmar Dam for urban use increased each year, except in 2015-2016 when water restrictions were put in place to reduce water use during the drought. It was interesting to note that in 2015-2016 the out transfers were almost equivalent to the in transfers from the inter-catchment transfer, highlighting the importance of this transfer.

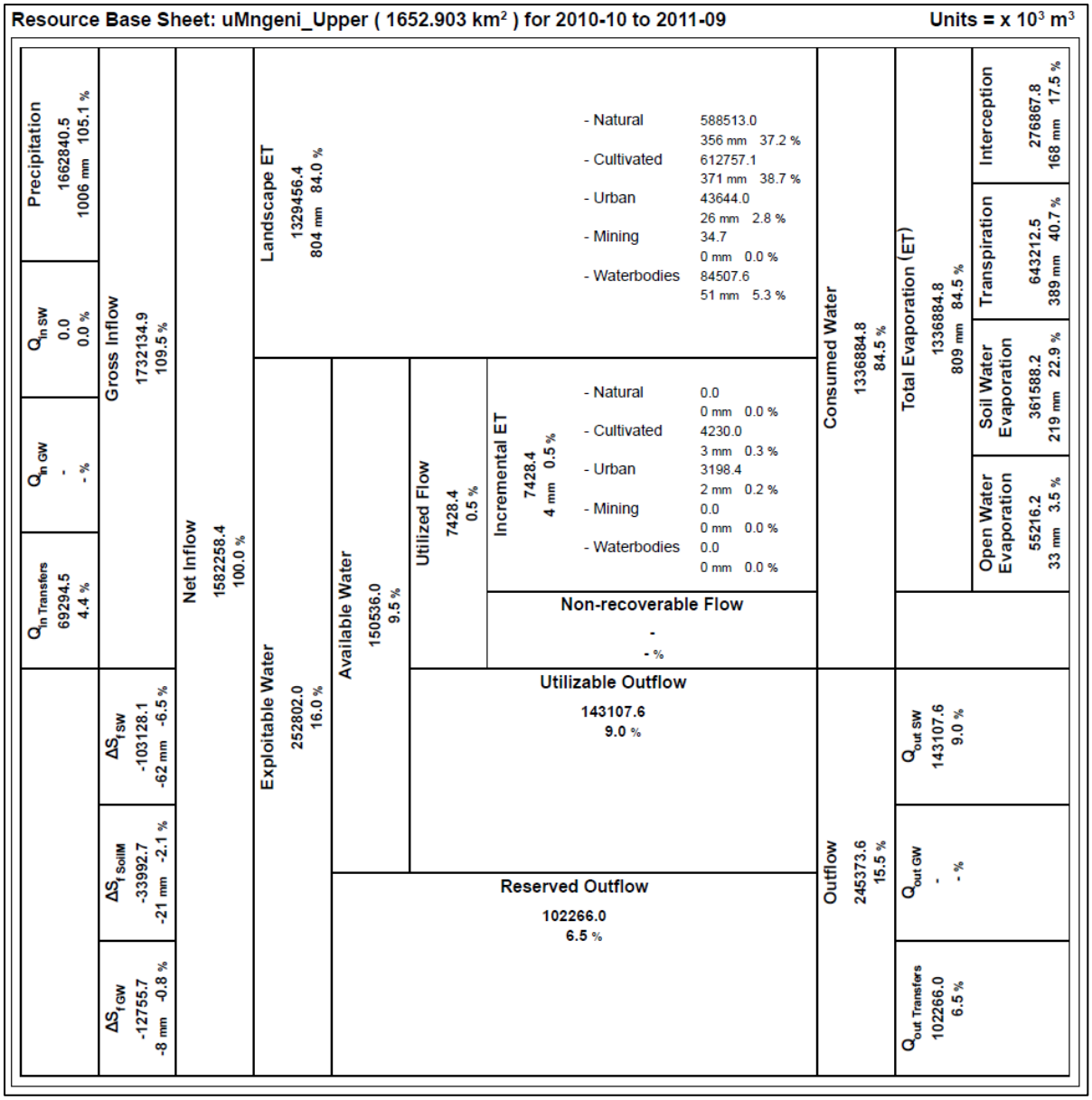

Figure 5-5 Modified Resource Base Sheet for the upper uMngeni for 2010-2011

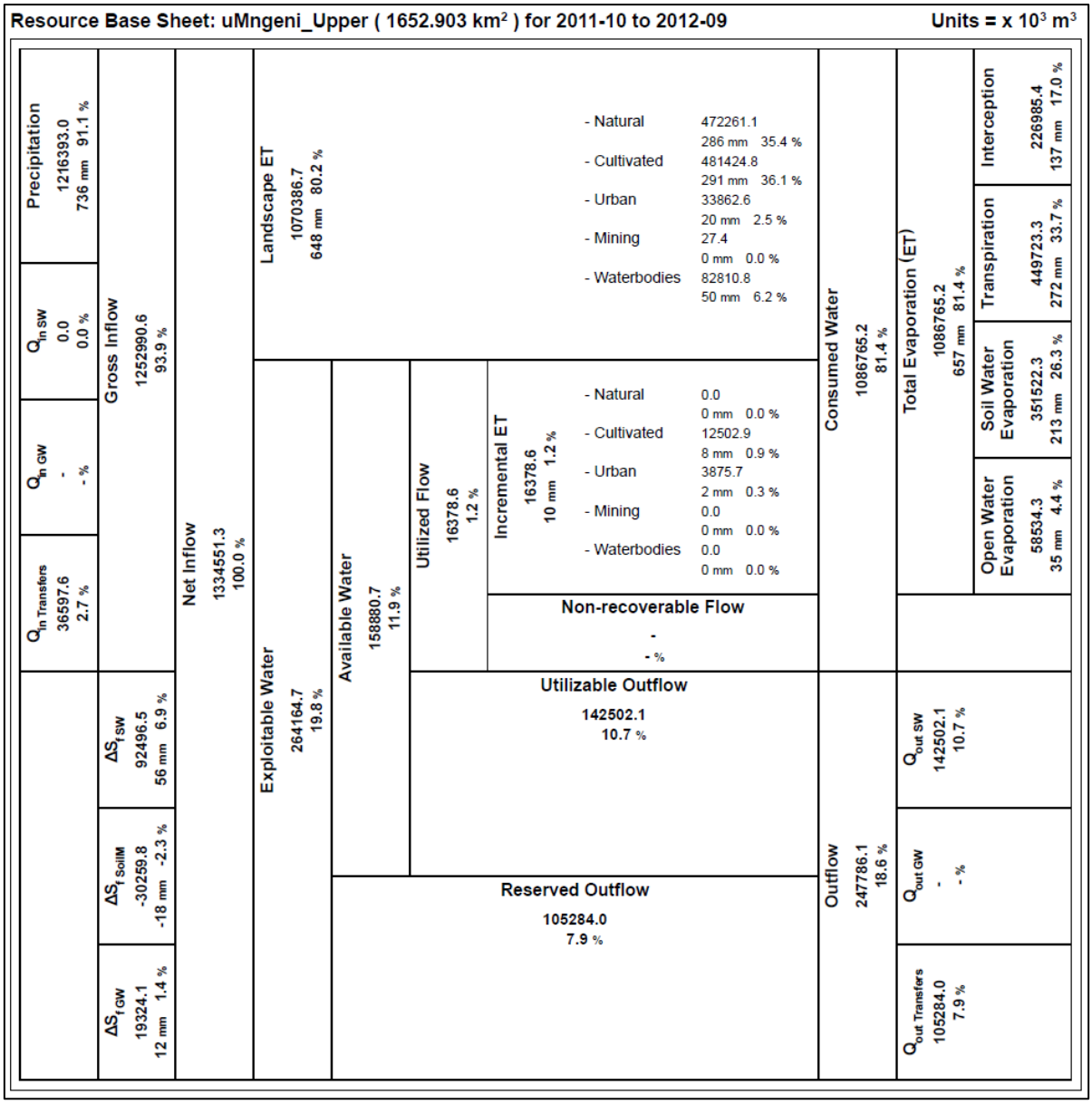

Figure 5-6 Modified Resource Base Sheet for the upper uMngeni for 2011-2012

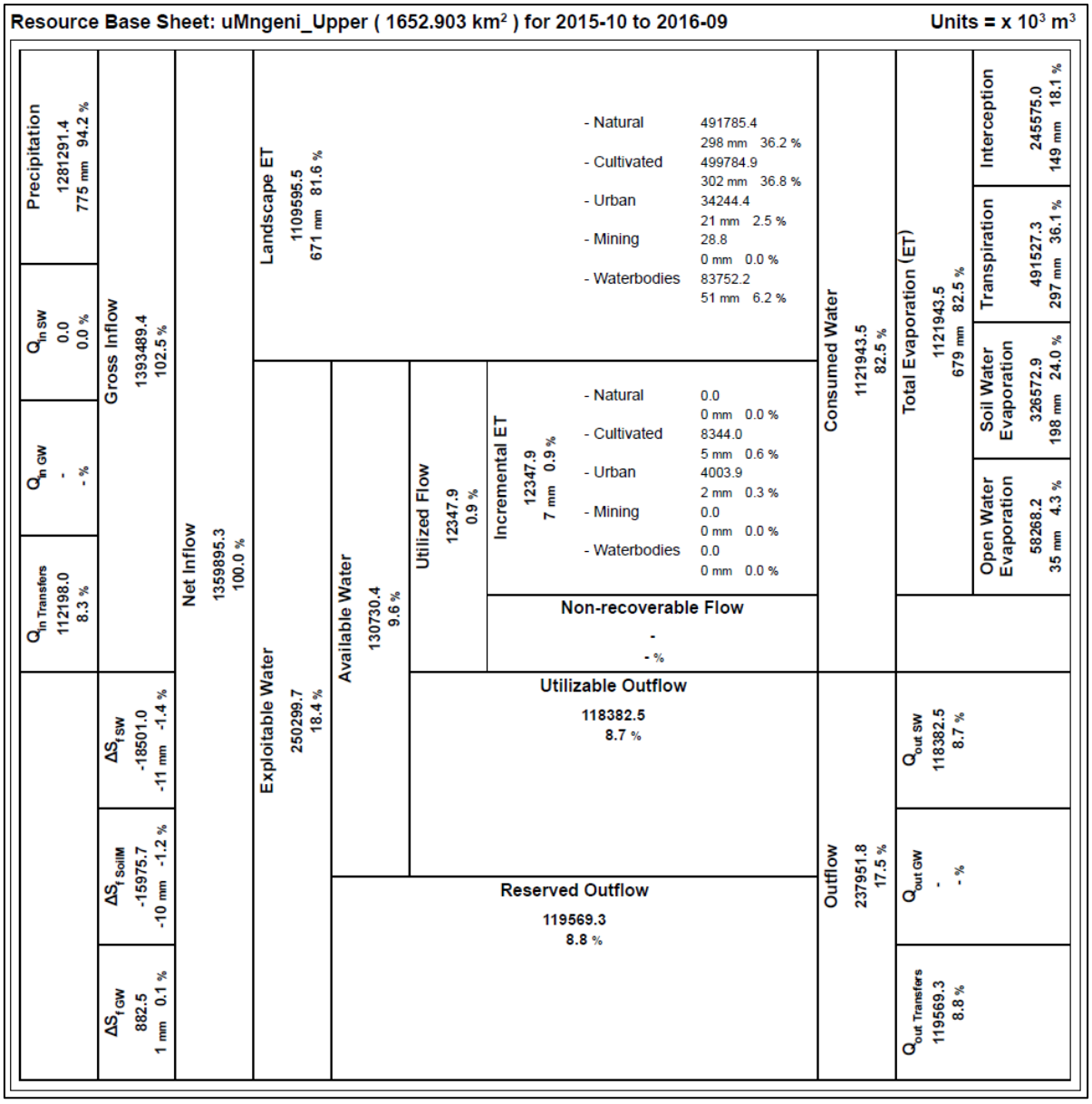

Figure 5-7 Modified Resource Base Sheet for the upper uMngeni for 2015-2016

5.5 Application of the Linked *ACRU* and eWater Source Models

The need for the routing of flows in river reaches and dams to improve the simulation of observed flows identified, at the end of Section 5.3, was addressed by linking *ACRU* to eWater Source using OpenMI, as described in Section 4.7, such that *ACRU* acts as a rainfall-runoff model and eWater Source as a river network and water management model. The role of eWater Source was to route flows downstream, provide natural flows to wetland areas modelled in *ACRU* and to distribute water to urban and irrigation users. *ACRU* is a daily timestep model and the flow reaches in the upper uMngeni are relatively short, it was thus necessary to run eWater Source at an hourly timestep and disaggregate the daily runoff

values from *ACRU* to hourly values. Two daily runoff disaggregation methods were used: (i) simply dividing the daily runoff into 24 equal hourly values, and (ii) disaggregation of the event stormflow volume using the SCS unit hydrograph (UH) approach (Schmidt and Schulze, 1987) that is used in the flow routing module of *ACRU* 3 and *ACRU* 2000. In this application of OpenMI, the *ACRU* and eWater Source linkable components were configured such that they exchanged data values using one-to-one pairs of model components and variables, each with a unique ID, rather than using the spatial characteristics of the modelled components.

Examples of the results of the linked model simulations are shown in Figure 5-8 (U2H007), Figure 5-9 (U2H013) and Figure 5-10 (U2H006) for a three month period in the 2008/2009 summer rainfall season. The verification of simulated flows, mentioned in Section 5.3, showed substantial over or underestimations in the magnitude of the flow in some events, most likely due to inaccurate rainfall estimates. The main purpose of this section was thus to show the effect of flow routing on both the magnitude and timing of flows, thus demonstrating and validating the developed ability to link the models for different domains. The hourly streamflow volumes output from eWater Source were aggregated to daily volumes and normalised to depths over the contributing catchment. The linked model simulation using the simple runoff disaggregation method resulted in substantial attenuation of the peak daily flows following a rainfall event. At a daily level there was no noticeable lagging of flows with the simple runoff disaggregation method. As anticipated, the SCS UH disaggregation method often resulted in higher daily runoff rates being calculated, but these peak daily flows following a rainfall event are lagged by a day, corresponding better to the timing of the measured flows following a rainfall event. These results highlighted the sensitivity of the simulated streamflows to: (i) the temporal distribution of rainfall and runoff within a day, and (ii) to a lesser extent, in this case study, to the lag and attenuation of flows down river reaches. More importantly in the context of this study, the results demonstrated that the linking of the *ACRU* and eWater Source models using OpenMI worked. However, the results for this short simulation period seem to indicate that there may be a mass balance error for the linked models when using the simple runoff disaggregation method - with the linked model streamflows always being less than the modelled streamflows from *ACRU* alone. This was more apparent for the Lions River and Mpendle Catchments than for the Karkloof catchment. An initial investigation into this error found that the error was not related to the units of measure when passing runoff between the models in the OpenMI connections. The individual model configurations were the same for both runoff disaggregation methods. It is unlikely that there is a mass balance error in the individual models, thus it would seem that the error is related to the model linking, and more

specifically to the dissagregation of daily to hourly runoff. With the SCS UH dissaggregation method, the runoff values are passed through an OpenMI *AdaptedOutput* in which the dissagregation takes place. In the simple runoff disaggregation method, the daily runoff values are passed to eWaterSource with the dissagregation to hourly values being done internally within eWater Source. This should be the starting point for further investigation of the problem. The cause of such mass balance errors can be difficult to pinpoint when passing values between models through two model wrappers and the OpenMI linkage mechanism.

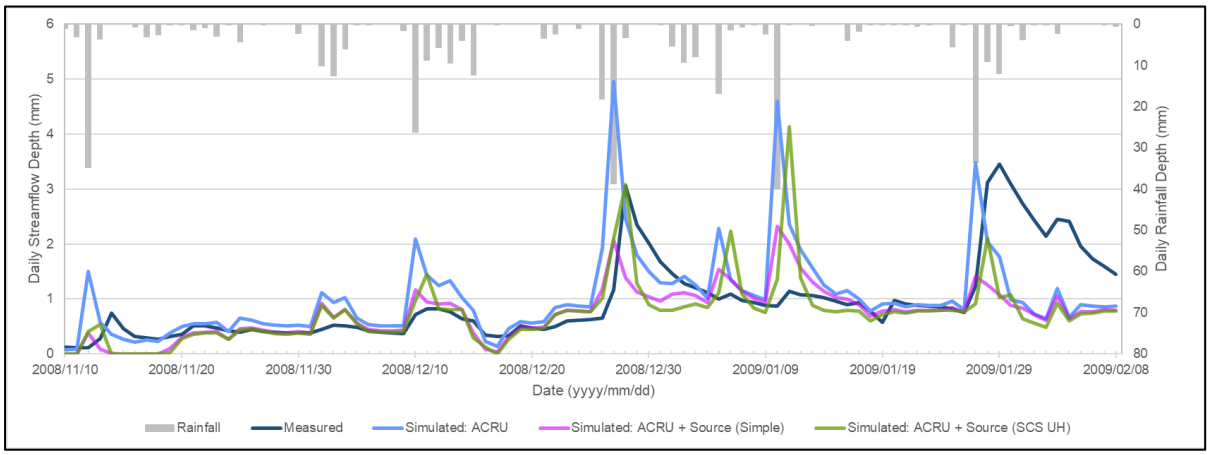

Figure 5-8 Example of daily rainfall and streamflow at gauge U2H007 (Lions River)

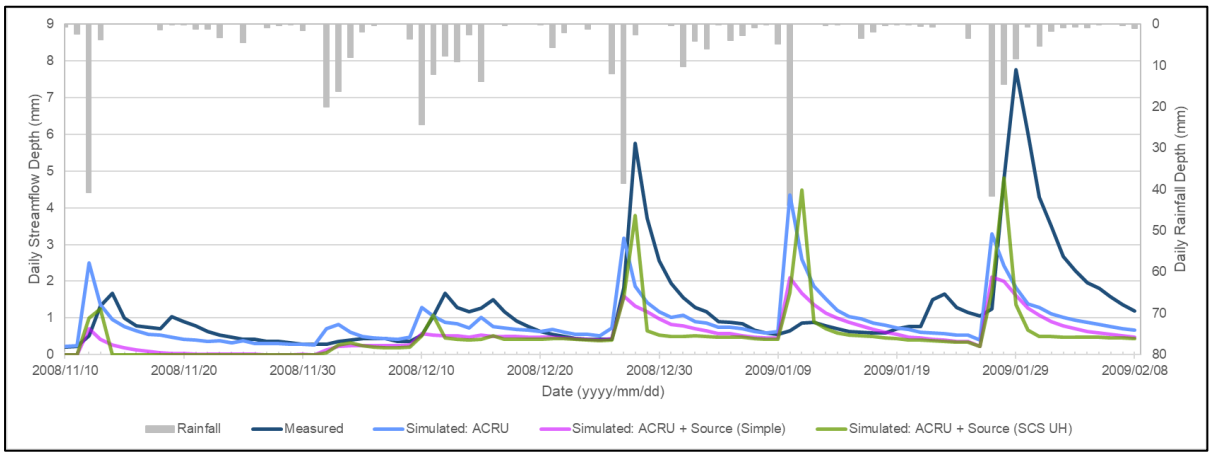

Figure 5-9 Example of daily rainfall and streamflow at gauge U2H013 (Mpendle)

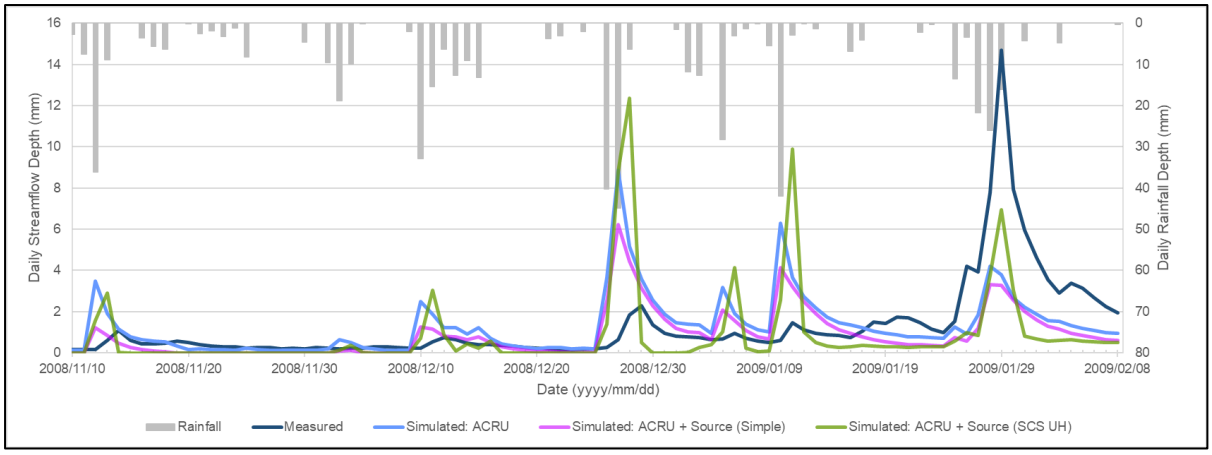

Figure 5-10 Example of daily rainfall and streamflow at gauge U2H006 (Karkloof)

The effect of the two runoff disaggregation methods to some extent masks the effect of the flow routing. To show the effect of the flow routing, the hourly simulated flows at the upstream and the downstream ends of the ungauged 17 km long main river reach in the Lions River_11 subcatchment are shown for a two week period in Figure 5-11, for the SCS UH runoff disaggregation method. The lag and attenuation of the streamflow hydrographs for the two big rainfall events on the 15th and the 22nd of February 2009 can be seen in Figure 5-11, even for this relatively short river reach. As anticipated, the lag time between the upstream and downstream peaks is in the order of 3 to 4 hours for these two events.

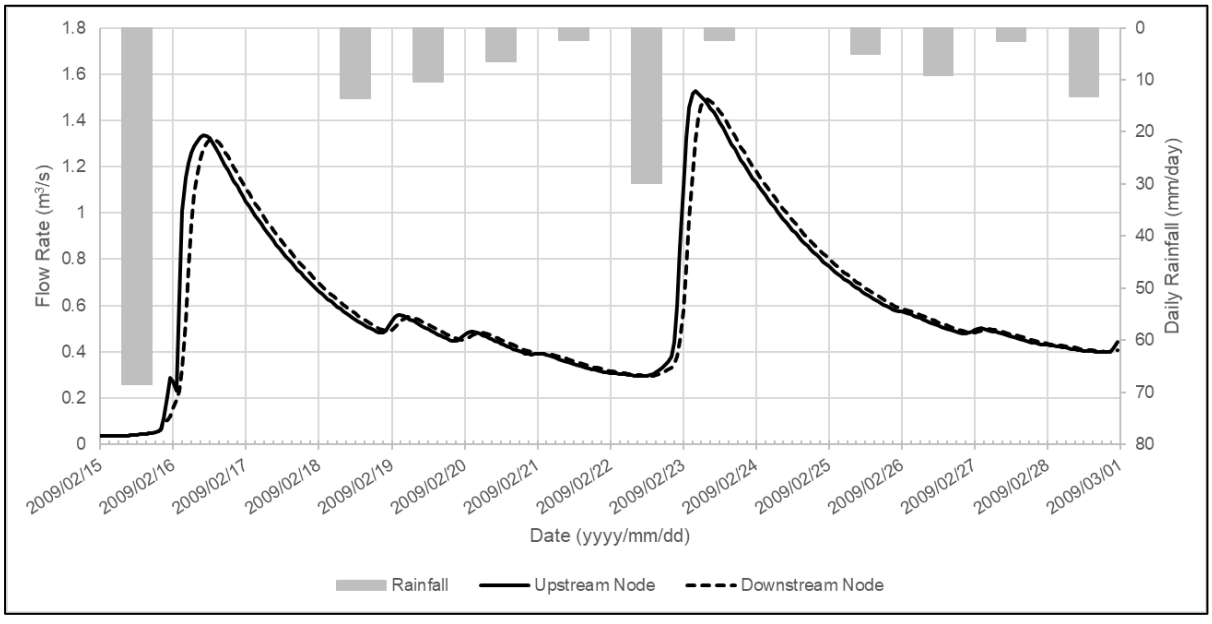

Figure 5-11 Hydrographs upstream and downstream of main river reach in the Lions River_12 subcatchment

Separate simulations were run for each of the Lions River, Mpendle and Karkloof catchments using the linked *ACRU* and eWater Source models, due to the slow execution of the simulation for the whole upper uMngeni Catchment for the full 2007 to 2016 simulation period. This problem needs to be investigated further, but was not entirely unexpected considering that detailed configurations of two model engines were being run simultaneously with the OpenMI linkable components transferring data between them at each timestep. In each of the Lions River, Mpendle and Karkloof catchments the execution time for the linked model was approximately two hours for an 8 month simulation period, with *ACRU* being run at a daily timestep and eWater Source at an hourly timestep. This is in the order of 100 times longer than the standalone *ACRU* simulations at a daily timestep for the same time period, without flow routing. These long execution times for the linked models would be impractical for (i) bigger catchment areas, configured at the same level of detail, (ii) when running the models for longer time periods, or (iii) when a large number of model runs are required. Running eWater Source at an hourly timestep is one reason for the longer execution time, given the increased number of timesteps to be evaluated in the simulation period, relative to a daily timestep. The cause of the long execution times was not investigated further as it was out of the scope of the study, given the complexities of such an investigation using software profiling tools, as was performed by Buahin and Horsburgh (2015). No references could be found in literature related to any problems with slow execution times by the eWater Source model, but as with the *ACRU* model it is expected that execution time would increase with increasing size and complexity of model configurations. Some reasons for the long execution times related to OpenMI, as discussed by Buahin and Horsburgh (2015), are listed briefly in Section 4.8. In addition to these OpenMI related causes, some potential areas for further investigation include the following:

- The compilation of *ACRU* Java code to .Net using IKVM.
- The IO and unit of measure conversion costs related to the OpenMI wrapper for *ACRU*.
- The IO, unit of measure conversion and runoff disaggregation costs related to the OpenMI wrapper for eWater Source.
- The use of the eWaterSource *GetInputMetaParameters* and *GetOutputMetaParameters* methods of the *SimulationHandler* class, and also use of the eWater Source *Functions* and *ModelledVariables* in the the OpenMI wrapper for eWater Source.
- The configuration of the linked *ACRU* and eWater Source models, which includes a large number of connections.

Both *ACRU* and eWater Source internally keep track of the mass balance of water in the systems being modelled and report any imbalances. However, it is more difficult to keep track of the combined mass balance of the linked models. Errors in the combined mass balance could easily occur due to differences in the units of measure used in the separate models, for the data passed between them using OpenMI, if incorrectly configured. During verification of the data exchanged between the two models using OpenMI, one discrepancy was found relating to the modelling of wetlands in *ACRU* and the quantity of water passed from eWater Source to the topsoil horizon in the *ACRU* wetlands. When *ACRU* is run on its own the soil moisture deficit in the topsoil horizon of a wetland is calculated after total evaporation and runoff have been calculated for the day, but before infiltration of effective rainfall. However, when the linked models are run the soil moisture deficit in the topsoil horizon of a wetland is calculated at the end of daily timestep and sent to eWater Source, that is, after infiltration of effective rainfall. This discrepancy in the timing of the calculation of the soil water deficit resulted in some differences in the modelled quantities of water being transferred from the river system in *ACRU* or eWater Source. In this case the discrepancy was not due to an apparent mass balance error, but could be explained due to the time of evaluation of a modelled state variable. This example, serves to highlight the need for the modeller responsible for linking the models to have a sound understanding of both models. Although OpenMI makes it possible to link two models, as has been demonstrated, the configuration of the linkages is not necessarily straightforward.

6 DISCUSSION AND CONCLUSIONS

Water is key to the health of people, health of ecosystems and economic prosperity in South Africa. Water resources are becoming limited in many catchments in South Africa, especially those in which large cities are located. Sound water policy and good management in accordance with South Africa's National Water Act (NWA, 1998) is required to ensure equitable and sustainable use of these water resources. South Africa is at a point where there are limited economically feasible options for development of new water storage and inter-catchment transfer infrastructure, and thus a paradigm shift is required towards demand management. An IWRM approach to water resources management will be key, thus there needs to be a move away from considering components of the water resource system in isolation towards a whole system view. Water accounting has a role to play in water resources management through quantifying water availability and use, improving understanding of water resource systems, assisting in quantifying socio-economic impacts of water management decisions and as a means of communication between stakeholders. In recent years there have been advances in some technologies that should greatly assist in quantification, understanding, prediction and optimisation of water resource availability and use. One example of such technological advancements is in remote sensing of meteorological and hydrological parameters, with higher resolution sensors and better algorithms. This study has focussed on investigating the use of some technological advances in computer modelling of water resource systems to improve the flexibility and suitability of the *ACRU* agrohydrological model as a tool for integrated modelling of water resource systems.

6.1 Summary of Study

The *ACRU* model is an existing and well established physical conceptual hydrological model that was developed in South Africa, but has been applied locally and internationally in numerous studies. The aim was to build on the existing core hydrological process functionality of the model by developing it further to provide improved representation of water resource system complexity and enabling integration with other models for use in multi-disciplinary assessments. The specific objectives for this research study were thus to:

(i) Restructure *ACRU* to make it more flexible, and thus to enable: (a) more realistic representation of the physical components of complex water resource systems, (b) better representation of engineered flows between catchments, and (c) options for

representation of hydrological processes in varying degrees of detail depending on availability of data.

(ii) Restructure *ACRU* to make it more extensible, to facilitate easier inclusion of new functionality including: (a) improved representations of hydrological processes, and (b) new analysis tools, such as a module for compiling water resource accounts.

(iii) Restructure *ACRU* to make it easier to include: (a) new data sources and formats, and (b) better handing of time series data.

(iv) Develop a means of linking *ACRU* with other models to facilitate integrated modelling studies.

6.1.1 Object-Oriented Restructuring for Greater Flexibility

The *ACRU* model was restructured from procedural FORTRAN 77 code to object-oriented Java code, resulting in the new *ACRU* 5 version of the model described in this study. Object-oriented programming techniques together with a revised design of the model structure, were used to make the model more flexible so that it would be easier to represent heterogeneous land cover/uses within catchments and engineered water transfers within and between catchments in more detail and thus to better represent the complexity of operational catchments. An XML-based model input file structure, defined in the *ModelData* schema, was developed to complement the object-oriented structure of the *ACRU* model engine.

More realistic representation of the physical components of complex water resource systems and engineered flows between catchments was achieved through: (i) the more explicit representation of these physical components as *ACRU* Component objects, and (ii) enabling relationships between Component objects to be specified. In addition better representation of engineered flows between catchments was achieved through a more flexible approach to the execution order of *ACRU* Process objects which enabled parallel processing such that each process on each Component object is executed each day before continuing to the next day. The more explicit representation of hydrological processes as Process objects enables simplified and more detailed representations of hydrological processes to be more easily interchanged, depending on availability of data. The *ACRU* Resource objects, used to represent different types of modelled resources such as water, sediment and nutrients, provide a uniform and robust means of recording the resources contained within or owned by each Component object.

6.1.2 Object-Oriented Restructuring for Greater Extensibility

A secondary objective of the restructuring was to make the model more extensible to facilitate the inclusion of new modules to better address the requirements of modelling for IWRM. It is important to note that the restructuring of *ACRU* was not intended to change any of the existing hydrological processes algorithms, though a few minor corrections and additions were made to the algorithms. Object-oriented programming techniques together with a revised object-oriented design of the model structure have made the model more easily extensible. The more explicit representation of hydrological processes as individual *ACRU* Process objects, enables improved or different representations of hydrological processes to be included in the model by simply interchanging the Process objects to be executed without changing any existing code in the model. The system of Component, Resource, Data and Process objects provides a foundational structure that can be used to build new modules and analysis tools as extensions to the existing *ACRU* functionality. This foundational model structure is complemented by the XML-based model input file structure which enables new Component types and the parameters and variables describing them to be included in the *ACRU* model engine without any changes to the code, except for the new Process classes in which they are utilised. The flexibility and extensibility of the restructured *ACRU* model was demonstrated by adding a new module to compile catchment-scale water resource accounts using the Water Accounting Plus (WA+) water accounting framework (Karimi *et al.*, 2013).

6.1.3 Object-Oriented Restructuring for Improved Data Handling

The object-oriented restructuring of the *ACRU* model also resulted in improved data handling through: (i) a common interface named *IDataReaderWriter* in the *ModelDataAccess* library, which makes it easier to include support for different model input and output data formats in *ACRU* without needing to add additional code to the *ACRU* model engine, (ii) improving the data handling and storage structures, especially for time series data, within the *ACRU* model engine using *ACRU* Data objects. The XML-based model input file structure was designed such that data values could be stored directly in the XML files or in a separate referenced data file. The data input and output handling was demonstrated through the development of a data reader/writer enabling implementation of *ACRU* as a model in the Delft-FEWS software. The XML-based model configuration file structure, specified in the *ModelConfiguration* schema, provides useful information that can be used in developing graphical user interfaces for configuring model setups.

6.1.4 Model Integration Using OpenMI

It was recognised that different models have different purposes and strengths in different domains, and further that it may not be wise, practical or economically feasible to build all the modelling functionality required for IWRM into a single model, such as *ACRU*, especially when there are many existing models, developed by experts in specific water resource domains, which could provide the required functionality. Thus, despite the improved extensibility of the *ACRU* model, methods of linking *ACRU* with other existing models were investigated. The OpenMI 2.0 model interface standard was implemented for *ACRU*, enabling integration with other OpenMI 2.0 compliant models. The OpenMI 2.0 model interface standard was also implemented for the eWater Source river network model to demonstrate linking *ACRU* to another OpenMI compliant model. Model integration using OpenMI was demonstrated by linking *ACRU* to eWater Source and running an integrated simulation in parts of the upper uMngeni Catchment. However, despite the successful implementation of an OpenMI compliant wrapper for *ACRU*, linking to another model is not straight-forward and requires a sound understanding of both models. This understanding is necessary to ensure the correct ordering of hydrological processes, that the variables being linked are compatible, and that the necessary unit of measure conversions occur, especially when the models are being run at different timesteps.

6.2 Conclusions

The *ACRU* model is an important repository of South African water research and knowledge. However, as can happen with legacy models, a point was reached where the model structure was hindering the further development of new subroutines representing hydrological processes and to make use of new more detailed datasets. This study has demonstrated that it is possible to leverage the valuable knowledge already existing in the *ACRU* model through restructuring the model using object-oriented design and programming techniques and the implementation of a model integration interface. The result is the *ACRU* 5 version of the model which is capable of more realistically representing the inherent complexity of water resources systems and can be linked with other models, designed for specific domains within the water resource system, to provide integrated holistic assessment to inform water management decisions. The *ACRU* model is now better suited to representing the complexity of water resources systems as a result of: (i) the increased flexibility provided for the configuration of catchments and the engineered flows between them, (ii) the increased flexibility provided for the order of execution of hydrological processes, and (iii) the improved extensibility of the model which will enable new

representations of hydrological processes to be easily incorporated as understanding of complex hydrological processes improves. The *ACRU* model is now better suited to support IWRM by integrating additional functionality through either: (i) extending the model, or (ii) use of the OpenMI interface to link to other models to assess, for example, the integrated water quantity, water quality, economic and social impacts of water management decisions. The *ACRU* model was previously typically used in a water resources planning context, but is now better suited for use in an operational modelling context due to the improved representation of state variables, improved time series data handling and the facility to hot-start the model. The successful restructuring of the *ACRU* model required more than just a new object-oriented design for the model engine, the design of an XML-based plus model input file structure was crucial to being able to effectively configure the model.

6.3 Summary of Contributions

This study has resulted in the development of a restructured version of the *ACRU* agrohydrological model that is more suitable than previous versions as a tool to support IWRM for both water resource planning and operations in South Africa. The design and implementation of the restructured model has also provided a unique conceptual structure and code implementation that could be utilised as a foundation to restructure other legacy models or to develop new models. The restructuring of the *ACRU* model was itself a process of learning over a period of time, and it is hoped that this study will encourage and assist other model developers in the restructuring of their legacy models.

The new and unique contributions that resulted from this study include the following:

- Design and development of a new and unique object-oriented structure for the *ACRU* 5 model engine in the Java programming language (Section 3.2), which enables more realistic model representation of the complexity of water resource systems, as described in Section 5.4.1, 5.4.2, 5.4.3, 5.4.4 and 5.4.5.
- Design and development of new model input and model configuration files using XML (Section 3.3), which complement the new object-oriented structure for the *ACRU* 5 model engine. Without these XML-based files it would not have been possible to utilise the new object-oriented structure of the model to its full potential.
- Design and development of an OpenMI 2.0 compliant wrapper for the *ACRU* model (Section 4.4), thus facilitating integration with models from different domains, thereby extending the scope of application of the *ACRU* model, especially as a tool for IWRM.

- The linking of the *ACRU* and eWater Source models using OpenMI, as demonstrated in Section 5.5.

Other developments that were not a core part of the research study but which are closely related to the application of the restructured *ACRU* model include:

- Development of a new methodology for configuring the restructured object-oriented *ACRU* model for the purpose of generating catchment-scale water resource accounts, as detailed in Clark (2015d) and in Appendix 8.9.
- Development of a water accounting module for *ACRU* based on WA+ (Clark (2015b) , Section 3.4, Appendix 8.8), which demonstrated the flexibility and extensibility of the restructured model, and was applied in Section 5.4.6.

6.4 Recommendations for Further Research and Development

Recommendations for potential further development related to restructuring of the *ACRU* 5 version of the model include the following:

- The flow routing functionality included in the *ACRU* 3 and *ACRU* 2000 versions, if it is to be retained, will need to be revised for use in the *ACRU* 5 version in which the improved handling of time series data is expected to make the implementation of flow routing easier.
- Investigate the feasibility of developing an algorithm to determine where parallel processing of catchments is required and thus try to optimise performance by reducing the quantity of catchment data loaded in memory at one time.
- Investigate the feasibility of developing an algorithm to determine the computation order of Process objects, using information in the *PProcessDataItem* and *PProcessResourceItem* objects, as a more flexible alternative to the hardcoded rules currently used in the *AAcruXmlProcesses* class. This would prevent the occasional problems related to the computation order of Components and Processes, discussed at the end of Section 3.2.2.6.
- Investigate modifying the object-oriented design of *ACRU* and optimising the Java model code to reduce model execution time by improving computational efficiency. In the *ACRU* 5 version Component objects are run sequentially and for each Component a list of Process objects is run sequentially, as shown in Figure 2-4. This makes some sense for the Components forming the river flow network where one reach must be modelled before the next reach downstream can be modelled. However, there is

scope for independent HRU Components to be run simultaneously in different threads on different processor cores.

Recommendations for potential further development of the XML model input files include the following:

- Investigate the use of a table-based format for storing model input data values, to be used in conjunction with the XML model input file, to make it easier for model users to insert and edit model input data values.
- Investigate reducing the size of the XML model input files by using shorter XML element names (tags) and condensing data values into standard string representations in place of using tags such as <*val*> and <*rec*>.

Recommendations for further research related to the application of *ACRU* as an OpenMI linkable component include the following:

- Investigate the linking of models on the basis of the spatial entities in each model, rather than the ID-based linkages used in this study.
- Investigate alternative methods of compiling the *ACRU* Java code to .Net, as a replacement for the IKVM software.
- Investigate the impact of OpenMI links between *ACRU* and other models on the model execution time of the linked models and whether this this impact can be reduced.

6.5 Lessons Learnt

The restructuring of the *ACRU* model using an object-oriented approach in Java was successful in that the main design objectives were achieved. However, some lessons learnt in the process are related in this section for consideration by future studies embarking on a similar model restructuring or model linking exercise.

6.5.1 Conceptual Design and Computational Efficiency of *ACRU*

The object-oriented design of the restructured *ACRU* model, and the concept of Component, Resource, Data and Process objects, has resulted in a better conceptual foundation for the model code which will make the model code easier to understand and extend. However, in retrospect, this better conceptual foundation appears to have come at the cost of some computational efficiency. The more flexible object-oriented model structure enables: (i) complex water resource systems to be configured in greater detail, and (ii) modelled

components in different subcatchments to be able to exchange data values on a timestep-by-timestep basis, termed “parallel processing” in this study. However, there were some trade-offs in the computational efficiency of the restructured model related to an increase in the model execution time partly as a result of parallel processing. Thus, there is scope for the design and implementation of the restructured *ACRU* model to be optimised further. When restructuring a model, consideration needs to be given to memory usage, the time related cost of read/write operations and how to make best use of multiple processor cores.

6.5.2 Application of OpenMI

OpenMI was relatively easy to implement as an OpenMI linkable component wrapper around the *ACRU* model engine, and similarly for the eWater Source model, using the FluidEarth implementation of the OpenMI Standard. However, the application of OpenMI to link two or more compliant models requires expert knowledge, both of OpenMI and the linked models. Expert knowledge is required to ensure that: (i) the correct variables are linked, (ii) all necessary variables are linked, (iii) the data values passed between variables are compatible, and (iv) a valid water balance in maintained in both models when run in linked mode. Expert knowledge is required regarding how timestepping works in each model and in OpenMI to ensure that the timestepping sequence is correct. Thus, the linking of two models in not trivial. In addition, if there are a large number of linked exchange items between the two models, it will be tedious to configure these manually. The execution time of the linked models may also be greater than the sum of the individual execution times of the standalone models, thus it will be necessary to weigh the increased execution time against the disadvantages of simply running the two models in series.

6.5.3 Good Modelling Requires Good Data

To implement IWRM, water managers will require detailed, accurate and accessible data and information. Models such as *ACRU* can assist water resources management by: (i) providing estimates of water availability, use and losses, and (ii) improving understanding of water resource systems. Water resource accounts provide a means of summarising catchment water availability and use. The case study in the upper uMngeni catchment demonstrated the new functionality developed in *ACRU* in this study. However, models such as *ACRU* are only as good as the data used to configure and verify them. The biggest lesson learnt from the case study was that, despite the improvements made to *ACRU* as tool for use in water resource system modelling, the availability and quality of data inputs for use in modelling are by far the greatest obstacle to the successful application of *ACRU* and other

water resource models. Thus, investment in maintaining and extending monitoring networks and research into the development and application of new sources of data, such as remote sensing, is crucial.

7 REFERENCES

Acocks, J. 1988. *Veld Types of South Africa*. Memoirs of the Botanical Survey of South Africa No.57. 3rd Edition. Botanical Research Institute, Pretoria, South Africa.

Aduah, MS, Jewitt, GPW and Toucher, MLW. 2017. Assessing suitability of the *ACRU* hydrological model in a rainforest catchment in Ghana, West Africa. *Water Science,* 31 (2): 198-214.

Ahuja, LR, Ascough Ii, JC and David, O. 2005. Developing natural resource models using the object modeling system: feasibility and challenges. *Advances in Geosciences,* 4: 29-36.

Alfredsen, K and Saether, B. 2000. An object-oriented application framework for building water resource information and planning tools applied to the design of a flood analysis system. *Environmental Modelling & Software,* 15 (3): 215-224.

Allen, PM, Arnold, JG and Byars, BW. 1994. Downstream channel geometry for use in planning-level models. *Journal of the American Water Resources Association,* 30 (4): 663-671.

Allen, RG, Pereira, LS, Raes, D and Smith, M. 1998. Crop evapotranspiration-Guidelines for computing crop water requirements-FAO Irrigation and drainage paper 56. *FAO, Rome,* 300: 6541.

Anderson, AJ, Mahlangu, MS, Cullis, J and Swartz, S. 2008. Integrated monitoring of water allocation reform in South Africa. *Water SA,* 34 (6): 731-737.

Anonymous. 2004. *ACRUView*: A visualisation and statistical package for use with the *ACRU* agrohydrological modelling system. In: eds. Schulze, RE and Pike, A, *Development and evaluation of an installed hydrological modelling system*. WRC Report No. 1155/1/04, Chapter 5. Water Research Commission, Pretoria, South Africa.

Argent, RM. 2004. An overview of model integration for environmental application - components, frameworks and semantics. *Environmental Modelling & Software,* 19 (3): 219-234.

Argent, RM and Houghton, B. 2001. Land and water resources model integration: software engineering and beyond. *Advances in Environmental Research,* 5 (4): 351-359.

Argent, RM, Maul, C and Krämerkämper, T. 2002. Data frameworks for environmental modelling. In: eds. Rizzoli, AE and Jakeman, AJ, *Proceedings of the 1st Biennial Meeting of the International Environmental Modelling and Software Society (iEMSs)*, Lugano, Switzerland, 324-329.

Argent, RM and Rizzoli, AE. 2004. Development of multi-framework model components. In: eds. Pahl-Wostl, C, Schmidt, S, Rizzoli, AE and Jakeman, AJ, *Proceedings of the 2nd Biennial Meeting of the International Environmental Modelling and Software Society (iEMSs)*, Osnabrück, Germany, 365-370.

Argent, RM, Vertessy, RA and Watson, FGR. 2000. A framework for catchment prediction modelling. *Hydro 2000. Proceedings of the 26th National and 3rd International Hydrology and Water Resources Symposium*, Perth, Australia, 706-711. The Institution of Engineers, Australia.

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.

Band, LE, Tague, CL, Brun, SE, Tenenbaum, DE and Fernandes, RA. 2000. Modelling watersheds as spatial object hierarchies: structure and dynamics. *Transactions in GIS,* 4 (3): 181-196.

Barthel, R, Götzinger, 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.

Beven, K and Kirkby, M. 1979. A physically based, variable contributing area model of basin hydrology. *Hydrological Sciences Journal,* 24 (1): 43-69.

Bian, L. 2003. The representation of the environment in the context of individual-based modeling. *Ecological Modelling,* 159 (2-3): 279-296.

Bian, L. 2007. Object-oriented representation of environmental phenomena: Is everything best represented as an object? *Annals of the Association of American Geographers,* 97 (2): 267 - 281.

Blaney, HF and Criddle, WD. 1950. *Determining water requirements in irrigated areas from climatological data.* Technical Publication 96. USDA-SCS, Washington DC, USA.

Blind, M and Gregersen, J. 2005. Towards an Open Modelling Interface (OpenMI) the HarmonIT project. *Advances in Geosciences,* 4: 69-74.

Booch, G. 1994. *Object-oriented analysis and design with applications*. Second Edition. Object Technology Series. Addison-Wesley, Boston, USA.

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. *Proceedings of the 9th IEEE International Symposium on High Performance Distributed Computing*, Pittsburgh, Pennsylvania, USA. 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, *Proceedings of the 5th Biennial Meeting of the International Environmental Modelling and Software Society (iEMSs)*, Ottawa, Canada.

Buahin, CA and Horsburgh, JS. 2015. Evaluating the simulation times and mass balance errors of component-based models: An application of OpenMI 2.0 to an urban stormwater system. *Environmental Modelling and Software,* 72: 92-109.

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 Modeling Interface (OpenMI): methodology and initial results. *Hydrology and Earth System Sciences,* 14 (3): 521-534.

Butler, AJE. 2001. *The development and evaluation of an operating rule framework for the ACRU agrohydrological modelling system*. Unpublished MSc Eng dissertation. School of Bioresources Engineering and Environmental Hydrology, University of KwaZulu-Natal, Pietermaritzburg, South Africa.

Campbell, KL, Kiker, GA and Clark, DJ. 2001. Development and testing of a nitrogen and phosphorus process model for Southern African water quality issues. ASAE Paper No. 012085. ASAE, St. Joseph, Michigan, USA.

Carr, R and Podger, G. 2012. eWater source - Australia's next generation IWRM modelling platform. *HWRS2012. Proceedings of the 34th Hydrology and Water Resources Symposium*, Sydney, Australia, 742-749. Engineers Australia.

Castronova, AM and Goodall, JL. 2010. A generic approach for developing process-level hydrologic modeling components. *Environmental Modelling & Software,* 25 (7): 819-825.

Charley, W, Pabst, A and Peters, J. 1995. *The Hydrologic Modeling System (HEC-HMS): Design and Development Issues*. TP-149. Hydrological Engineering Center, US Army Corps of Engineers, Davis, California, USA.

Clark, DJ. 2013. *ACRU* model development. In: eds. Clark, DJ and Smithers, JC, *Model integration for operational Water resources planning and management*. WRC Report No. 1951/1/12, Chapter 4. Water Research Commission, Pretoria, South Africa.

Clark, DJ. 2015a. *Development and Assessment of an Integrated Water Resources Accounting Methodology for South Africa*. WRC Report 2205/1/15. Water Research Commission (WRC), Pretoria, South Africa.

Clark, DJ. 2015b. Development of a methodology for water use quantification and accounting. In: ed. Clark, DJ, *Development and Assessment of an Integrated Water Resources Accounting Methodology for South Africa*. WRC Report No. 2205/1/15, Chapter 4. Water Research Commission (WRC), Pretoria, South Africa.

Clark, DJ. 2015c. Sabie-Sand Catchment case study. In: ed. Clark, DJ, *Development and Assessment of an Integrated Water Resources Accounting Methodology for South Africa*. WRC Report No. 2205/1/15, Chapter 6. Water Research Commission (WRC), Pretoria, South Africa.

Clark, DJ. 2015d. uMngeni Catchment case study. In: ed. Clark, DJ, *Development and Assessment of an Integrated Water Resources Accounting Methodology for South Africa*. WRC Report No. 2205/1/15, Chapter 5. Water Research Commission (WRC), Pretoria, South Africa.

Clark, DJ. 2016. *Further development and assessment of an integrated water resources accounting methodology for South Africa: Deliverable 3: Progress report - Year 1*. Unpublished report to the Water Research Commission (WRC) for Deliverable 3 of WRC Project K5/2512. Centre for Water Resources Research (CWRR), University of KwaZulu-Natal, Pietermaritzburg, South Africa.

Clark, DJ. 2017a. *Further development and assessment of an integrated water resources accounting methodology for South Africa: Deliverable 4: Annual report - Year 1*. Unpublished report to the Water Research Commission (WRC) for Deliverable 4 of WRC Project K5/2512. Centre for Water Resources Research (CWRR), University of KwaZulu-Natal, Pietermaritzburg, South Africa.

Clark, DJ. 2017b. *Further development and assessment of an integrated water resources accounting methodology for South Africa: Deliverable 5: Progress report - Year 2*. Unpublished report to the Water Research Commission (WRC) for Deliverable 5 of WRC Project K5/2512. Centre for Water Resources Research (CWRR), University of KwaZulu-Natal, Pietermaritzburg, South Africa.

Clark, DJ. 2018. Investigation of satellite remotely sensed rainfall for use in hydrological modelling in the upper uMngeni Catchment, South Africa. Paper in preparation. Centre for Water Resources Research, University of KwaZulu-Natal, Pietermarithburg, South Africa.

Clark, DJ, Bastiaanssen, WGM, Smithers, JC and Jewitt, GPW. 2015. A review of water accounting frameworks for potential application in South Africa. In: ed. Clark, DJ, *Development and Assessment of an Integrated Water Resources Accounting*

Methodology for South Africa. WRC Report No. 2205/1/15, Chapter 2. Water Research Commission (WRC), Pretoria, South Africa.

Clark, DJ, Hughes, DA, Smithers, JC, Thornton-Dibb, SLC, Lutchminarain, A and Forsyth, DA. 2012a. *Deployment, maintenance and further development of SPATSIM-HDSF: Volume 1 - SPATSIM-HDSF modelling framework*. WRC Report No. 1870/1/12. Water Research Commission, Pretoria, South Africa.

Clark, DJ, Kiker, GA and Schulze, RE. 2001. Object-oriented restructuring of the *ACRU* agrohydrological modelling system. *Proceedings of the 10th South African National Hydrology Symposium*, Pietermaritzburg, South Africa, 26-28.

Clark, DJ and Lutchminarain, A. 2013. Implementation of OpenMI for model linking. In: eds. Clark, DJ and Smithers, JC, *Model integration for operational Water resources planning and management*. WRC Report No. 1951/1/12, Chapter 5. Water Research Commission, Pretoria, South Africa.

Clark, DJ, Lutchminarain, A and Smithers, JC. 2013. Review and evaluation of model linkage mechanisms. In: eds. Clark, DJ and Smithers, JC, *Model integration for operational Water resources planning and management*. WRC Report No. 1951/1/12, Chapter 3. Water Research Commission, Pretoria, South Africa.

Clark, DJ and Smithers, JC. 2013. *Model integration for operational Water resources planning and management*. WRC Report No. 1951/1/12. Water Research Commission, Pretoria, South Africa.

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.

Clark, DJ, Smithers, JC, Thornton-Dibb, SLC and Lutchminarain, A. 2012b. *Deployment, maintenance and further development of SPATSIM-HDSF: Volume 3 - ACRU agrohydrological model*. WRC Report No. 1870/3/12. Water Research Commission, Pretoria, South Africa.

Colvin, J, Ballim, F, Chimbuya, S, Everard, M, Goss, J, Klarenberg, G, Ndlovu, S, Ncala, D and Weston, D. 2008. Building capacity for co-operative governance as a basis for integrated water resource managing in the Inkomati and Mvoti catchments, South Africa. *WaterSA,* 34 (6): 681-689.

Cook, S and Daniels, J. 1994. *Designing object systems*. Prentice-Hall, UK.

Croke, B, Andrews, F, Jakeman, AJ, Cuddy, S and Luddy, A. 2005. Redesign of the IHACRES rainfall-runoff model. *Proceedings of the 29th Hydrology and Water Resources Symposium*, Canberra, Australia. Engineers Australia.

CSIR. 2003. *Guidelines for Human Settlement Planning and Design: Volume 2*. Boutek Report No. BOU/E2001. Compiled by Boutek at the Council for Scientific and Industrial Research (CSIR) for the Department of Housing, Pretoria, South Africa.

CSIR. 2013. South African Functional Typology Population Dataset. [Dataset]. Council for Scientific and Industrial Research (CSIR), Pretoria, South Africa.

Dahmann, JS, Fujimoto, RM and Weatherly, RM. 1997. The Department of Defense High Level Architecture. *Proceedings of the 29th Winter Simulation Conference*, Atlanta, Georgia, USA, 142-149. IEEE Computer Society, Washington, DC, USA.

David, O, Ascough, J, Leavesley, G and Ahuja, L. 2010. Rethinking Modeling Framework Design: Object Modeling 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, Schneider, I and Leavesley, G. 2004. Metadata and modeling frameworks: The object modeling system example. *Transactions of the 2nd Biennial Meeting of the International Environmental Modelling and Software Society, iEMSs 2004*, Osnabrück, Germany, 439-443. Citeseer.

Davis, G. 2015. Response on the eWater forum to a query regarding whether eWater Source was OpenMI compliant. [Personal Communication]. 30/09/2015.

Deltares. 2018. Delft-FEWS Software Community. [Internet]. Available from: http://oss.deltares.nl/web/delft-fews/home. [Accessed: 18/01/2018].

Donchyts, G, Hummel, S, Vaneçek, S, Groos, J, Harper, A, Knapen, R, Gregersen, J, Schade, P, Antonello, A and Gijsbers, 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, *Proceedings of the 5th Biennial Meeting of the International Environmental Modelling and Software Society (iEMSs)*, Ottawa, Canada, 1050-1057.

DSO. 2016. List of Registered Dams in South Africa, 2016. List of Registered Dams Feb 2016.xls available from https://www.dwa.gov.za/DSO/Publications.aspx. [Dataset]. Dam Safety Office (DSO), Department of Water and Sanitation (DWS), Pretoria, South Africa.

Dutta, D, Wilson, K, Welsh, WD, Nicholls, D, Kim, S and Cetin, L. 2013. A new river system modelling tool for sustainable operational management of water resources. *Journal of Environmental Management,* 121 (Supplement C): 13-28.

DWA. 2013. National Water Resource Strategy (2nd Edition): Water for an Equitable and Sustainable Future. Department of Water Affairs. Pretoria, South Africa.

DWAF. 2004. National Water Resource Strategy. Department of Water Affairs and Forestry, Pretoria, South Africa.

DWAF and Umgeni Water. 2004. *Mooi-Mgeni Transfer Scheme Phase 2: Feasibility Study Main Report*. DWAF Report No. PB V200-00-1501. Prepared by GMKS (Pty) Ltd for Department of Water Affairs & Forestry and Umgeni Water, South Africa.

DWS. 2015. *Strategic Overview of the Water Sector in South Africa: 2015*. Directorate: Water Macro Planning, Department of Water and Sanitation (DWS), Pretoria, South Africa.

Eagleson, PS. 1983. *Some problems of parameterization of land surface heat and moisture fluxes*. Report to the Fourth Session of the Joint Scientific Committee, Venice, Italy. World Meteorological Organisation, Geneva, Switzerland.

Egenhofer, M and Frank, A. 1992. Object-oriented modeling for GIS. *Urisa Journal,* 4 (2): 3-19.

Elshorbagy, A and Ormsbee, L. 2006. Object-oriented modeling approach to surface water quality management. *Environmental Modelling & Software,* 21 (5): 689-698.

eWater CRC. 2017. eWater Source (public version 4.1.0.4337. [Software]. eWater Cooperative Research Centre (CRC), Canberra, Australia.

Ezemvelo KZN Wildlife and GeoTerraImage. 2013. KwaZulu-Natal Land Cover 2011 V1. Unpublished GIS Coverage [Clp_KZN_2011_V1_grid_w31.zip]. [Dataset]. Produced by Ezemvelo KZN Wildlife (Biodiversity Research and Assessment) and GeoTerraImage (Pty) Ltd, P. O. Box 13053, Cascades, Pietermaritzburg, 3202, South Africa.

Forbes, KA, Kienzle, SW, Coburn, CA, Byrne, JM and Rasmussen, J. 2011. Simulating the hydrological response to predicted climate change on a watershed in southern Alberta, Canada. *Climatic Change,* 105 (3): 555-576.

Foster, T, Brozovi´c, N, Butler, AP, Neale, CMU, Raes, D, Steduto, P, Fereres, E and Hsiao, TC. 2017. AquaCrop-OS: An open source version of FAO's crop water productivity model. *Agricultural Water Management,* 181: 18–22.

Frevert, D, Fulp, T, Zagona, E, Leavesley, G and Lins, H. 2006. Watershed and River Systems Management Program: Overview of Capabilities. *Journal of Irrigation and Drainage Engineering,* 132 (2): 92-97.

Frijters, J. 2017. The End of IKVM.NET. [Internet]. Available from: http://weblog.ikvm.net/. [Accessed: 15/01/2018].

Galton, A. 2004. Fields and objects in space, time, and space-time. *Spatial Cognition & Computation,* 4 (1): 39-68.

Garrote, L and Becchi, I. 1997. Object-oriented software for distributed rainfall-runoff models. *Journal of Computing in Civil Engineering,* 11 (3): 190-194.

Gärtner, H, Bergmann, A and Schmidt, J. 2001. Object-oriented modeling of data sources as a tool for the integration of heterogeneous geoscientific information. *Computers & Geosciences,* 27 (8): 975-985.

Ghashghaei, M, Bagheri, A and Morid, S. 2013. Rainfall-runoff Modeling in a Watershed Scale Using an Object Oriented Approach Based on the Concepts of System Dynamics. *Water Resources Management,* 27 (15): 5119-5141.

Gijsbers, P, Hummel, S, Vaneçek, S, Groos, J, Harper, A, Knapen, R, Gregersen, J, Schade, P, Antonello, A and Donchyts, G. 2010. From OpenMI 1.4 to 2.0. 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.

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. *Advances in Geosciences,* 4: 37-44.

Hallowes, L, Schulze, RE, Horan, MJC and Pike, A. 2004. South African National Quaternary Catchments Database: Refinements to, and links with, the *ACRU* model as a framework for installed modelling systems. In: eds. Schulze, RE and Pike, A, *Development and evaluation of an installed hydrological modelling system*. WRC Report No. 1155/1/04, Chapter 6. Water Research Commission, Pretoria, South Africa.

Hargreaves, GH and Samani, ZA. 1985. Reference crop evapotranspiration from temperature. *Transactions of the American Society of Agricultural Engineers,* 1: 96-99.

Harpham, Q, Cleverley, P and Kelly, D. 2014. The FluidEarth 2 implementation of OpenMI 2.0. *Journal of Hydroinformatics,* 16 (4): 890-906.

Harpham, Q, Lhomme, J, Parodi, A, Fiori, E, Jagers, B and Galizia, A. 2016. Using OpenMI and a model map to integrate WaterML2 and NetCDF data sources into flood

modeling of Genoa, Italy. *Journal Of The American Water Resources Association,* 52 (4):

Hofman, D. 2005. LIANA Model Integration System - architecture, user interface design and application in MOIRA DSS. *Advances in Geosciences,* 4: 9-16.

Hoheisel, A. 2002. Model coupling and integration via XML in the M3 simulation. In: eds. Rizzoli, AE and Jakeman, AJ, *Proceedings of the 1st Biennial Meeting of the International Environmental Modelling and Software Society (iEMSs)*, Lugano, Switzerland, 611–616.

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.

Javadi, S, Kiapasha, MS and Mohammadi, K. 2009. Object oriented simulation; Its application in water reservoir management and operation. *Journal of Agricultural Science and Technology,* 11 (3): 331-340.

Jones, JW, Keating, BA and Porter, CH. 2001. Approaches to modular model development. *Agricultural Systems,* 70 (2-3): 421-443.

Joyce, RJ, Janowiak, JE, Arkin, PA and Xie, P. 2004. CMORPH: A method that produces global precipitation estimates from passive microwave and infrared data at high spatial and temporal resolution. *Journal of Hydrometeorology,* 5 (3): 487-503.

Kang, K and Merwade, V. 2011. Development and application of a storage–release based distributed hydrologic model using GIS. *Journal of Hydrology,* 403 (1): 1-13.

Kang, K, Merwade, V, Chun, JA and Timlin, D. 2016. Flexibility on storage-release based distributed hydrologic modeling with object-oriented approach. *Journal of Hydrology,* 540: 17-25.

Karimi, P, Bastiaanssen, WGM and Molden, D. 2013. Water Accounting Plus (WA+) - a water accounting procedure for complex river basins based on satellite measurements. *Hydrology and Earth System Sciences,* 17 (7): 2459-2472.

Kienzle, SW. 2011. Effects of area under-estimations of sloped mountain terrain on simulated hydrological behaviour: A case study using the *ACRU* model. *Hydrological Processes,* 25: 1212-1227.

Kienzle, SW, Lorentz, SA and Schulze, RE. 1997. *Hydrology and water quality of the Mgeni Catchment*. WRC Report Number TT87/97. Water Research Commission, Water Research Commission, Pretoria, South Africa.

Kienzle, SW, Nemeth, MW, Byrne, JM and MacDonald, RJ. 2012. Simulating the hydrological impacts of climate change in the upper North Saskatchewan River basin, Alberta, Canada. *Journal of Hydrology,* 412-413: 76-89.

Kienzle, SW and Schmidt, J. 2008. Hydrological impacts of irrigated agriculture in the Manuherikia catchment, Otago, New Zealand. *Journal of Hydrology (New Zealand),* 47 (2): 67-84.

Kiker, GA. 2001. *ACRU*2000 model. In: eds. Lynch, SD and Kiker, GA, *ACRU model development and user support*. WRC Report No. 636/1/01. Chapter 4. Water Research Commission, Pretoria, South Africa.

Kiker, GA and Clark, DJ. 1999. Further development of an initial design of an object-oriented structure for the *ACRU* model. [Software]. School of Bioresources Engineering and Environmental Hydrology, University of Natal, Pietermaritzburg, South Africa.

Kiker, GA and Clark, DJ. 2001a. Development and testing of a natural vegetation, herbivore, and fire model for southern African rangeland management. ASAE Paper No. 017025. ASAE, St. Joseph, Michigan, USA.

Kiker, GA and Clark, DJ. 2001b. The development of a Java-based, object-oriented modeling system for simulation of southern African hydrology. ASAE Paper No. 012030. ASAE, St. Joseph, Michigan, USA.

Kiker, GA and Clark, DJ. 2001c. Testing and validation of a Java-based, object-oriented modeling system in the Mgeni River watershed, KwaZulu-Natal, South Africa. ASAE Paper No. 012031. ASAE, St. Joseph, Michigan, USA.

Kiker, GA, Clark, DJ, Martinez, CJ and Schulze, RE. 2006. A Java-based, object-oriented modeling system for Southern African hydrology. *Transactions of the ASABE,* 49 (5): 1419-1433.

Kiker, GA and David, O. 1998. Initial design of an object-oriented structure for the *ACRU* model. [Software]. School of Bioresources Engineering and Environmental Hydrology, University of Natal, Pietermaritzburg, South Africa.

Knapen, MJR, Verweij, P, Wien, JE 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.

Knapen, R. 2015. E-mail responding to query regarding progress with development of a Java SDK for OpenMI Version 2.0. [Personal Communication]. 18/09/2015.

Knapen, R, Janssen, S, Roosenschoon, O, Verweij, P, De Winter, W, Uiterwijk, M and Wien, J-E. 2013. Evaluating OpenMI as a model integration platform across disciplines. *Environmental Modelling & Software,* 39: 274-282.

Kokkonen, T, Jolma, A and Koivusalo, H. 2003. Interfacing environmental simulation models and databases using XML. *Environmental Modelling & Software,* 18 (5): 463-471.

Kralisch, S, Krause, P and David, O. 2005. Using the object modeling system for hydrological model development and application. *Advances in Geosciences,* 4 (1-2): 75–81.

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.

Kumar, M, Bhatt, G and Duffy, CJ. 2010. An object-oriented shared data model for GIS and distributed hydrologic models. *International Journal of Geographical Information Science,* 24 (7): 1061-1079.

Kummerow, C, Simpson, J, Thiele, O, Barnes, W, Chang, ATC, Stocker, E, Adler, RF, Hou, A, Kakar, R, Wentz, F, Ashcroft, P, Kozu, T, Hong, Y, Okamoto, K, Iguchi, T, Kuroiwa, H, Im, E, Haddad, Z, Huffman, G, Ferrier, B, Olson, WS, Zipser, E, Smith, EA, Wilheit, TT, North, G, Krishnamurti, T and Nakamura, K. 2000. The Status of the Tropical Rainfall Measuring Mission (TRMM) after Two Years in Orbit. *Journal of Applied Meteorology,* 39 (12): 1965-1982.

Kusangaya, S, Warburton Toucher, M and Archer van Garderen, E. 2017. Use of *ACRU*, a distributed hydrological model, to evaluate how errors from downscaled rainfall are propagated in simulated runoff in uMngeni Catchment, South Africa. *Hydrological Sciences Journal,* 62 (12): 1995-2011.

Lafore, R. 2002. *Object-oriented programming in C++*. 4th Edition. Sams Publishing, Indianapolis, Indiana, USA,

Lal, AMW, Van Zee, R and Belnap, M. 2005. Case study: Model to simulate regional flow in South Florida. *Journal of Hydraulic Engineering,* 131 (4): 247-258.

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.

Leone, A and Chen, DY. 2007. Implementation of an object oriented data model in an information system for water catchment management: Java JDO and Db4o Object Database. *Environmental Modelling & Software,* 22 (12): 1805-1810.

Linacre, ET. 1977. A simple formula for estimating evaporation rates in various climates, using temperature data alone. *Agricultural Meteorology,* 18: 409-424.

Lindenschmidt, KE, Hesser, FB and Rode, M. 2005. Integrating water quality models in the High Level Architecture (HLA) environment. *Advances in Geosciences,* 4: 51-56.

Liu, DF and Stewart, TJ. 2004. Object-oriented decision support system modelling for multicriteria decision making in natural resource management. *Computers & Operations Research,* 31 (7): 985-999.

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 modeling frameworks. 909-915. TR, Ahuja, LR 2009. An Exploratory Investigation on the Invasiveness of Environmental Modeling 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*. WRC Report 1156/1/04. Water Research Commission, Pretoria, South Africa.

Lynch, SD and Kiker, GA. 2001a. *ACRU model development and user support*. WRC Report No. 636/1/01. Water Research Commission, Pretoria, South Africa.

Lynch, SD and Kiker, GA. 2001b. Introduction to the *ACRU* restructuring project. In: eds. Lynch, SD and Kiker, GA, *ACRU model development and user support*. WRC Report No. 636/1/01. Chapter 1. Water Research Commission, Pretoria, South Africa.

Maaren, H and Moolman, J. 1985. *The effects of farm dams on hydrology.* Proceedings, 2nd South African National Hydrology Symposium. ACRU Report, 22, 329-337. Department of Agricultural Engineering, University of Natal, Pietermaritzurg, South Africa.

Maila, D, Crafford, J, Mathebula, V, Naidoo, N and Visser, W. 2018. *National Water Accounts For South Africa: Systems, Methods and Initial Results*. WRC Project K5/2419. Water Research Commission (WRC), Pretoria, South Africa.

Martinez, CJ, Campbell, KL, Annable, MD and Kiker, GA. 2008. An object-oriented hydrologic model for humid, shallow water-table environments. *Journal of Hydrology,* 351 (3-4): 368-381.

McCarthy, GT. 1938. The unit hydrograph and flood routing. *Conference of the North Atlantic Division, US Army Corps of Engineers*, New London, Connecticut, USA.

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.

McDonnell, J, Sivapalan, M, Vaché, K, Dunn, S, Grant, G, Haggerty, R, Hinz, C, Hooper, R, Kirchner, J and Roderick, M. 2007. Moving beyond heterogeneity and process complexity: A new vision for watershed hydrology. *Water Resources Research,* 43 (7): W07301.

Meier, KB. 1997. *Development of a spatial database for agrohydrological model applications in Southern Africa.* Unpublished MSc Eng dissertation. Department of Agricultural Engineering, University of Natal, Pietermaritzburg, South Africa.

Molden, D, Frenken, K, Barker, R, de Fraiture, C, Mati, B, Svendsen, M, Sadoff, C, Finlayson, M, Atapattu, S, Giordano, M, Inocencio, A, Lannerstad, M, Manning, N, Molle, F, Smedema, B and Vallee, D. 2007. Trends in water and agricultural development. In: ed. Molden, D, *Water for Food, Water for Life: A Comprehensive*

Assessment of Water Management in Agriculture. Earthscan, London, UK and International Water Management Institute, Colombo, Sri Lanka.

Molina, JL, Bromley, J, García-Aróstegui, JL, Sullivan, C and Benavente, J. 2010. Integrated water resources management of overexploited hydrogeological systems using Object-Oriented Bayesian Networks. *Environmental Modelling & Software,* 25 (4): 383-397.

Moore, RV and Tindall, CI. 2005. An overview of the open modelling interface and environment (the OpenMI). *Environmental Science & Policy,* 8 (3): 279-286.

Moult, NG. 2005. *The development of a catchment scale irrigation systems model for sugarcane*. Unpublished MSc Eng dissertation. School of Bioresources Engineering and Environmental Hydrology, University of KwaZulu-Natal, Pietermaritzburg, South Africa.

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, Australia.

Novella, NS and Thiaw, WM. 2012. African Rainfall Climatology Version 2 for Famine Early Warning Systems. *Journal of Applied Meteorology and Climatology,* 52 (3): 588-606.

NWA. 1998. *National Water Act, Act No. 36 of 1998*. Government Printers, Pretoria, South Africa.

OMG. 2017. *OMG® Unified Modeling Language® (OMG UML®): Version 2.5.1*. Object Management Group (OMG), Needham, Massachusetts, USA.

Open Geospatial Consortium. 2014. OGC® Open Modelling Interface Interface Standard. http://www.opengis.net/doc/IS/openmi/2.0.

OpenMI Association. 2010a. *Migrating Models for the OpenMI (Version 2.0)*. OpenMI Document Series. The OpenMI Association, Delft, Netherlands.

OpenMI Association. 2010b. *The OpenMI 'in a Nutshell' for the OpenMI (Version 2.0)*. OpenMI Document Series. The OpenMI Association, Delft, Netherlands.

OpenMI Association. 2010c. *OpenMI Standard 2 Reference for the OpenMI (Version 2.0)*. OpenMI Document Series. The OpenMI Association, Delft, Netherlands.

OpenMI Association. 2010d. *OpenMI Standard 2 Specification for the OpenMI (Version 2.0)*. OpenMI Document Series. The OpenMI Association, Delft, Netherlands.

OpenMI Association. 2010e. *Scope for the OpenMI (Version 2.0)*. OpenMI Document Series. The OpenMI Association, Delft, Netherlands.

OpenMI Association. 2010f. *What's New in OpenMI 2.0*. OpenMI Document Series. The OpenMI Association, Delft, Netherlands.

OpenMI Association. 2017. OpenMI. [Internet]. The OpenMI Association. Available from: http://www.openmi.org/. [Accessed: 18/12/2017].

Pegram, GGS, Sinclair, S and Bárdossy, A. 2016. *New Methods of Infilling Southern African Raingauge Records Enhanced by Annual, Monthly and Daily Precipitation Estimates Tagged with Uncertainty*. WRC Report 2241/1/15. Water Research Commission (WRC), Pretoria, South Africa.

Pegram, GGS, Sinclair, S, Vischel, T and Nxumalo, N. 2010. *Soil Moisture from Satellites: Daily Maps Over RSA for Flash Flood Forecasting, Drought Monitoring, Catchment Management & Agriculture*. WRC Report 1683/1/10. Water Research Commission (WRC), Pretoria, South Africa.

Penman, HL. 1948. Natural evaporation from open water, bare soil and grass. *Proceedings of the Royal Society*, London, United Kingdom. A193, 120-146.

Pike, A, Schulze, RE, Hallowes, L, Thornton-Dibb, SLC, Clark, DJ, Horan, MJC, Taylor, V and Consultants, W. 2004. New developments in, and refinements to, supporting software, documentation, user support andpropontion of the *ACRU* agrohydrological modelling system. In: eds. Schulze, RE and Pike, A, *Development and evaluation of an installed hydrological modelling system*. WRC Report No. 1155/1/04, Chapter 4. Water Research Commission, Pretoria, South Africa.

Pollard, S and du Toit, D. 2008. Integrated water resource management in complex systems: How the catchment management strategies seek to achieve sustainability and equity in water resources in South Africa. *WaterSA,* 34 (6): 671-679.

Rahman, J, Perraud, J, Hotham, H, Murray, N, Leighton, B, Freebairn, A, Davis, G and Bridgart, R. 2005. Evolution of TIME. In: eds. Zerger, A and Argent, RM, *MODSIM 2005 International Congress on Modelling and Simulation*, Melbourne, Australia, 697-703. Modelling and Simulation Society of Australia and New Zealand.

Rahman, J, Seaton, S, Perraud, J, Hotham, H, Verrelli, D and Coleman, J. 2003. It's TIME for a new environmental modelling framework. In: ed. Post, DA, *MODSIM 2003 International Congress on Modelling and Simulation*, Townsville, Australia, 1727-1732. Modelling and Simulation Society of Australia and New Zealand.

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.

Reitsma, R and Carron, J. 1997. Object-oriented simulation and evaluation of river basin operations. *Journal of Geographic Information and Decision Analysis,* 1 (1): 10-24.

Ritchie, JT. 1972. A model for predicting evaporation from a row crop with incomplete cover. *Water Resources Research,* 8 (5): 1204-1213.

Rumbaugh, J, Blaha, M, Premerlani, W, Eddy, F and Lorensen, W. 1991. *Object-oriented modeling and design*. Prentice Hall, Englewood Cliffs, New Jersey, USA.

Schmidt, EJ and Schulze, RE. 1987. *Flood volume and peak discharge from small catchments in southern Africa, based on the SCS technique*. Technology Transfer Report TT/3/87. Water Research Commission, Pretoria, South Africa.

Schmidt, J, Kienzle, SW and Srinivasan, MS. 2009. Estimating increased evapotranspiration losses caused by irrigated agriculture as part of the water balance of the Orari catchment, Canterbury, New Zealand. *Journal of Hydrology (New Zealand),* 48 (2): 73-94.

Schulze, K, Hunger, M and Döll, P. 2005. Simulating river flow velocity on global scale. *Advances in Geosciences,* 5: 133–136.

Schulze, RE. 1975. Catchment evapotranspiration in the Natal Drakensburg. Unpublished PhD thesis. Department of Geography, University of Natal, Pietermaritzburg, South Africa.

Schulze, RE. 1983. *Agrohydrology and -Climatology of Natal*. Water Research Commission, Pretoria, South Africa.

Schulze, RE. 1984. *Hydrological Models for Application to Small Rural Catchments in Southern Africa : Refinements and Development.* WRC Report No. 63/2/84. Water Research Commission, Pretoria, South Africa.

Schulze, RE. 1989. *ACRU : Background, Concepts and Theory*. WRC Report No. 154/1/89. Water Research Commission, Pretoria, South Africa.

Schulze, RE (ed.). 1995a. *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. 1995b. Soil water budgeting and total evaporation. In: ed. Schulze, RE, *Hydrology and agrohydrology: A text to accompany the ACRU 3.00 agrohydrological modelling system*. WRC Report No. TT69/95. Chapter 2. Water Research Commission, Pretoria, South Africa.

Schulze, RE. 1995c. Streamflow. In: ed. Schulze, RE, *Hydrology and agrohydrology: A text to accompany the ACRU 3.00 agrohydrological modelling system*. WRC Report No. TT69/95. Chapter 10. Water Research Commission, Pretoria, South Africa.

Schulze, RE. 1995d. Verification Studies. In: ed. Schulze, RE, *Hydrology and agrohydrology: A text to accompany the ACRU 3.00 agrohydrological modelling system*. WRC Report No. TT69/95. Chapter 22. Water Research Commission, Pretoria, South Africa.

Schulze, RE. 2013. *Modelling Impacts Of Land Use On Hydrological Responses In South Africa With The ACRU Model By Sub-Delineation Of Quinary Catchments Into Land Use Dependent Hydrological Response Units*. Internal report. Centre for Water Resources Research, University of KwaZulu-Natal, Pietermaritzburg, South Africa.

Schulze, RE, Angus, GR, Lynch, SD and Smithers, JC. 1995. *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. Chapter 2. Water Research Commission, Pretoria, South Africa.

Schulze, RE, George, WJ, Angus, GR and Lynch, SD (eds.). 1990. *ACRU-2.0: User Manual*. WRC Report No. 154/2/89. Water Research Commission, Pretoria, South Africa.

Schulze, RE and Horan, MJC. 2008. Section 4.2: Soils Hydrological Attributes. In: ed. Schulze, RE, *South African Atlas of Climatology and Agrohydrology*. WRC Report 1489/1/06. Water Research Commission, Pretoria, South Africa.

Schulze, RE, Lorentz, SA, Kienzle, SW and Perks, LA. 2004. Case Study 3: Modelling the Impacts of Land-use and Climate Change on Hydrological Responses in the Mixed Underdeveloped/Developed Mgeni Catchment, South Africa. In: eds. Kabat, P, Claussen, M, Dirmeyer, PA, Gash, JHC, Bravo de Guenni, L, Meybeck, M, Pielke, RA, Vörösmarty, CI, Hutjes, RWA and Lütkemeier, S, *Vegetation, Water, Humans and the Climate: A New Perspective on an Interactive System*. Springer, Berlin, Heidelberg.

Schulze, RE and Maharaj, M. 2008a. Section 7.3: Daily Maximum Temperatures. In: ed. Schulze, RE, *South African Atlas of Climatology and Agrohydrology*. WRC Report 1489/1/06. Water Research Commission, Pretoria, South Africa.

Schulze, RE and Maharaj, M. 2008b. Section 7.5: Daily Minimum Temperatures. In: ed. Schulze, RE, *South African Atlas of Climatology and Agrohydrology*. WRC Report 1489/1/06. Water Research Commission, Pretoria, South Africa.

Schulze, RE and Schütte, S. 2015. Local and accumulated downstream impacts of dryland sugarcane on streamflows in three South African catchments: Can a case be made for sugarcane to become a ‘streamflow reduction activity’? *Proceedings of the South African Sugar Technologists' Association*.

Schulze, RE and Smithers, JC. 2004. The *ACRU* Modelling System as of 2002 : Background, Concepts, Structure, Output, Typical Applications and Operations. In: ed. Schulze, RE, *Modelling as a Tool in Integrated Water Resources Management: Conceptual Issues and Case Study Applications*. WRC Report 749/1/04. Chapter 2. Water Research Commission, Pretoria, South Africa.

Schütte, S. 2014. Linkages between selected hydrological ecosystem services and land use changes, as indicated by hydrological responses. A case study on the Mpushini/ Mkhondeni Catchments, South Africa. Unpublished Master of Science in Hydrology thesis, Centre for Water Resources Research, University of KwaZulu-Natal, Pietermaritzburg, South Africa.

Schütte, S and Schulze, RE. 2017. Projected impacts of urbanisation on hydrological resource flows: A case study within the uMngeni Catchment, South Africa. *Journal of Environmental Management,* 196: 527-543.

Shane, RM, Zagona, EA, McIntosh, D, Fulp, TJ and Goranflo, HM. 1996. Project object. *Civil Engineering,* 66 (1): 61-63.

Shaw, EM. 1994. *Hydrology in practice*. Third Edition. Chapman and Hall, London, UK.

Shuttleworth, WJ. 2010. Back to the basics of understanding ET. *Kovacs Colloquium. Hydrocomplexity: New Tools for Solving Wicked Water Problems*, Paris, France. IAHS.

Silvert, W. 1993. Object-oriented ecosystem modelling. *Ecological Modelling,* 68 (1-2): 91-118.

Silvert, W. 2001. Modelling as a discipline. *International Journal of General Systems,* 30 (3): 261-282.

Simonovic, SP, Fahmy, H and El-Shorbagy, A. 1997. Use of object-oriented modeling for water resources planning in Egypt. *Water Resources Management,* 11: 243-261.

Sinclair, S and Pegram, GGS. 2010. A comparison of ASCAT and modelled soil moisture over South Africa, using TOPKAPI in land surface mode. *Hydrol. Earth Syst. Sci.,* 14 (4): 613-626.

Sinclair, S and Pegram, GGS. 2013. *HYLARSMET: A Hydrologically Consistent Land Surface Model for Soil Moisture and Evapotranspiration Modelling over Southern Africa using Remote Sensing and Meteorological Data*. WRC Report 2024/1/12. Water Research Commission (WRC), Pretoria, South Africa.

SLIM. 2014. Quaternary Catchment Boundaries of South Africa. [Dataset]. Directorate: of Spatial & Land Information Management (SLIM), Department of Water and Sanitation, Pretoria, South Africa.

Smithers, JC. 2014. *Flood hazard determination - hydraulic modelling, KZN flood risk study: Phase 2: Deliverable 2.3*. Jeffares and Green, P.O. Box 794, Hilton, South Africa.

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*. WRC Report No. TT69/95. Chapter 13. Water Research Commission, Pretoria, South Africa.

Smithers, JC, Chetty, KT, Frezghi, MS, Knoesen, DM and Tewolde, MH. 2013. Development and assessment of a daily time-step continuous simulation modelling approach for design flood estimation at ungauged locations: *ACRU* model and Thukela Catchment case study. *Water SA,* 39 (4): 00-00.

Smithers, JC, Gray, RP, Johnson, S and Still, D. 2017. Modelling and water yield assessment of Lake Sibhayi. *Water SA,* 43: 480-491.

Smithers, JC and Schulze, RE (eds.). 1995. *ACRU Agrohydrological Modelling System : User Manual Version 3.00*. WRC Report No. TT70/95. Water Research Commission, Pretoria, South Africa.

Spanou, M and Chen, D. 2000. An object-oriented tool for the control of point-source pollution in river systems. *Environmental Modelling & Software,* 15: 35-54.

StatsSA. 2012. Census 2011. [Dataset]. Statistics South Africa, Pretoria, South Africa. www.statssa.org.za. [Accessed: 20/09/2017].

Stephens, T. 2010. *Manual on Small Earth Dams: A Guide to Siting, Design, and Construction*. Food and Agriculture Organization of the United Nations, Rome, Italy.

Sydelko, PJ, Dolph, J, Taxon, T and Majerus, K. 1999. A dynamic object-oriented architecture approach to ecosystem modeling and simulation. *Proceedings of the 1999 American Society for Photogrammetry and Remote Sensing (ASPRS) Annual Conference*, Portland, Oregon, USA, 410-420.

Sydelko, PJ, Hlohowskyj, I, Majerus, K, Christiansen, J and Dolph, J. 2001. An object-oriented framework for dynamic ecosystem modeling: application for integrated risk assessment. *Science of the Total Environment,* 274 (1-3): 271-281.

Tague, CL and Band, LE. 2004. RHESSys: Regional Hydro-Ecologic Simulation System-An object-oriented approach to spatially distributed modeling of carbon, water, and nutrient cycling. *Earth Interactions,* 8: 1-42.

Tarboton, KC and Schulze, RE. 1991. The *ACRU* modelling system for large catchment water resources management. In: eds. Van de Ven, FHM, Gutknecht, D, Loucks, DP and Salewicz, KA, *Hydrology for the Water Management of Large River Basins*, Vienna, Austria, 219-232. IAHS Publication No 201.

Tarboton, KC and Schulze, RE. 1992. *Distributed hydrological modelling system for the Mgeni catchment*. WRC Report 234/1/92. Water Research Commission (WRC), Pretoria, South Africa.

Teweldebrhan, AT. 2003. The Hydrosalinity Module of *ACRU* Agrohydrological Modelling System *(ACRU Salinity*): Module Development and Evaluation. Unpublished Master of Science in Hydrology thesis, School of Bioresources Engineering and Environmental Hydrology, University of KwaZulu-Natal, Pietermaritzburg, South Africa.

Thornthwaite, CW. 1948. An approach towards a rational classification of climate. *Geographical Review,* 38: 55-94.

Thornton-Dibb, SLC, Smithers, JC and Clark, DJ. 2013. Review and evaluation of river network models. In: eds. Clark, DJ and Smithers, JC, *Model integration for operational Water resources planning and management*. WRC Report No. 1951/1/12, Chapter 2. Water Research Commission, Pretoria, South Africa.

Umgeni Water. 2016. *Infrastructure Master Plan 2016: 2016/2017 - 2046/2047*. Planning Services, Engineering & Scientific Services Division, Umgeni Water, P.O. Box 9, Pietermaritzburg, 3200, South Africa.

USDA. 1985. *National Engineering Handbook: Section 4 - Hydrology*. United States Department of Agriculture (USDA) Soil Conservation Service, Washington DC, USA.

Valerio, A, Rajaram, H and Zagona, E. 2010. Incorporating groundwater-surface water interaction into river management models. *Ground Water,* 48 (5): 661-673.

Wang, J, Endreny, TA and Hassett, JM. 2005a. A flexible modeling package for topographically based watershed hydrology. *Journal of Hydrology,* 314 (1-4): 78-91.

Wang, J, Hassett, J, Endreny, T and McDonnell, J. 2000. *Criteria for selection of models for water quality management in urbanizing areas*. College of Environmental Science and Forestry, Syracuse, New York, USA.

Wang, J, Hassett, JM and Endreny, TA. 2005b. An object oriented approach to the description and simulation of watershed scale hydrologic processes. *Computers & Geosciences,* 31 (4): 425-435.

Warburton, ML. 2011. Challenges in modelling hydrological responses to impacts and interactions of land use and climate change. Unpublished Doctor of Philosophy in Hydrology thesis, School of Bioresources Engineering and Environmental Hydrology, University of KwaZulu-Natal, Pietermaritzburg.

Warburton, ML, Schulze, RE and Jewitt, GPW. 2010. Confirmation of *ACRU* model results for applications in land use and climate change studies. *Hydrology and Earth System Sciences,* 14 (12): 2399.

Weddepohl, JP. 1988. Design rainfall distributions for southern Africa. Unpublished MSc dissertation. Department of Agricultural Engineering, University of Natal, Pietermaritzburg, South Africa.

Weepener, HL, van den Berg, HM, Metz, M and Hamandawana, H. 2011a. *The development of a hydrologically improved Digital Elevation Model and derived products for South Africa based on the SRTM DEM*. WRC Report 1908/1/11. Water Research Commission, Pretoria, South Africa.

Weepener, HL, van den Berg, HM, Metz, M and Hamandawana, H. 2011b. Flow direction raster based on SRTM 90m DEM. SRTM90m_Flowdir_sfd.tif. [Dataset]. WRC Report 1908/1/11, WRC, Pretoria, South Africa.

Weepener, HL, van den Berg, HM, Metz, M and Hamandawana, H. 2011c. Flowpath improved DEM based on SRTM 90m DEM. SRTM90m_Flowpath_improved.tif. [Dataset]. WRC Report 1908/1/11, Water Research Commission, Pretoria, South Africa.

Weepener, HL, van den Berg, HM, Metz, M and Hamandawana, H. 2011d. Flowpaths based on SRTM 90m DEM. SRTM90m_flow_paths.shp. [Dataset]. WRC Report 1908/1/11, Water Research Commission, Pretoria, South Africa.

Weepener, HL, van den Berg, HM, Metz, M and Hamandawana, H. 2011e. Gap-filled DEM based on SRTM 90m DEM. SRTM90m_Gapfilled.tif. [Dataset]. WRC Report 1908/1/11, Water Research Commission, Pretoria, South Africa.

Wegner, P. 1990. Concepts and paradigms of object-oriented programming. *ACM SIGPLAN OOPS Messenger,* 1 (1): 7-87.

Werner, M, Schellekens, J, Gijsbers, P, van Dijk, M, van den Akker, O and Heynert, K. 2013. The Delft-FEWS flow forecasting system. *Environmental Modelling and Software,* 40: 65-77.

Wirth, N. 2006. Good ideas, through the looking glass. *IEEE Computer Society,* 39 (1): 28-39.

WWF-SA. 2017. *Scenarios for the Future of Water in South Africa*. World Wide Fund for Nature – South Africa (WWF-SA) and the Boston Consulting Group (BCG), Cape Town, South Africa.

Yang, A, Dutta, D, Vaze, J, Kim, S and Podger, G. 2017. An integrated modelling framework for building a daily river system model for the Murray–Darling Basin, Australia. *International Journal of River Basin Management,* 15 (3): 373-384.

Zagona, EA, Fuip, TJ, Shane, R, Magee, T and Goranflo, HM. 2001. RiverWare: A generalized tool for complex reservoir system modeling. *JAWRA Journal of the American Water Resources Association,* 37 (4): 913-929.

Zhang, M, Bu, X and Yue, P. 2017. GeoJModelBuilder: an open source geoprocessing workflow tool. *Open Geospatial Data, Software and Standards,* 2 (8): 1-8.

8 APPENDICES

8.1 Background to the Development of the *ACRU* Model

The *ACRU* model was originally developed as part of a distributed catchment evapotranspiration study (Schulze, 1975) conducted in the early 1970's in the Drakensberg region of KwaZulu-Natal, South Africa (Schulze and Smithers, 2004). The *ACRU* model was then further developed as an agrohydrological model as a result of research to compile an agrohydrological and agroclimatological atlas for KwaZulu-Natal (Schulze, 1983). Since its inception the *ACRU* model has been under continual development and refinement by staff and postgraduate students in what was the Agricultural Catchments Research Unit (*ACRU*) in the Department of Agricultural Engineering at the former University of Natal, then subsequently in the former 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 the current custodian of the *ACRU* model. This continued development and refinement of the model has been driven by requirements for better representation of hydrological systems and processes, new research and water management questions, availability of better datasets, and advances in computer systems and software development tools. Development and refinement of the model has taken place as part of numerous research projects, many of which were funded by the South African Water Research Commission (WRC). A timeline summarising development of the *ACRU* modelling system from its inception to present is shown in Table 8-1.

Table 8-1 Timeline summarising development of the *ACRU* modelling system (Schulze and Smithers, 2004)

***ACRU* 1**	
Early 1970s	Original development as a distributed catchment evapotranspiration model in the FORTRAN programming language (Schulze, 1975).
Early 1980s	Subsequent development as an agrohydrological model as a result of research to compile an agrohydrological and agroclimatological atlas for KwaZulu-Natal (Schulze, 1983).
1984	First user documentation for the *ACRU* 1 version published (Schulze, 1984).

***ACRU* 2**	
Late 1980s	Development of the *ACRU* 2 version of the model.
Late 1980s	Development of an *ACRU* configuration editor, known as the *Menubuilder*.
1989/1990	User documentation updated for the *ACRU* 2 version as a separate theory document (Schulze, 1989) and user manual (Schulze *et al.*, 1990).

Table 8-1 (cont.) Timeline summarising development of the *ACRU* modelling system (Schulze and Smithers, 2004)

***ACRU* 3**	
Early 1990s	Development of the *ACRU* 3 version of the model (Schulze, 1995a; Smithers and Schulze, 1995). Development of the *ACRU Menubuilder* to assist in preparing model input files (Smithers and Schulze, 1995).
1995	User documentation updated for the *ACRU* 3 version as a separate theory document (Schulze, 1995a) and user manual (Smithers and Schulze, 1995).
Early 2000s	Development of a new *ACRU* configuration editor, together with the *South African National* Quaternary Catchment Database (SANQCD), known as the *ACRU Agrohydrological Modelling System* (*AAHMS*) (Hallowes *et al.*, 2004; Pike *et al.*, 2004).
2001-2002	Development of the *ACRUView* time series analysis tool (Anonymous, 2004; Pike *et al.*, 2004).

***ACRU* 2000**	
1998-2002	Development of the object-oriented *ACRU* 2000 version of the model using the Java programming language (Clark *et al.*, 2001; Kiker, 2001; Kiker and Clark, 2001b; Kiker and Clark, 2001c; Kiker *et al.*, 2006).

***ACRU* 4**	
2008-2011	Development of the *ACRU* 4 version of the modelling system (Clark *et al.*, 2009; Clark *et al.*, 2012a; Clark *et al.*, 2012b), including the: • *ACRU* model structure, • *ModelData* XML model input file structure, • *ModelConfiguration* XML model configuration file structure, • *XmlModelFiles* library, • *ModelDataAccess* library for .Net, • *Configuration Editor* graphical user interface (GUI), • *TSAnalysis* time series analysis tools, • inclusion of *ACRU* in the *SPATSIM-HDSF* modelling framework, and • model input data file converter.

***ACRU* 5**	
2012	Development of OpenMI 1.4 compliant Java and .Net wrappers for the *ACRU* 5 version of the model (Clark and Lutchminarain, 2013).
2012-2014	Further development of *ACRU* 4 version of the modelling system (Clark, 2013), to create the *ACRU* 5 version, including refinements to the: • *ACRU* model structure (especially handling of time series data), • *ModelData* XML model input file structure, • *ModelConfiguration* XML model configuration file structure, and • *XmlModelFiles* library (model input data file updater). Development of a Java version of the *ModelDataAccess* library.
2015-2017	Development of OpenMI 2.0 compliant .Net wrapper for the *ACRU* 5 version of the model.
2017	Development of a data reader/writer for the Delft-FEWS PI XML file format in the Java version of the *ModelDataAccess* library.

Schulze and Smithers (2004) explain that following initial development in the early 1970s the *ACRU* model was then further developed as an agrohydrological model as a result of research to compile an agrohydrological and agroclimatological atlas for KwaZulu-Natal (Schulze, 1983). This first version of the *ACRU* model is referred to here as the *ACRU* 1 version and is described in Schulze (1984). In the late 1980s the *ACRU* model was further developed resulting in the *ACRU* 2 version and the Disk Operating System (DOS) based *Menubuilder* user interface utility was developed to assist model users in configuring the model (Schulze, 1989; Schulze *et al.*, 1990). In the early 1990s further model development took place resulting in the *ACRU* 3 version (Schulze, 1995a; Schulze *et al.*, 1995). In the early 2000s a database of *ACRU* model inputs such as information on soils, land cover/use and climate for each Quaternary Catchment in South Africa was developed based on a database developed by Meier (1997). This database known as the South African National Quaternary Catchment Database (SANQCD) was created in Microsoft Access and included a Windows based graphical user interface enabling users to create or edit *ACRU* configurations (Hallowes *et al.*, 2004; Pike *et al.*, 2004). The SANQCD together with the new user interface utility is known as the *ACRU Agrohydrological Modelling System* (AAHMS). Also in the early 2000s a standalone utility known as *ACRUView* was developed to graph and perform statistical analyses on times series of *ACRU* input and output data values (Anonymous, 2004; Pike *et al.*, 2004). Versions 1 to 3 of the *ACRU* model were developed in the FORTRAN 77 programming language. These versions used a flat text file structure for the main model input data file and two alternative formats of text file for daily time series data.

For several reasons, discussed in Section 2.2, work began in 1998 to restructure the *ACRU* model using the object-oriented software design approach in the Java programming language (Clark *et al.*, 2001; Kiker, 2001; Kiker and Clark, 2001b; Kiker and Clark, 2001c; Kiker *et al.*, 2006). Given the new model structure and the advent of the new millennium this new version of the model was named *ACRU* 2000. The *ACRU* 2000 version used the same algorithms as in the *ACRU* 3 version for representing hydrological processes, but provided a more flexible and extensible model structure to support future model development. After its completion in 2002 the *ACRU* 2000 version remained largely a research version for a few years. Several new modules were added to the *ACRU* 2000 version including: (i) dam and river operations (Butler, 2001), (ii) the *ACRU-Veld* module for modelling mixed vegetation land cover and utilisation by herbivores (Kiker and Clark, 2001a), (iii) the *ACRU-NP* module for Nitrogen and Phosphorus modelling (Campbell *et al.*, 2001), (iv) *ACRUSalinity* module for salinity modelling (Teweldebrhan, 2003), (v) *ACRUCane* for advanced sugarcane and irrigation modelling (Moult, 2005), and (vi) shallow water table modelling (Martinez *et al.*,

2008). Despite the new object-oriented structure of the *ACRU* 2000 version of the model, the model input files still used a flat text file structure, which did not enable the full potential of the new object-oriented model structure to be used.

In 2008 work began on developing new model configuration and model input data file designs using Extensible Markup Language (XML) to complement the object-oriented structure of the model (Clark *et al.*, 2009; Clark *et al.*, 2012a; Clark *et al.*, 2012b). Changes were made to the *ACRU* 2000 version of the model to implement the new model configuration and data file structure and this new version of the model became known as the *ACRU* 4 version (Clark *et al.*, 2009; Clark *et al.*, 2012a; Clark *et al.*, 2012b). A model input data file converter was also developed to convert *ACRU* 3 and *ACRU* 2000 model input data files to the new XML-based format. A new graphical user interface known as the *Configuration Editor* was developed based on the XML model configuration and data file structure (Clark *et al.*, 2009; Clark *et al.*, 2012a; Clark *et al.*, 2012b). Simultaneously the *TSAnalysis* time series visualisation and analysis tool was developed to replace the *ACRUView* and SPATSIM TSOFT time series analysis tools (Clark *et al.*, 2009; Clark *et al.*, 2012a). When the SPATSIM modelling framework was restructured, as part of WRC Project K5/1490, to create the hydrological modelling framework known as SPATSIM-HDSF, graphical and data interfaces were created to enable the *ACRU* 4 version to be run within SPATSIM-HDSF, reading from, and writing to, the SPATSIM-HDSF database structure (Clark *et al.*, 2009; Clark *et al.*, 2012a; Clark *et al.*, 2012b). Since 2010 the *ACRU* 4 version has replaced the *ACRU* 3 version, as the version of the model taught to undergraduate hydrology and agricultural engineering students at UKZN and in courses presented to hydrology practitioners. However, the *ACRU* 3 version is still used for some research applications within the CWRR for large national-scale simulations where the time taken to run simulations is a limitation of the *ACRU* 4 version.

From 2012 onwards further changes were made to the structure of the model and the model configuration and data files (Clark, 2013). Most of these changes were related to improving the handling of time series data, resulting in what will be referred to as the *ACRU* 5 version of the model for the purposes of this document. The concept of Resource classes to represent resources such as water, nutrients and sediment was an important addition to the *ACRU* 5 version. Initial development of a water accounting module (Clark, 2015a) for *ACRU* started in 2014 and is under continued development. A model input data file updater was developed to model input data files to be updated easily to keep up with changes to the model. A Java version of the *ModelDataAccess* library was developed to enable easier addition of data reader/writer classes for additional model data input formats. A data

reader/writer for the Delft-FEWS PI XML file format was included in the Java version of the *ModelDataAccess* library.

With increasing recognition of the need for integrated water resources management (IWRM), considering integrated water quantity, water quality, economic and social impacts, it was recognised that some means of linking *ACRU* to other domain specific models would be necessary. To achieve this, Open Modelling Interface (OpenMI) compliant Java and .Net wrappers (OpenMI version 1.4) were created for the *ACRU* 5 version of the model enabling *ACRU* to be linked to other OpenMI compliant models (Clark and Lutchminarain, 2013). OpenMI enables the models to be linked on a timestep-by-timestep basis, such that feedbacks between the models can be represented. In 2015 an OpenMI 2.0 compliant .Net wrapper was developed for the *ACRU* 5 version of the model.

8.2 Notation Used In UML Class Diagrams

The Unified Modelling Language (UML) is a notation developed to visually describe object-oriented models of real-world systems to facilitate communication and enable model designs to be documented in a manner that is independent of the computer programming language used to implement the model. A UML specification is maintained by the Object Management Group (OMG) in (OMG, 2017).

In this document, UML class diagrams have been used to describe the design of the new object-oriented structure of the *ACRU* model and the OpenMI wrapper classes for *ACRU* and eWater Source. The initial design of the restructured *ACRU* model using UML was created using the Rational Rose [https://www-03.ibm.com/software/products/en/enterprise] software. UML software such as Rational Rose enables generation of code from UML designs and reverse engineering of code to back to UML. The UML class diagrams in Chapter 3, describing the *ACRU* model, were created using the ObjectAid software [http://www.objectaid.com/] which automatically reverse engineered the Java code to UML enabling UML diagrams to be quickly generated based on actual model code. The UML class diagrams in Chapter 4, describing the OpenMI wrappers, were created using the Microsoft Visual Studio UML tools [https://www.visualstudio.com/] which automatically reverse engineered the .Net C# code to UML enabling UML diagrams to be quickly generated based on actual code.

The notation used to describe classes and interfaces in the UML class diagrams created using the ObjectAid software, is shown in Figure 8-1. Each class or interface is shown as

block with a yellow background, with name of the class at the top followed by the name of the package to which it belongs. The class attributes, constructors and methods are listed, with an indication of their visibility (public [+], protected [#], private [-]) and whether they are static^{S} or final^{F}. In the class diagrams in Chapter 3, some or all of the attributes and methods may have been hidden to reduce the size of the diagrams.

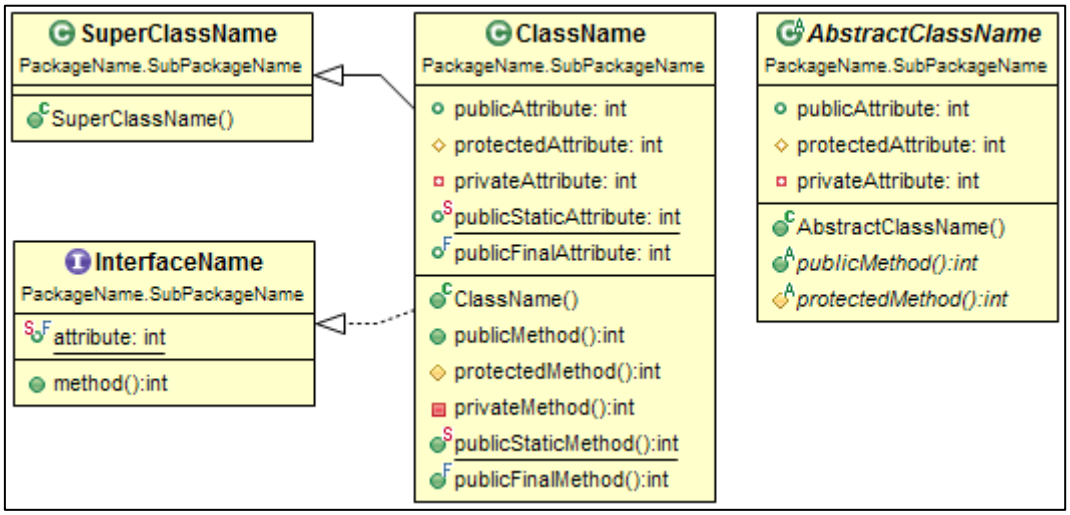

Figure 8-1 Notation used in UML class diagrams created using ObjectAid software

The notation used to describe classes and interfaces in the UML class diagrams created using the Microsoft Visual Studio software, is shown in Figure 8-2. Classes are shown as blocks with a blue background and interfaces as blocks with a green background. The name of a class is shown at the top followed by the class stereotype and the name of the superclass. The class fields, properties, constructors and methods are listed, with an indication of their visibility (public, protected, private). In the class diagrams in Chapter 4, some or all of the fields, properties and methods may have been hidden to reduce the size of the diagrams.

The notation used to describe the relationships between classes and interfaces in the UML class diagrams is shown in Figure 8-3. The inheritance relationships between classes are a core part of object-oriented design enabling a hierarchy of more general to more specialised classes to be created, where each class has a “type of” relationship with its superclass. Both the Java and C# programming languages permit only single inheritance. A realisation relationship is used when a class implements an interface. The class is obligated to implement the behaviour specified in each interface that it implements. Aggregation relationships are used to indicate situations where one class forms part of another class, such as an engine being part of a car. Association relationships are used to indicate any

other type of relationship between classes. In computer code the composition, aggregation and association relationships are implemented in the same way. The ObjectAid and Visual Studio UML tools, which create UML diagrams by reverse engineering the code, are thus not able to distinguish between these relationship types and so association relationships are used in all such cases.

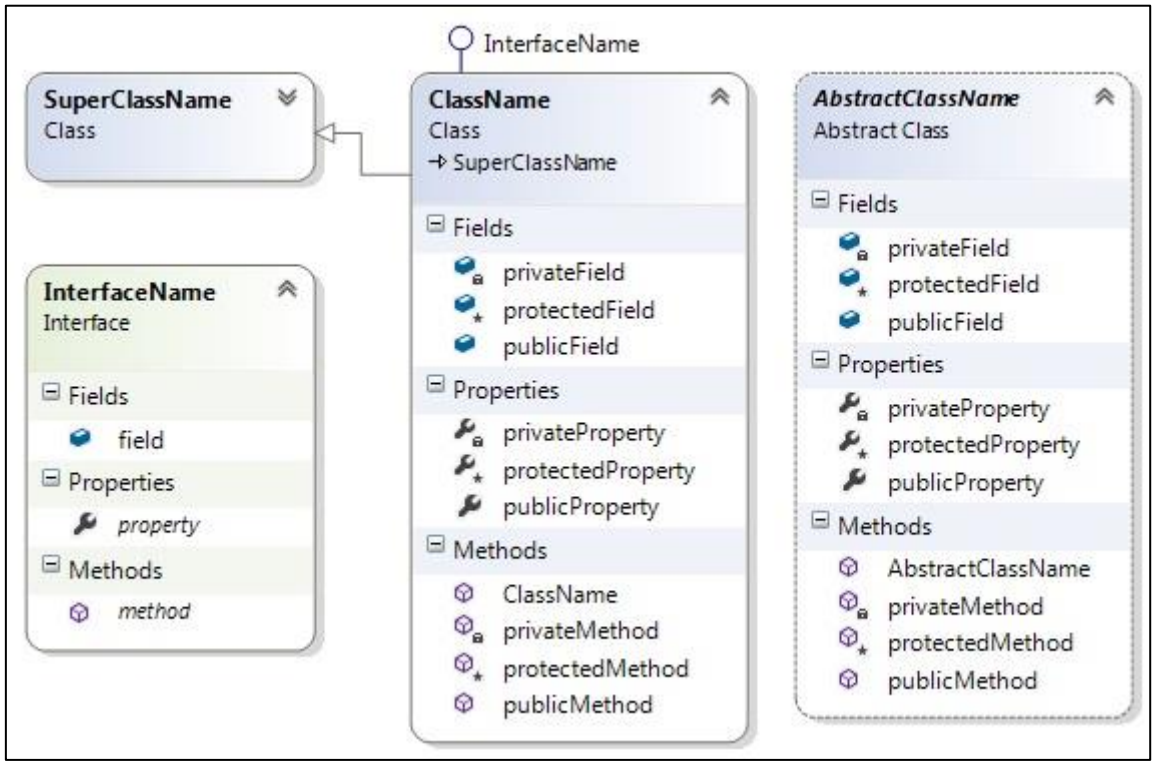

Figure 8-2 Notation used in UML class diagrams created using Visual Studio software

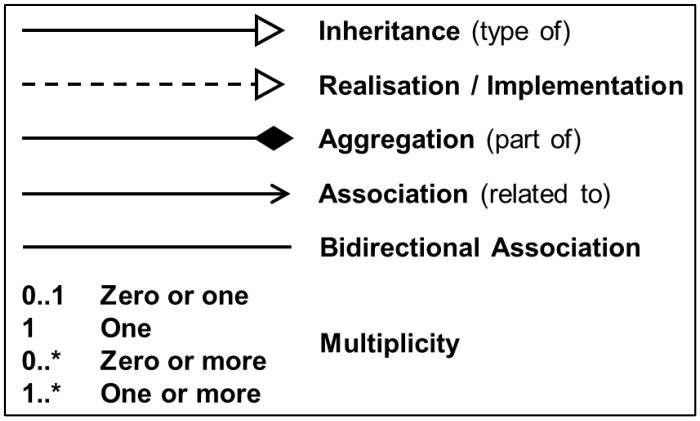

Figure 8-3 UML notation describing relationships between classes and interfaces

8.3 Initial Object-Oriented Design of the ACRU Model Structure

The initial conceptual object-oriented design of the *ACRU* model structure by Kiker and David (1998) and Kiker and Clark (1999) was briefly described in Section 3.2.1. The main classes and interfaces that form the foundation of the *ACRU* model are shown in more detail in Figure 8-4. The abstract *MModel* class is extended by the *MAcru2k* class which represents the *ACRU* model engine. The *MAcru2k* class contains two instance variables, where *spatialEntityPool* contains a list of instances of *CComponent* representing spatial entities in a hydrological system, and *computationOrder* contains a list of pointers to the spatial entities in the correct order for computing water flows within the system. The *WaterFlow* interface has one method, *flowWater* and is implemented by *MAcru2k*, *CSpatialUnit* and all subclasses of *PProcess* that represent water flows. In *MAcru2k* the *flowWater* method contains the algorithm that controls the ordered computation of Processes in the ordered list of spatial Components specified in *spatialEntityPool*. The three abstract high-level *CComponent*, *DData* and *PProcess* foundation classes are also shown, with *CComponent* and *PProcess* being subclasses of *CNode*. Instances of the *CNode* class may be related in the sense of being "next to" each other, for example, Components in the spatial sense and Processes in the sequential sense. Each instance of *CNode* may be "next to" zero or more other instances of *CNode*. The *CNode* contains several methods that enable these "next to" relationships to be set and queried. The *DData* class has methods to set, retrieve and increment a double precision floating point value stored within the class. The *CComponent* class has a Data class named *DStorage* that could be used to record the water stored within an instance of *CComponent*.

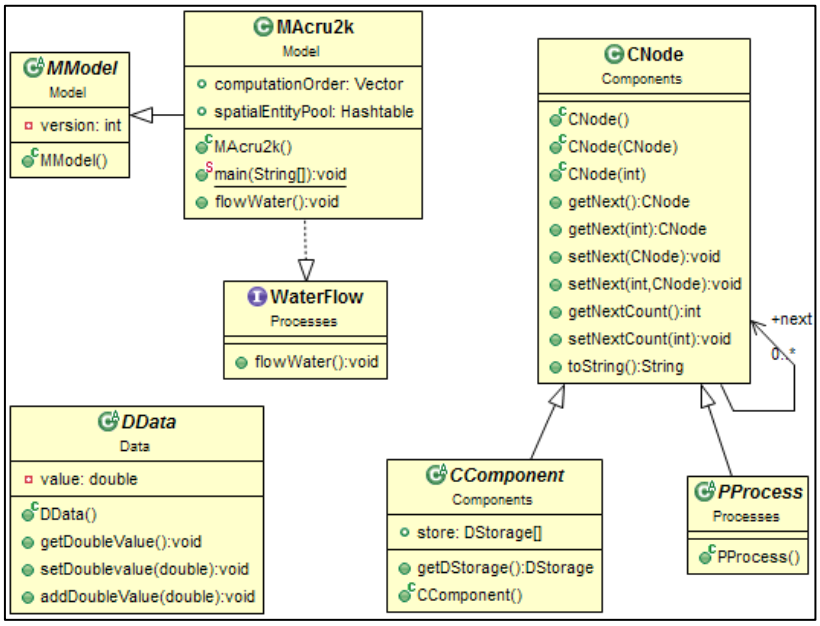

Figure 8-4 Initial design of the *ACRU* model: main classes and interfaces

8.3.1.1 Component classes

In the initial conceptual design hydrological systems are conceived as being composed of: (i) spatial entities, which could represent a region, line or point on a map, and (ii) entities representing the conceptual vertical layers that are subcomponents of some types of spatial entity. The classes representing spatial entities are shown in Figure 8-5. The spatial entities are represented by the *CSpatialUnit* class which is a subclass of *CComponent*. Instances of *CSpatialUnit* each have an identity number and an area. *CSpatialUnit* implements the *WaterFlow* interface and thus its *flowWater* method, which is called by the *flowWater* method in *MAcru2k*. The *CSpatialUnit flowWater* method contains the algorithm that controls the ordered computation of the list of *Processes* associated with an instance of *CSpatialUnit*. There are two subclasses of *CSpatialUnit*: (i) the *CLandSegment* class representing units of land, and (ii) the *CReach* class representing flow network reaches through *CRiver*, *CStream*, *CGully* and *CDam* subclasses. Each instance of *CSpatialUnit* is associated with an instance of *CClimate* representing the local climatic conditions for each spatial entity.

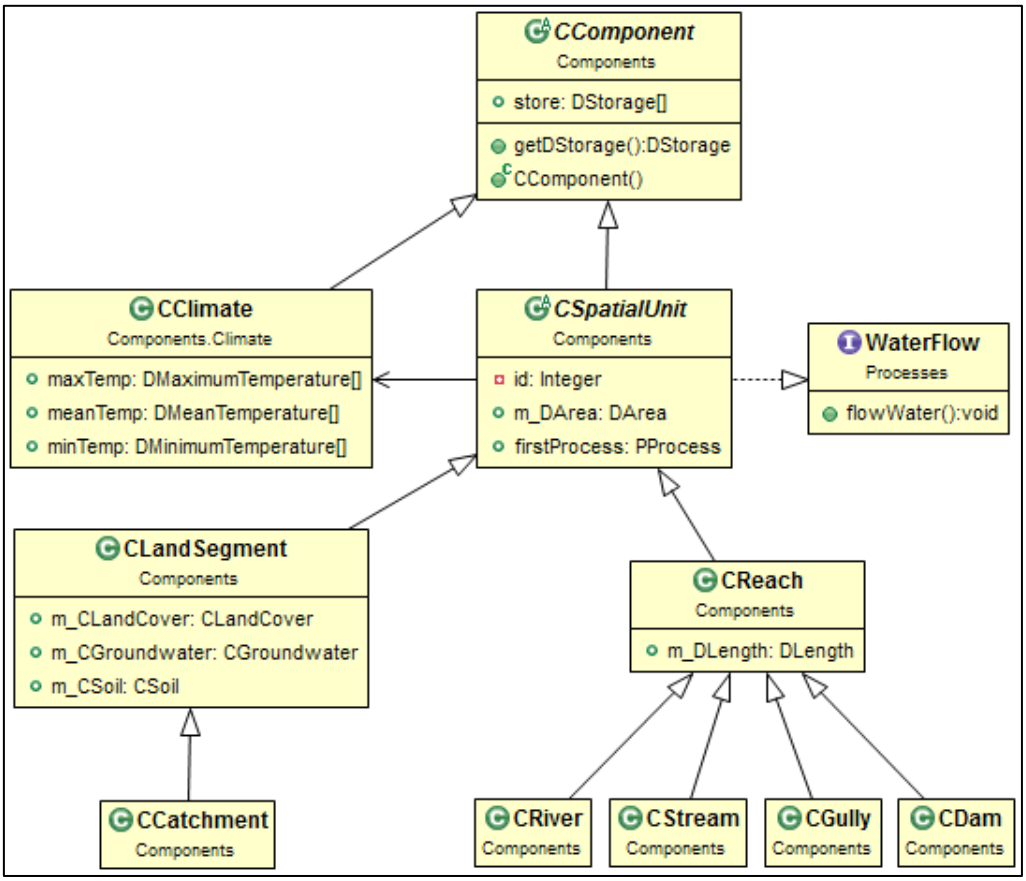

Figure 8-5 Initial design of the *ACRU* model: spatial Component classes

As shown in Figure 8-6 the *CLandSegment* class is composed of several classes representing vertical layers, starting at the top with *CClimate* inherited from *CSpatialUnit*. The land cover layer is represented by the *CLandCover* class and its subclasses. *CLandCover* has a *CImpervious* subclass representing impervious land cover, and a *CPervious* subclass representing pervious land cover such as vegetation. The *CVegetation* class is in turn composed of component classes *CLeafCanopy*, *CSeeds*, *CStems* and *CRoots*. The soil layer is represented by the *CSoil* class which in turn can be composed of one or more soil sublayers represented by the *CHorizon* class. The groundwater layer is represented by the *CGroundwater* class. The *CCatchment* class is a subclass of *CLandSegment*, but it is not clear in the initial design how this class would be used.

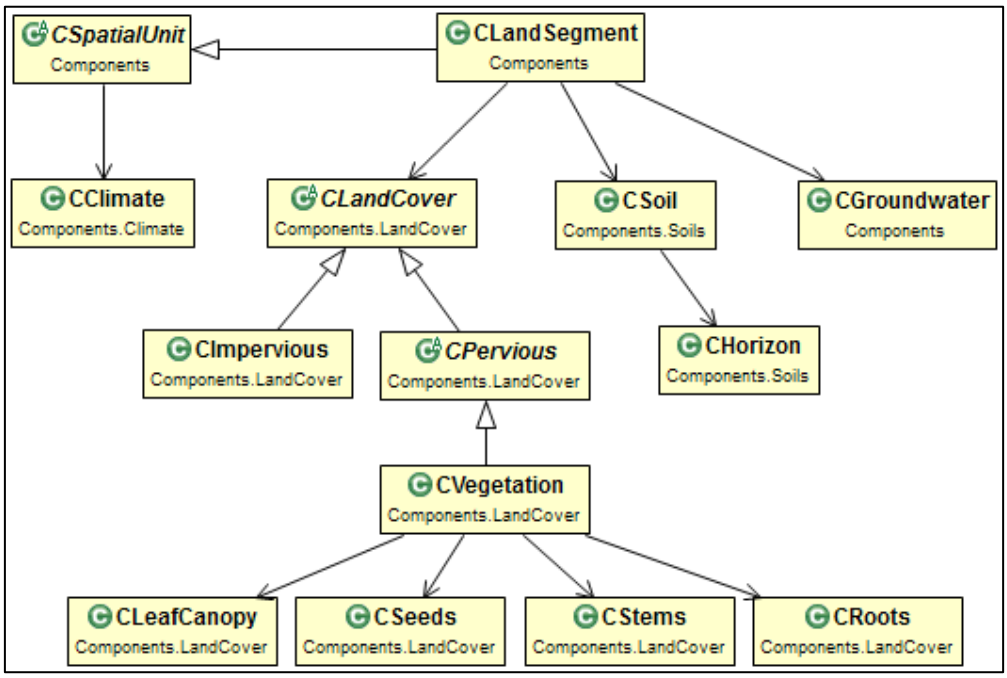

Figure 8-6 Initial design of the *ACRU* model: CLandSegment subcomponent classes

8.3.1.2 Data classes

The *DData* class and some examples of its subclasses are shown in Figure 8-7. Some of these subclasses are simple, such as *DArea* representing a single constant value, or *DMaximumTemperature* which could represent a time series of values. The *DStorage* class is more complex and is itself composed of other data classes and contains specialised methods to access these. All the subclasses inherit the methods of *DData* but may override them to provide subclass specific behaviour such as range checking.

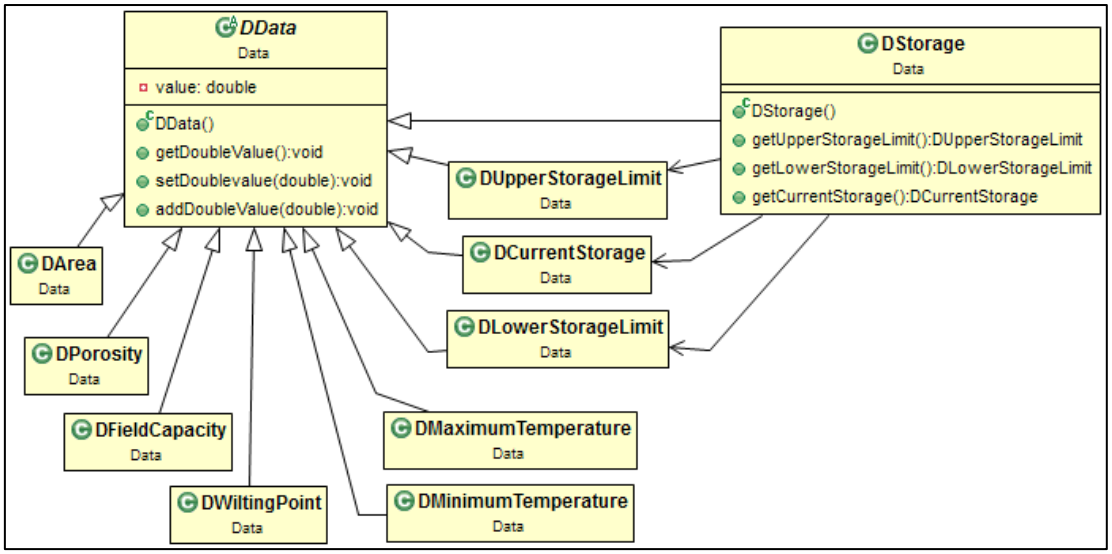

Figure 8-7 Initial design of the *ACRU* model: example Data classes

8.3.1.3 Process classes

The *PProcess* class and some examples of its subclasses are shown in Figure 8-8. The abstract PProcess class has several abstract subclasses, each representing a generalised category of hydrological process, such as interception, evaporation, transpiration, surface flow and subsurface flow. Each of these generalised subclasses then has one or more subclasses representing more specific processes or specific methods of modelling the process, for example the *PSCSRunoff* class uses the SCS method (USDA, 1985; Schmidt and Schulze, 1987) to estimate runoff.

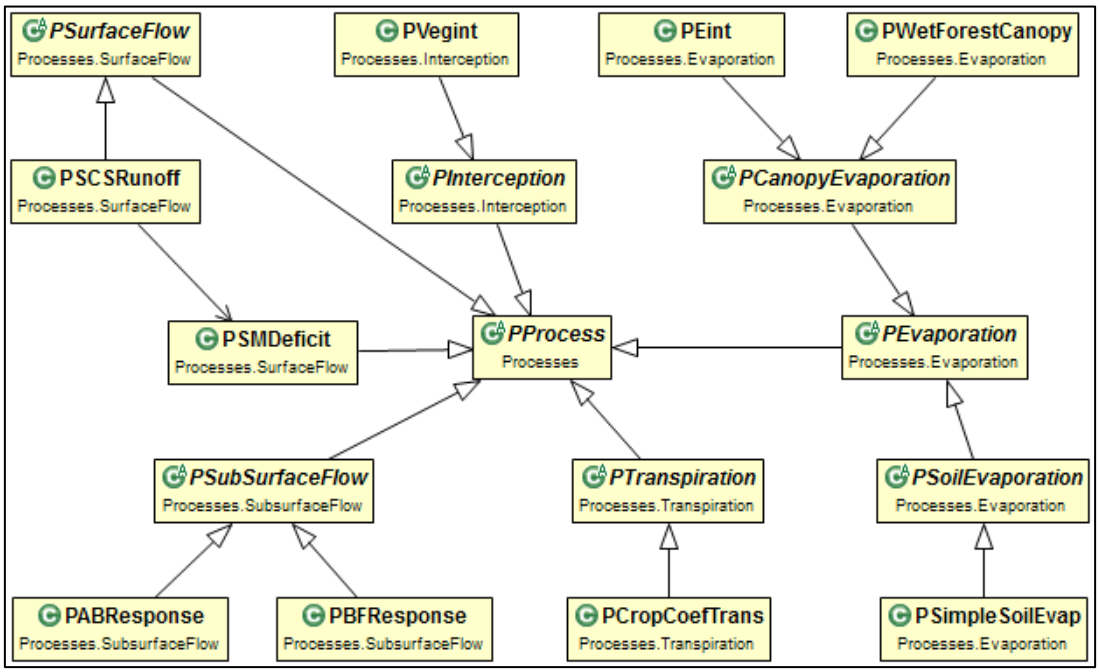

Figure 8-8 Initial design of the *ACRU* model: Process classes

An example of how Process, Component and Data classes are related is shown in Figure 8-9. The *PABResponse* class models the saturated downward movement of water from the soil's A-horizon to B-horizon when the soil moisture in the A-horizon is above drained upper limit. The *PABResponse* class implements the *WaterFlow* interface and the *flowWater* method, which is called by the *flowWater* method in the instance of *CSpatialUnit* on which the Process acts. The *PABResponse* class specifically acts on the two *CHorizon* Component objects representing the A-horizon and B-horizon belonging to a *CLandSegment* Component object. The *flowWater* method retrieves values for Component attributes (represented by subclasses of *DData*) such as drained upper limit and current soil moisture storage from the relevant *CHorizon* Component objects, then calculates whether there is any movement of water, and if so, adjusts the soil moisture storage in both *CHorizon* Component objects.

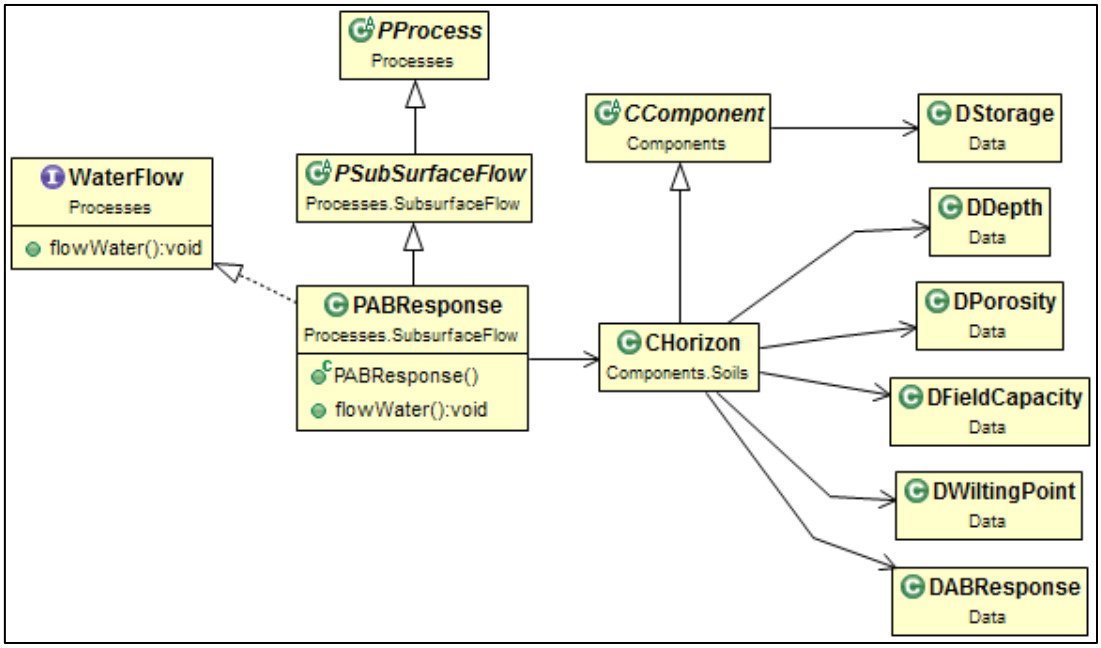

Figure 8-9 Initial design of the *ACRU* model: example of Component, Data, Process class relationships

8.4 Refined Object-Oriented Design of the *ACRU* Model Structure

This section contains some additional details of the refined object-oriented design of the *ACRU* model structure described in Section 3.2.2.

8.4.1 Java Generics Used in the *DData*, *DData_State* and *RResource* classes

In the *ACRU* 5 version extensive use was made of Java generic typing in the *DData*, *DData_State* and *RResource* classes, as described in Table 8-2 and Table 8-3, to simplify the use of different data value types in subclasses and ensure better type safety.

Table 8-2 Description of generic types used in the *DData* and *DData_State* classes

Type	Description
B	The base type of data structure used by an instance of *DData*
V	The value type of individual values in the data structure used by an instance of *DData*
DP	The time series data point type (sub-class of *DTSDataPoint*) used by an instance of *DData* to store a time-stamped value as part of a time series of data points
TS	The time series type (sub-class of *DTimeSeries*) used by an instance of *DData* to store a time series of data points

Table 8-3 Description of generic types used in the *RResource* class

Type	Description
V	The value type of individual quantity values used by an instance of *RResource*
IV	The type of data structure used to store a referenced list of source, destination, owner or location quantity values in *RResource*
DQ	The type (sub-class of *DData_State* with value type V) which records the quantity of resource currently stored within the container *CComponent* of an instance of *RResource*.
DPQ	The time series data point type (sub-class of *DTSDataPoint* with value type V) used by an instance of *RResource* to store a time-stamped storage quantity value as part of a time series of data points
TSQ	The time series type (sub-class of *DTimeSeries* with value type V) used by an instance of RResource to store a time series of storage quantity data points
DSI	The type (sub-class of *DData_State* and using the IV data structure) which records either: (i) the ownership quantities of the resource currently stored within the container *CComponent* of an instance of *RResource*, or (ii) the quantity the resource owned by the container *CComponent* of an instance of *RResource*, but stored in another instance of *CComponent*.
DI	The type (sub-class of *DData* and using the IV data structure) which records for the container *CComponent* of an instance of *RResource* either: (i) the source instances of CComponent from which quantities of resource were received, or (ii) the destination instances of *CComponent* to which quantities of resource were sent.
DPI	The time series data point type (sub-class of *DTSDataPoint* and using the IV data structure) used by an instance of *RResource* to store a time-stamped referenced list of source, destination, owner or location quantity values in *RResource* as part of a time series of data points
TSI	The time series type (sub-class of *DTimeSeries* and using the IV data structure) used by an instance of *RResource* to store a time series of referenced lists of source, destination, owner or location quantity values in *RResource*

8.4.2 Component Classes

An example is shown in Figure 8-10 of how impervious areas are represented conceptually using Component classes. The *ACRU* model distinguishes between impervious areas that are adjunct to the river flow network (*CAdjunctImperviousArea*), thus contributing runoff directly to the network, and those that are disjunct (*CDisjunctImperviousArea*), such that runoff contributes to the surface water on an adjacent pervious spatial unit of land. Spatially an instance of the *CImperviousArea* class would be part of an instance of the *CSubCatchment* class which in turn would be part of an instance of the *CCatchment* class. An instance of *CImperviousArea* could typically be composed of just two vertical subcomponent layers *CClimate* and *CLandCover*. The *CImperviousLandCover* class is used to represent an impervious land cover compared to the *CVegetation* class which represents a pervious vegetation land cover.

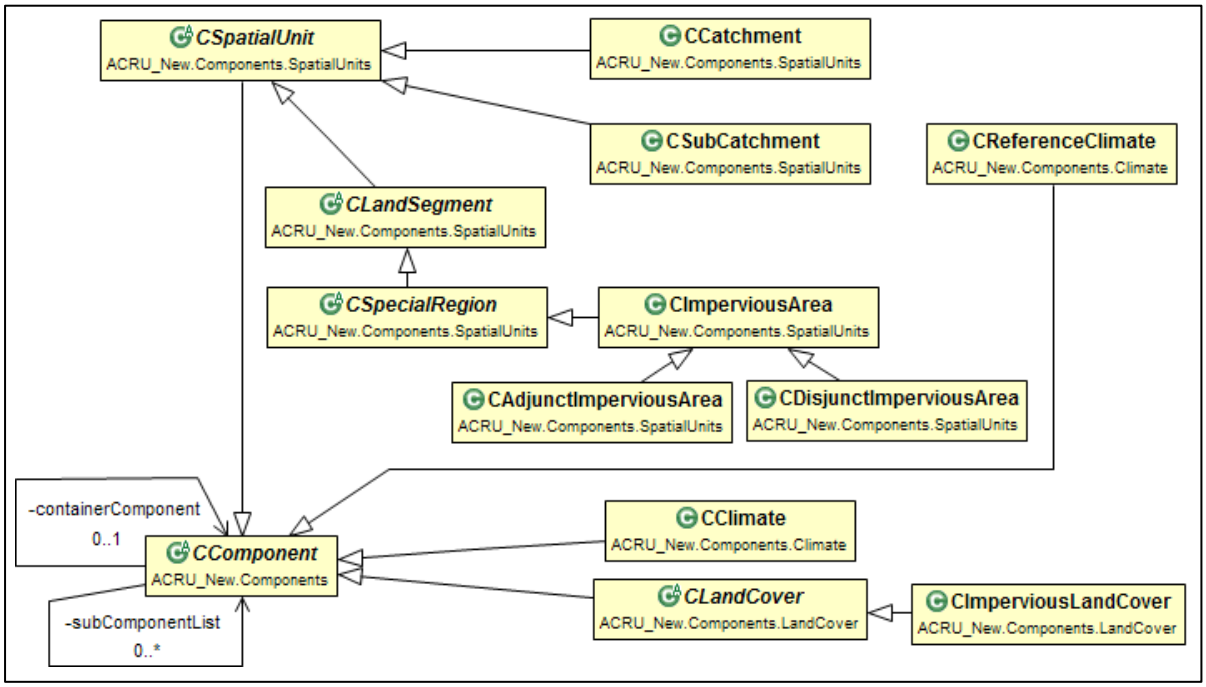

Figure 8-10 *ACRU* 5 design: *CImperviousArea* subcomponent Component classes

8.4.3 Data Classes

More detailed descriptions are provided in this section of the Data class structure and classes for storing and handling time series data.

8.4.3.1 Main Data classes

The details of the main classes of the new Data class structure are shown in Figure 8-11. This structure was designed to provide a hierarchy of powerful and flexible non-abstract Data classes to which parameter or variable identities, data ranges and other attributes can be assigned when they are instantiated as objects during model setup.

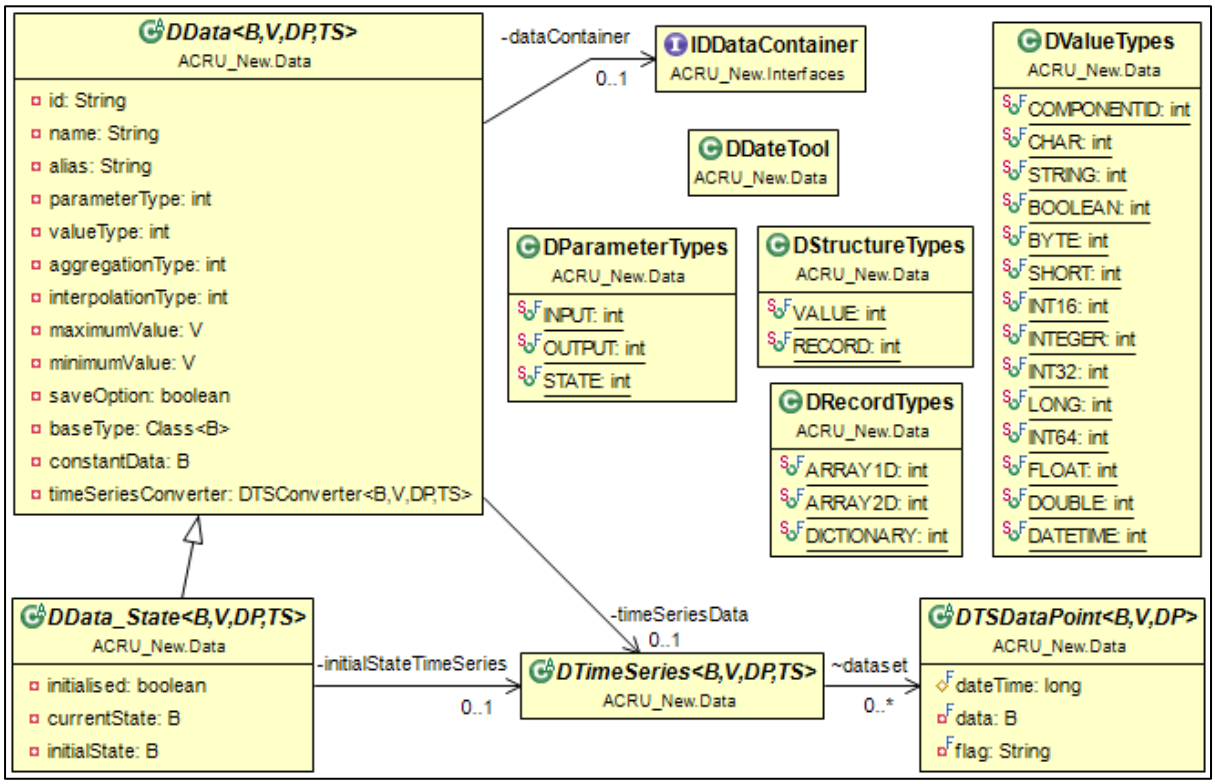

Figure 8-11 *ACRU* 5 design: main Data classes and associated data type description classes

Better provision has been made for different data structures and data value types in the *ACRU* 5 version of the Data classes. In the *ACRU* 5 version the term “base type” refers to different data structures, for example a single data value, an array of data values or a lookup table containing pairs keys and data values. The term “value type” refers to different data value types, for example string, integer or double precision floating point value types. Different data structures and data value types are implemented as subclasses of the *DData* and *DData_State* classes, as shown in Figure 8-12. For the simpler *DData_String*, *DData_Integer* and *DData_Double* classes the base type and the value type are the same, for example for *DData_Double* the base type and the value type are both *java.lang.Double*. For the *DData_CompID* class which represents the ID of a *CComponent* object, the base type and the value type are both *java.lang.String*. For the more complicated

DData_ListCompID class which represents a list of a *CComponent* object IDs, the base type is *ListCompID* and the value type is *java.lang.String*. The *ListCompID* base type is simply a wrapper class for a *java.util.Vector* list of *java.lang.String* objects. For the *DData_RefCompIDInteger* class the value type is *java.lang.Integer* and the *RefCompIDInteger* is a wrapper class for a *java.util.Hashtable* list with *CComponent* object IDs as keys and instances of *java.lang.Integer* as the referenced values. The *DStructureTypes*, *DRecordTypes* and *DValueTypes* classes, shown in Figure 3-8, each contain static variables that describe different data structures and data value types in a format that corresponds to the way in which this information is stored in the XML-based model configuration file described in Section 3.3.2. The value types in the *DValueTypes* class specify the data value type of a model parameter or variable, for example *DValueTypes.DOUBLE* for a Data object containing one constant double precision floating point value or a time series of double precision floating point values. The structure types in the *DStructureTypes* class specify whether, for example, a variable consists of: (i) one constant value, (ii) a set of constant values, termed a "record" here, (iii) a time series containing one value for each timestep, or (iv) a time series containing a set of values (a record) for each timestep. The *DRecordTypes* class contains the different record type options which are: (i) a one dimensional array, (ii) a two dimensional array, and (iii) a lookup table of key-value pairs, referred to as a "dictionary" in some programming languages. In addition, the *DParameterTypes* class contains static variables that specify whether a model parameter or variable is: (i) a model input, (ii) a model output, or (iii) a state variable which is inherently both a model input and a model output variable.

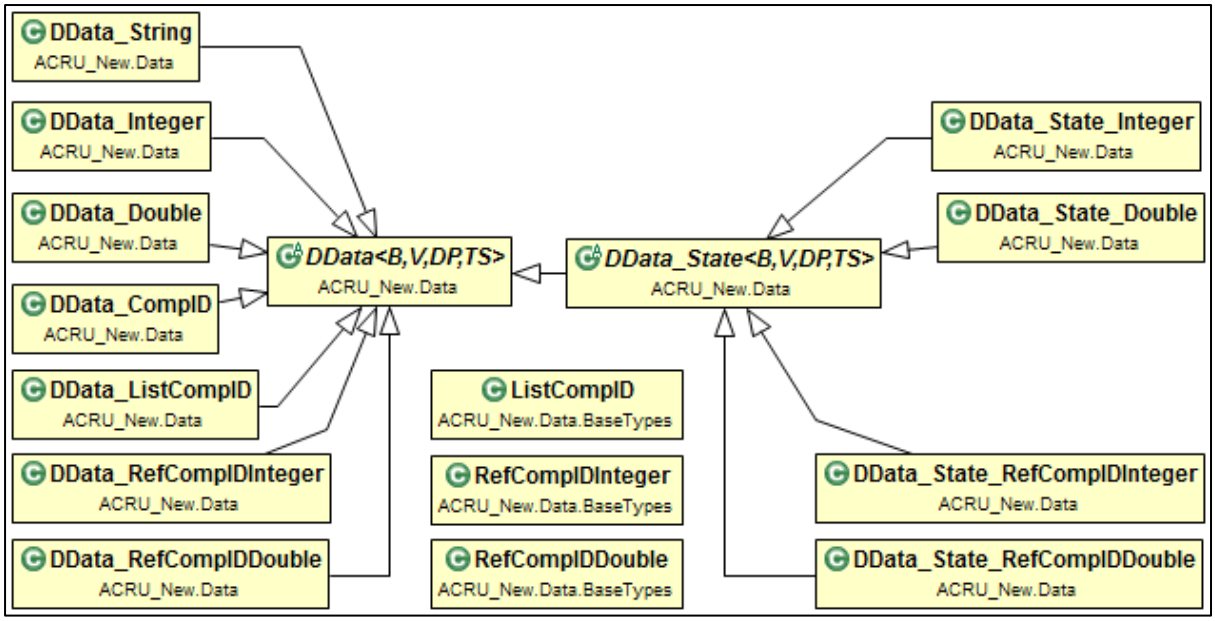

Figure 8-12 *ACRU* 5 design: subclasses of Data classes

8.4.3.2 Time series data classes

The new Data structure of the *ACRU* 5 version also provides better handling of time series data and more flexibility with regard to the types of time series that can be represented, as shown in Figure 8-13. The types of time series that can be defined are listed in the *DTSTypes* class and the permissible time series timesteps for non-breakpoint time series are listed in the *DTimeSteps* class. Different time series variables are aggregated up for coarser timesteps in different ways, for example, daily rainfall values would typically be summed to provide an annual value, but daily temperature values would be averaged. The *DTSAggregationTypes* class contains a list of the aggregation types that may be assigned to a Data object. The *DTSInterpolationTypes* class similarly contains a list of interpolation types, indicating how a time series variable should be interpolated to a finer timestep, but this functionality requires further development. Each instance of *DData* is also associated with an instance of the *DTSConverter* class, which enables conversion between different time series types, where appropriate, through aggregation or interpolation. The *DDateTool* class was created as a utility class to simplify use of the *java.util.Date* class by providing methods to assist with formatting dates and times as strings and returning the next or previous date or time relative to a specified data or time for a specified timestep.

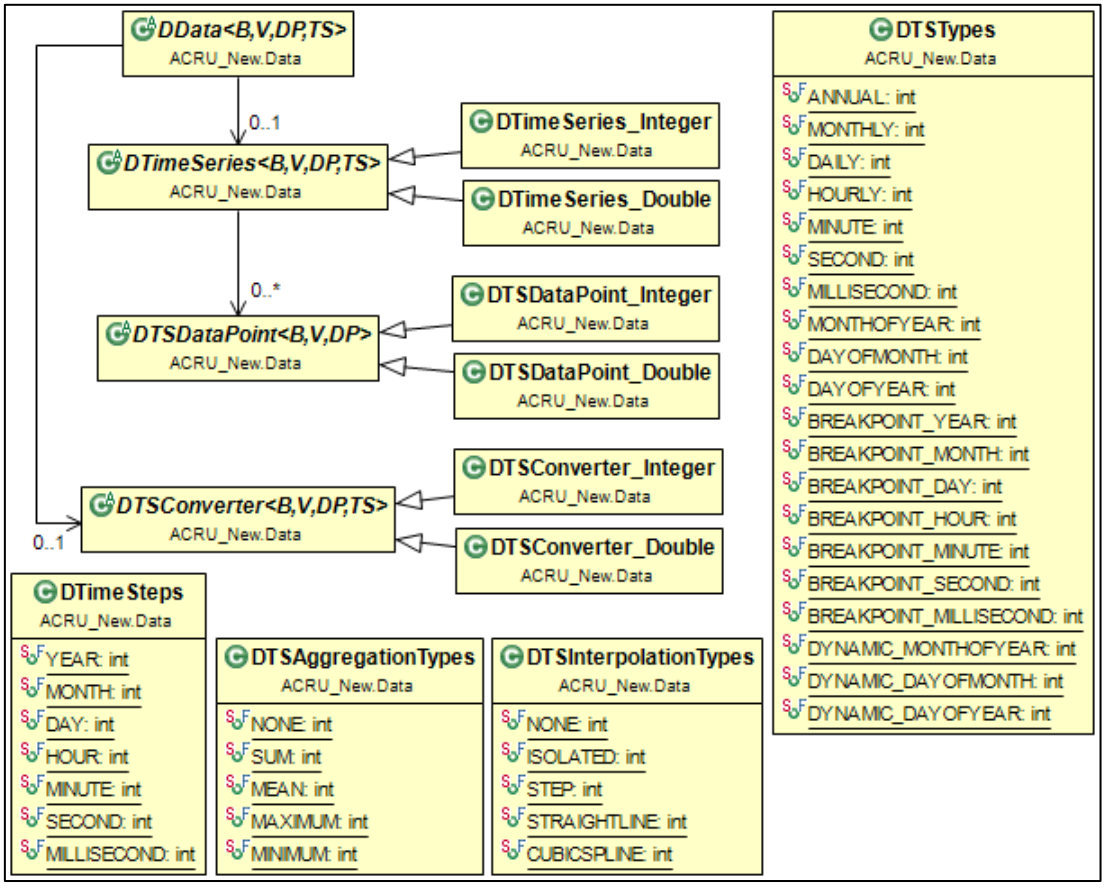

Figure 8-13 *ACRU* 5 design: time series related Data classes

8.5 Design and Development of an XML *ACRU* Model Input File Structure and Related Software Tools

The various file types and software tools related to the *ACRU* 5 version of the model are shown in Figure 8-14. The design and development of an XML *ACRU* model input file structure and related software tools are described in this section.

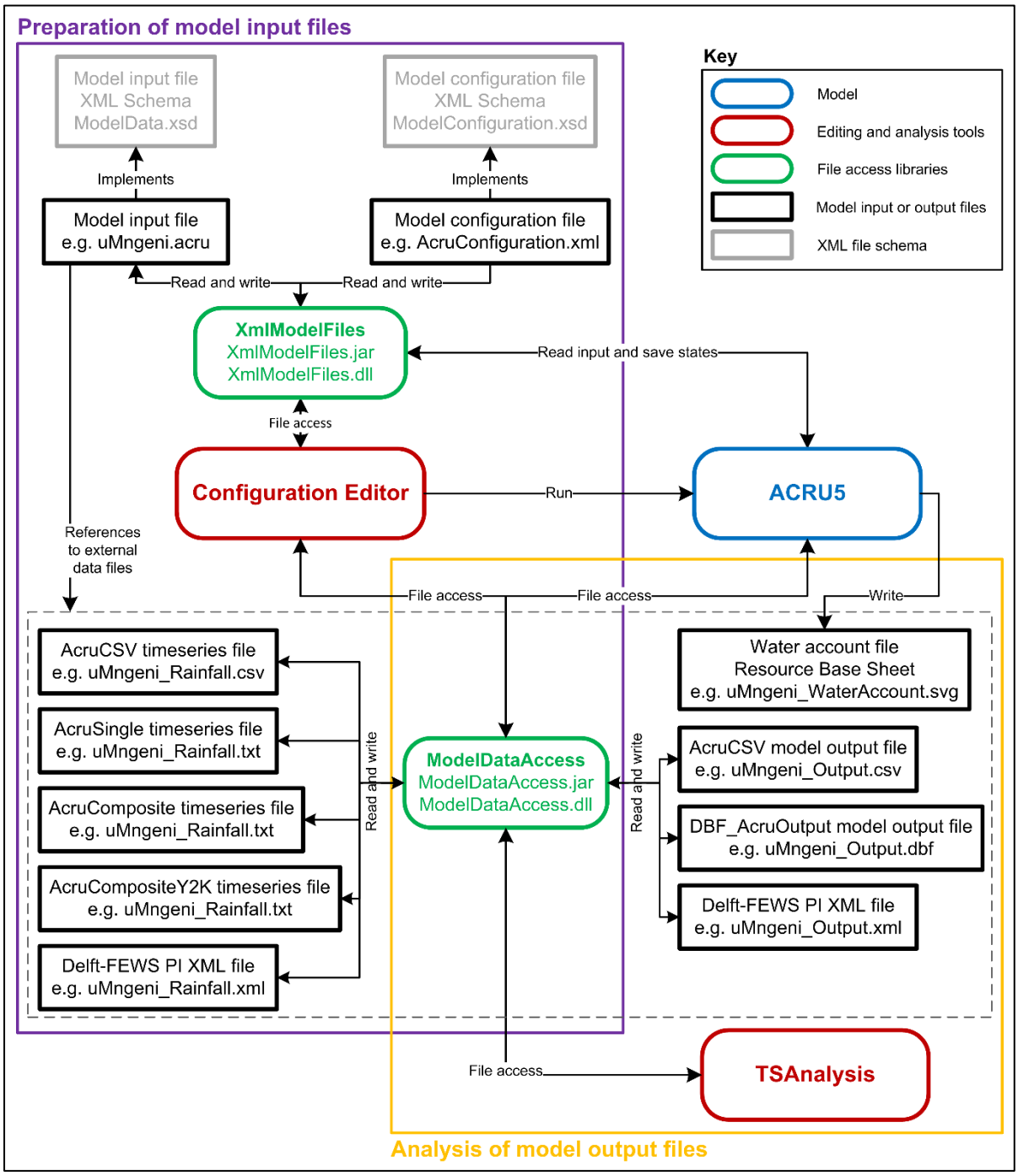

Figure 8-14 Software and files related to the *ACRU* 5 version of the model

8.5.1 *ModelData* Schema

The *ModelData* schema provides a data model describing the structure of an XML-based model input file that complements the object-oriented design of the *ACRU* model. 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. A different implementation of the *ModelData* schema would be used for each configuration of the *ACRU* model, that is, one XML model data file for each study catchment. The *ModelData* schema must be used in conjunction with the *ModelConfiguration* schema (Appendix 8.5.2). The root element of the *ModelData* schema, *Model*, and its sub-elements are shown in Figure 8-15.

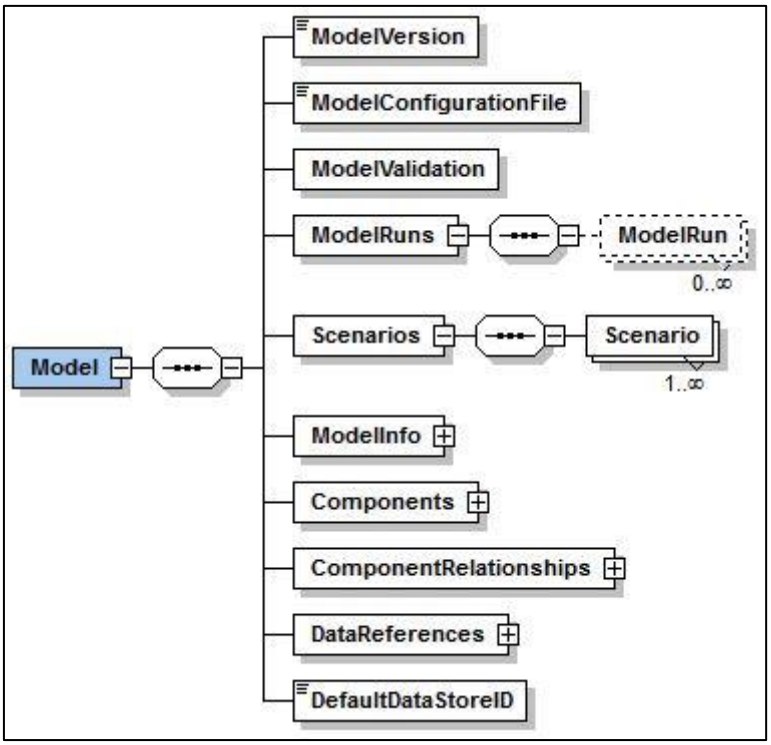

Figure 8-15 The *Model* element and main sub-elements of the *ModelData* schema

The *ModelVersion* element is used to store the version number of the *ACRU* model for which a *ModelData* file was created as a means of version control. The *ModelConfigurationFile* element is used to record the *ModelConfiguration* file associated with a *ModelData* file; this is another aspect of version control to ensure that the model version, *ModelData* file and *ModelConfiguration* file are all compatible. The *ModelValidation* element is used to hold information about whether the data in the *ModelData* file has been checked to be valid according to the associated *ModelConfiguration* file and when this was last checked. The purpose of the *DefaultDataStore* element is to enable a default data store to be specified, for example the name of a database or the default '*local'* for the *ModelData* file.

8.5.1.1 *ModelRuns* element

The *ModelRuns* element contains a set 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 optionally the 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.

8.5.1.2 *Scenarios* element

In water resource planning it is often useful to be able to model two or more different scenarios. A mechanism for setting up scenarios has been included in the data model, though this made the data model mode complicated. Each *Model* element contains 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, including: an ID, a description and an optional base scenario ID identifying an associated base scenario together with which this scenario should be applied. 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 values 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 would typically not have overlapping parameter or variable values.

8.5.1.3 *ModelInfo* element

A model configuration may include several global parameters and option variables. These are specified in the *ModelInfo* element, configured as a list of *Data* elements within a *DataList* element, as shown in Figure 8-16, to be consistent with the *Component* element. The *Data* element is described in Appendix 8.5.1.7.

Figure 8-16 The *ModelInfo* element of the *ModelData* schema

8.5.1.4 *Components* element

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, HRUs, rivers, dams, vegetation, soil). The *Component* element and its sub-elements are shown in Figure 8-17. Each *Component* element stores: (i) a unique ID for the component represented, (ii) a name for the component, (iii) the type of component being represented, and (iv) a configuration component ID. The component type is a reference to a *ComponentType* element (Appendix 8.5.2.2) in the *ModelConfiguration* schema, similar to the *ACRU* model's Component classes described in Section 3.2.2.3. The configuration component ID is a reference to a configuration *Component* element within the *ComponentConfiguration* element (Appendix 8.5.2.6) in the *ModelConfiguration* schema.

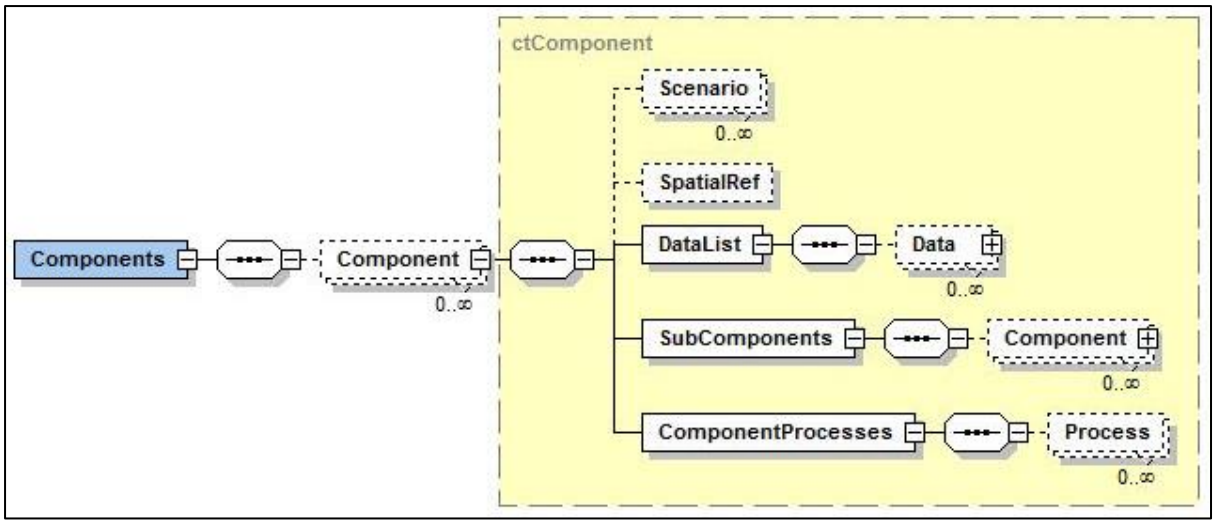

Figure 8-17 The *Components* element in the *ModelData* schema

A *Component* element may include zero or more component *Scenario* elements. These Component *Scenario* elements each contain two items of information: (i) a scenario ID which is a reference to one of the model *Scenario* elements, and (ii) 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 a 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.

Setting *Component* element scenarios would require corresponding *Relationship* element scenarios to be set.

The *SpatialRef* element within a *Component* element enables a spatial reference to be stored, for example, a spatial reference may refer to a feature in an ESRI shapefile or geodatabase. These component *SpatialRef* elements each contain two items of information, (i) a data reference ID which is a reference to one of the model *DataRef* elements (Appendix 8.5.1.6) which, for example, may store information for an ESRI shapefile, and (ii) an ID that will be used to identify a particular spatial entity within the spatial data reference, for example, a particular feature in an ESRI shapefile.

A modelled *ACRU* Component is described by data parameters and variables. The *ComponentElement* has a *DataList* element containing a list of *Data* elements. The *Data* element is described in Appendix 8.5.1.7.

The *SubComponents* element within a *Component* element contains a list of *Component* elements which are subcomponents of the parent *Component* element, for example a HRU or 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 an optional list of *Process* elements belonging to the parent *Component* element. The ordered list of *Process* elements contains information about which hydrological process algorithms are to be run for the parent *Component* element.

8.5.1.5 *ComponentRelationships* element

The physical components making up a hydrological system to be modelled, for example 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. Each *Model* element contains a *ComponentRelationships* element, shown in Figure 8-18, 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. As shown in Figure 8-18, 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 typically only be required when *Component* element scenarios are set.

Figure 8-18 The *Relationships* element in the *ModelData* schema

8.5.1.6 *DataReferences* element

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 datasets 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 user selected format. These considerations lead to the concept of data references in the *ModelData* schema. Each *Model* element contains a *DataReferences* element which contains a list of zero or more *DataRef* elements as shown in Figure 8-19. 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 dataset, for example “ACRU_SingleFormat” or “DelftFewsPiXml”. A *DataRef* element may also contain zero or more *Param* elements, where each *Param* element stores a 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 8-19 The *DataReferences* element in the *ModelData* schema

8.5.1.7 *Data* element

The *Data* elements used within the *ModelInfo* and *Component* elements are identical. A schema diagram of the *Data* element is shown in Figure 8-20. The *Data* element, in conjunction with the *ModelConfiguration* schema's *DataDef* element (Appendix 8.5.2.3), has been designed such that a particular model parameter or variable 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 hot-started.

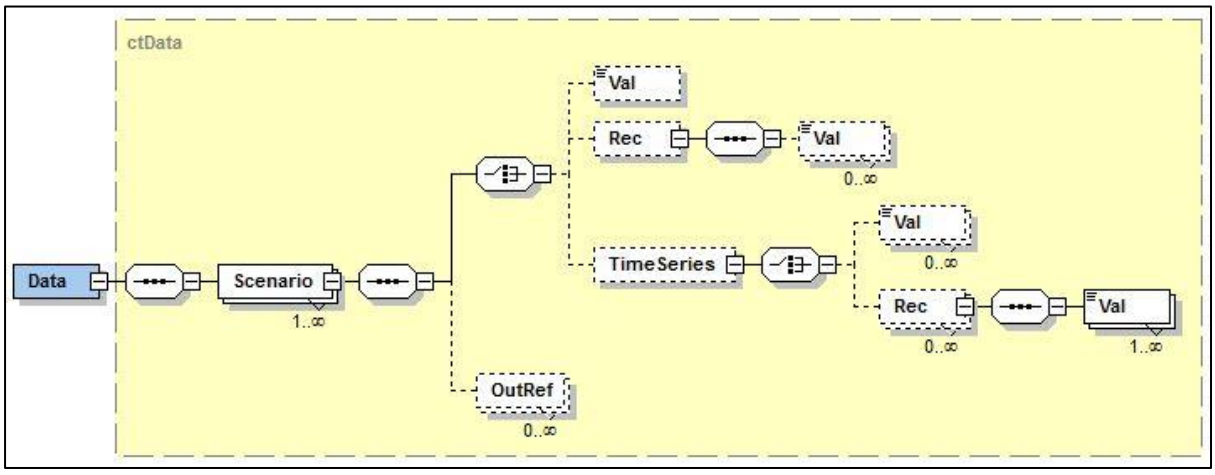

Figure 8-20 The *Data* element in the *ModelData* schema

Each *Data* element may 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 a single data value or a reference to a single 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 date/time values, for example "yyyy/MM/dd".

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 single data value. A *Scenario-Rec-Val* element stores a data value and also a key used to identify the data value within the set. For a 1-D array *Rec* the key would be an integer index, for a 2-D array *Rec* the key would be two comma separated integers (a row index and a column index) and for a dictionary *Rec* the key would be either an integer or a string. A *Scenario-TimeSeries-Val* element stores a single data value, a timestamp and an optional data quality flag. A *TimeSeries-Rec* element is similar to a *Scenario-Rec* element but in addition stores a timestamp 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.

8.5.2 *ModelConfiguration* Schema

The *ModelConfiguration* schema complements the *ModelData* schema by providing a data model for storing information describing permitted component configurations and relationships and also metadata type information about model parameters and variables for use in the *ACRU* model and associated software utilities such as the Configuration Editor. A large proportion of the information stored in a *ModelConfiguration* file is not required by the *ACRU* model but is required by software utilities used to display, edit and analyse data values stored in a *ModelData* file. A single implementation of the *ModelConfiguration* schema would be used for many catchment configurations of the *ACRU* model. A different implementation of the *ModelConfiguration* schema would only be required if changes were made to the *ACRU* model. An implementation 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. The root element of the *ModelConfiguration* schema, *ModelConfiguration*, and its sub-elements are shown in Figure 8-21.

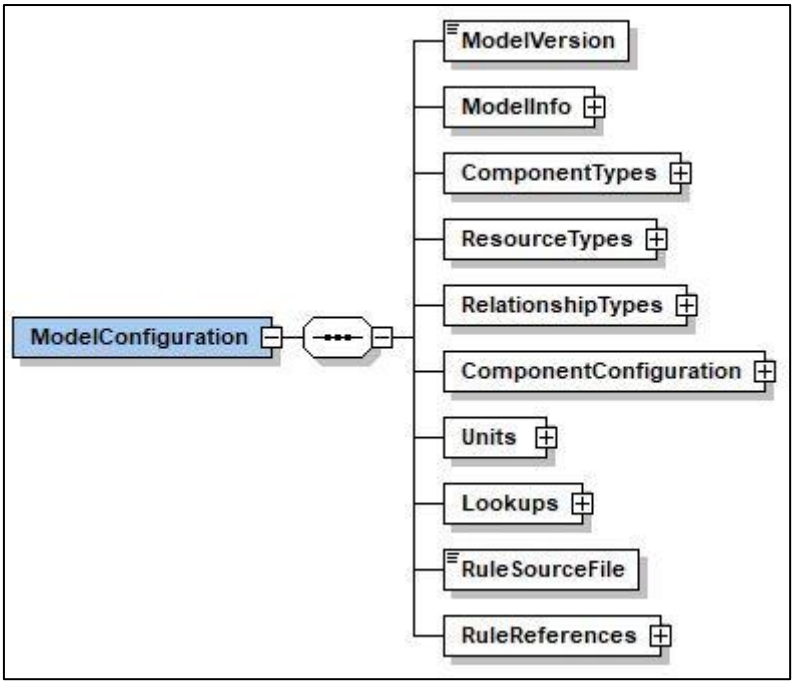

Figure 8-21 The *ModelConfiguration* element and main sub-elements of the *ModelConfiguration* schema

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

8.5.2.1 *ModelInfo* element

The *ModelInfo* element in the *ModelConfiguration* schema, shown in Figure 8-22, is related to the *ModelInfo* element in the *ModelData* schema. The *DataDefinitions* element contains a list of zero or more *DataDef* elements, described in Appendix 8.5.2.3, where each *DataDef* element contains metadata information about a general model option, parameter or variable. The *DataGroups* element contains a list of zero or more *DataGroup* elements where each *DataGroup* element contains an ordered list of *DataDef* IDs. Data groups provide a means of grouping model parameters or variables for display purposes in software utilities.

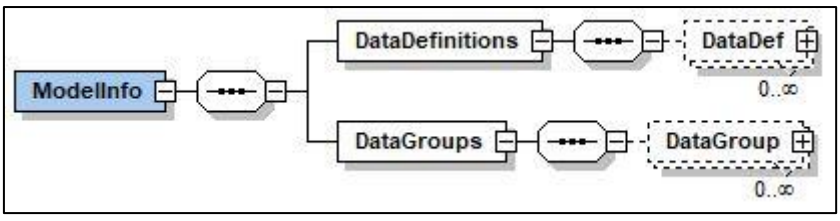

Figure 8-22 The *ModelInfo* element in the *ModelConfiguration* schema

8.5.2.2 *ComponentTypes* element

The *ComponentTypes* element, shown in Figure 8-23, 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 and soil horizons). Each *ComponentType* element stores: (i) a unique ID for the component type represented, (ii) a name for the component type, (iii) the name of the model Java class that is to be associated with the component type, and (iv) help text and description information for the component type. The *ComponentType* element defines a type of component by means of the parameters and variables describing its characteristics. These characteristics are defined by means of the *DataDefinitions* and *DataGroup* elements, described in Appendix 8.5.2.3, which work in the same way as the similarly named elements in the *ModelInfo* element. 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 flows.

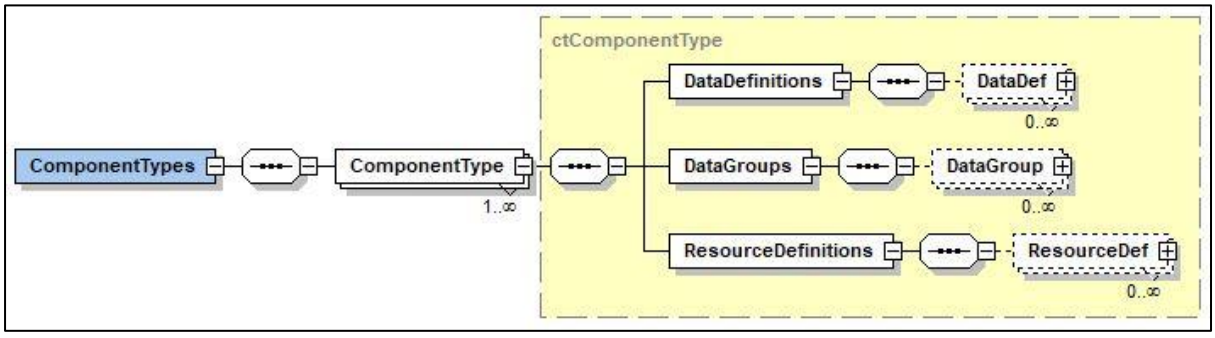

Figure 8-23 The *ComponentTypes* element in the *ModelConfiguration* schema

8.5.2.3 *DataDef* and *DataGroup* elements

The *DataDef* elements used in the *ModelInfo* and *ComponentType* elements are identical. The *DataDef* element, shown in Figure 8-24, contains several attributes, described in Table 8-4, which are used to define each data option, variable or parameter. 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 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, whether each data point is represented by an individual data value or a table of data values. 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 used as input.

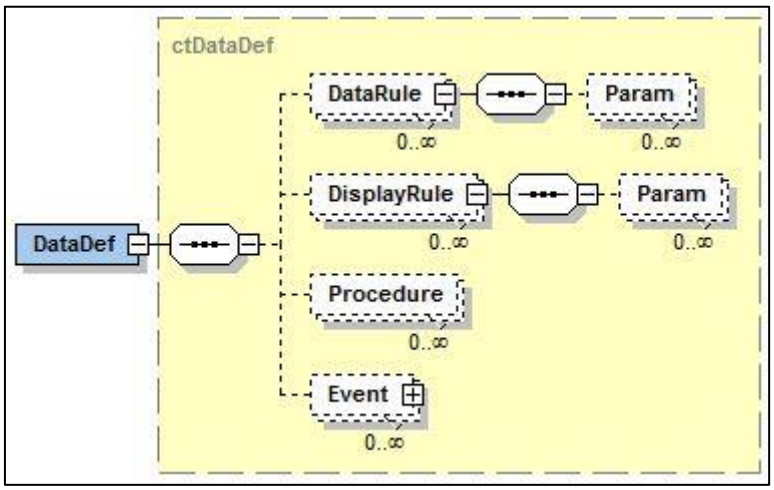

Figure 8-24 The *DataDef* element in the *ModelConfiguration* schema

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 (Appendix 8.5.2.9) is to be run and which model options or 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 options or parameters or variables. A *DisplayRule* element contains information about which software method in the rule source file (Appendix 8.5.2.9) is to be run and which other model options or parameters or variables may be required in determining whether to display the target parameter or variable.

Table 8-4 Attributes of the *DataDef* element in the *ModelConfiguration* schema

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, CublicSpline)
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 8-25 contains: (i) a unique ID for the data group, (ii) a name, (iii) a description, (iv) a parent data group, if applicable, and (v) a list of *DataDef* element IDs which belong to the group.

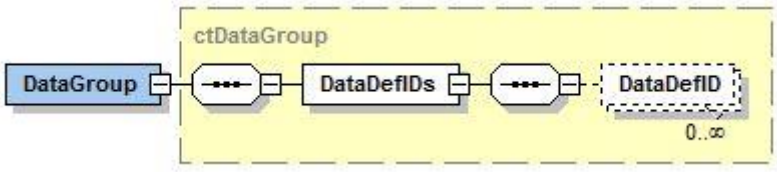

Figure 8-25 The *DataGroup* element in the *ModelConfiguration* schema

8.5.2.4 *ResourceTypes*, *ResourceDefinitions* and *ResourceDef* elements

The *ResourceTypes* element, shown in Figure 8-26, contains a list of zero or more *ResourceType* elements specifying the resource types that can be modelled, for example water. A *ResourceType* element stores a unique ID, name and description for the resource type, and also the name of the model Java class to be associated with the resource type. The *ResourceDefinitions* element, within a *ComponentType* element (Figure 8-23), contains a list of *ResourceDef* elements which are used to define the resource types, which can be modelled for that component type. Each *ResourceDef* element has a *ResourceTypeID* attribute, specifying the resource type, and contains one or more *Param* elements, each with an ID for the resource parameter and a reference to the ID of a *DataDef* element that stores the resource parameter values. For example, the ResourceDef for WATER in *ACRU* contains three *Param* elements: (i) *ID*=Quantity *DataID*=WATER, (ii) *ID*=Sources *DataID*=WATER_S, and (iii) *ID*=Destinations *DataID*=WATER_D.

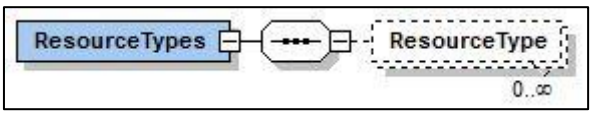

Figure 8-26 The *ResourceTypes* and *ResourceType* element in the *ModelConfiguration* schema

8.5.2.5 *RelationshipTypes* element

The *RelationshipTypes* element, shown in Figure 8-27, contains a list of zero or more *RelationshipType* elements specifying the relationship types that may be used in a *ModelData* file. 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 RiverA is on the left-hand side of the relationship and RiverB is on the right-hand side of the relationship, therefore RiverA is upstream of RiverB and RiverB is downstream of RiverA.

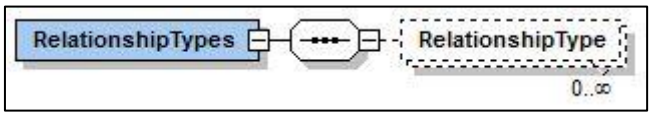

Figure 8-27 The *RelationshipTypes* and *RelationshipType* elements in the *ModelConfiguration* schema

8.5.2.6 *ComponentConfiguration* element

The *ComponentType* element, shown in Figure 8-23, 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 8-28 is used for this purpose. The *ComponentConfiguration* element contains three sub-elements *Components*, *PermissibleRelationships* and *AutomaticRelationships*.

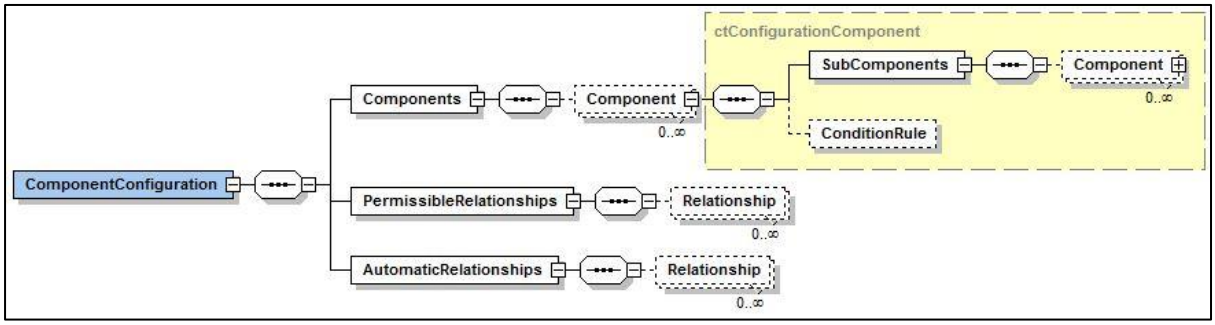

Figure 8-28 The *ComponentConfiguration* element in the *ModelConfiguration* schema

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 8-17. Each configuration *Component* element stores: (i) a unique ID for the configuration component, (ii) a name for the configuration component, (iii) the component type ID, (iv) the minimum and maximum permitted occurrences of the configuration component within its parent configuration component, and (v) 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.

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.

8.5.2.7 *Units* element

The *Units* element, shown in Figure 8-29, consists of a list of 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.

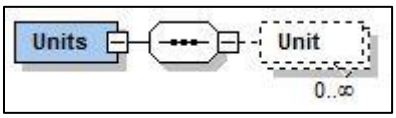

Figure 8-29 The *Units* and *Unit* elements in the *ModelConfiguration* schema

8.5.2.8 *Lookups* element

The *Lookups* element, shown in Figure 8-30, contains a list of zero or more *Lookup* elements. It is used to store lookup lists for model parameters, typically model option parameters, which have a finite number of permissible parameter values. Each *Lookup* element contains a list of *LookupItem* elements each of which of which contains ID, name and description information about an individual lookup item.

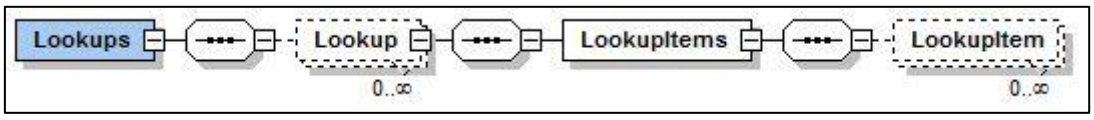

Figure 8-30 The *Lookups* element in the *ModelConfiguration* schema

8.5.2.9 *RuleSourceFile* and *RuleReferences* elements

The *RuleSourceFile* element stores the name of a text file containing the source 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. For the *ACRU* model, a file named Ruleset.cs, written in the C# programming language, contains a class named *RuleSet* which contains a public method for each data rule and each display rule. This *RuleSet* file is not pre-compiled, for ease of reference and editing, and is compiled on the fly, for example within the Configuration Editor software. The *RuleReferences* element stores references to software libraries, usually Dynamic Link Libraries (DLLs), that are required by the file specified in the *RuleSourceFile* element to enable this file to be compiled on the fly.

8.5.3 *XmlModelFiles* Libraries

XML is useful for structuring data files used to configure hydrological models, but although they are text-based, they are difficult to edit. For this reason the *XMLModelFiles* software library was developed to facilitate easier 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 *XMLModelFiles* library consists of three packages: *XmlModelFiles.ModelData*, *XmlModelFiles.ModelConfiguration* and *XmlModelFiles.ModelRules*. The elements within an XML file are in most cases represented by a matching code object. The process of saving the information within a code object to XML is referred to as serialization and the reverse operation is referred to as deserialization. A simplified UML class diagram of the main classes in the *XmlModelFiles.ModelData* package is shown in Figure 8-31. The *ModelDataDocument* class represents a *ModelData* file and contains code to serialize and deserialize between *ModelData* files and their matching code objects. The *ModelElement*, *ModelRunElement*, *ScenarioElement*, *ModelInfoElement*, *ComponentElement*, *RelationshipElement* and *DataRefElement* classes represent the corresponding elements in the *ModelData* schema. The *IDataContainer* interface enables the *ModelInfoElement* and *ComponentElement* classes can be referred to in a generic manner when working with the *DataElement* class. The *DataElement* and *DataScenarioElement* classes represent the *Data* and *Scenario* elements in the *ModelData* schema. The *DataValElement*, *DataRecElement* and *DataTSElement* classes, together with their associated classes, represent the *Val*, *Rec* and *TimeSeries* elements in the *ModelData* schema, and are used to

store the actual data values for a modelling scenario. Classes named *DataRule*, *DisplayRule*, *DataProcedure* and *DataEvent* (not shown in the figure) were created to contain code used to configure and run data and display rules based on the information stored in the *DataDef* elements of a *ModelConfiguration* file.

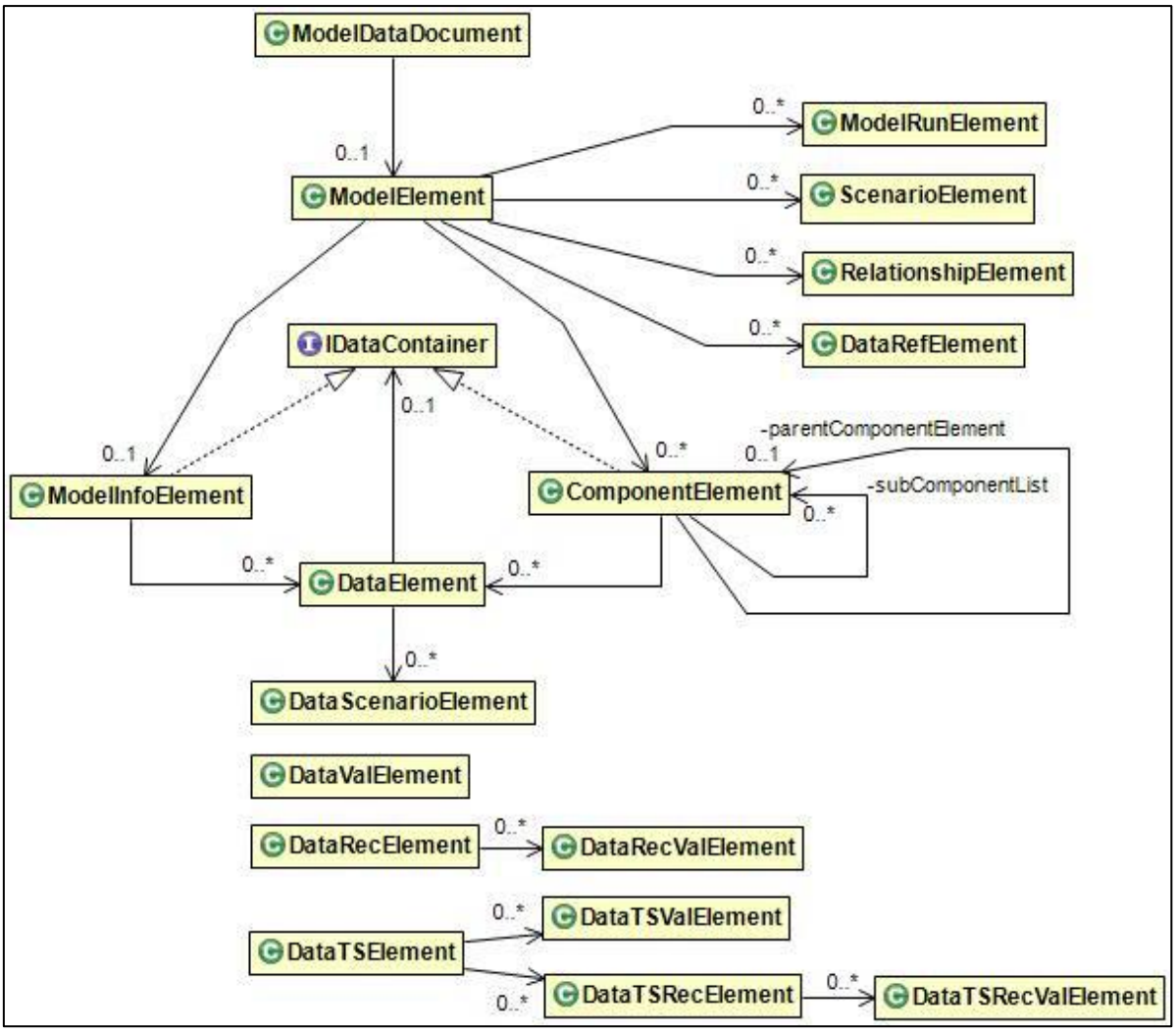

Figure 8-31 Simplified UML diagram of the *XmlModelFiles.ModelData* package

A simplified UML diagram of the main classes in the *XmlModelFiles.ModelConfiguration* package is shown in Figure 8-32. The *ModelConfigurationDocument* class represents a *ModelConfiguration* file and contains code to serialize and deserialize between XML elements in the *ModelConfiguration* files and their matching classes. The other classes represent the corresponding elements in the *ModelConfiguration* XML 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.

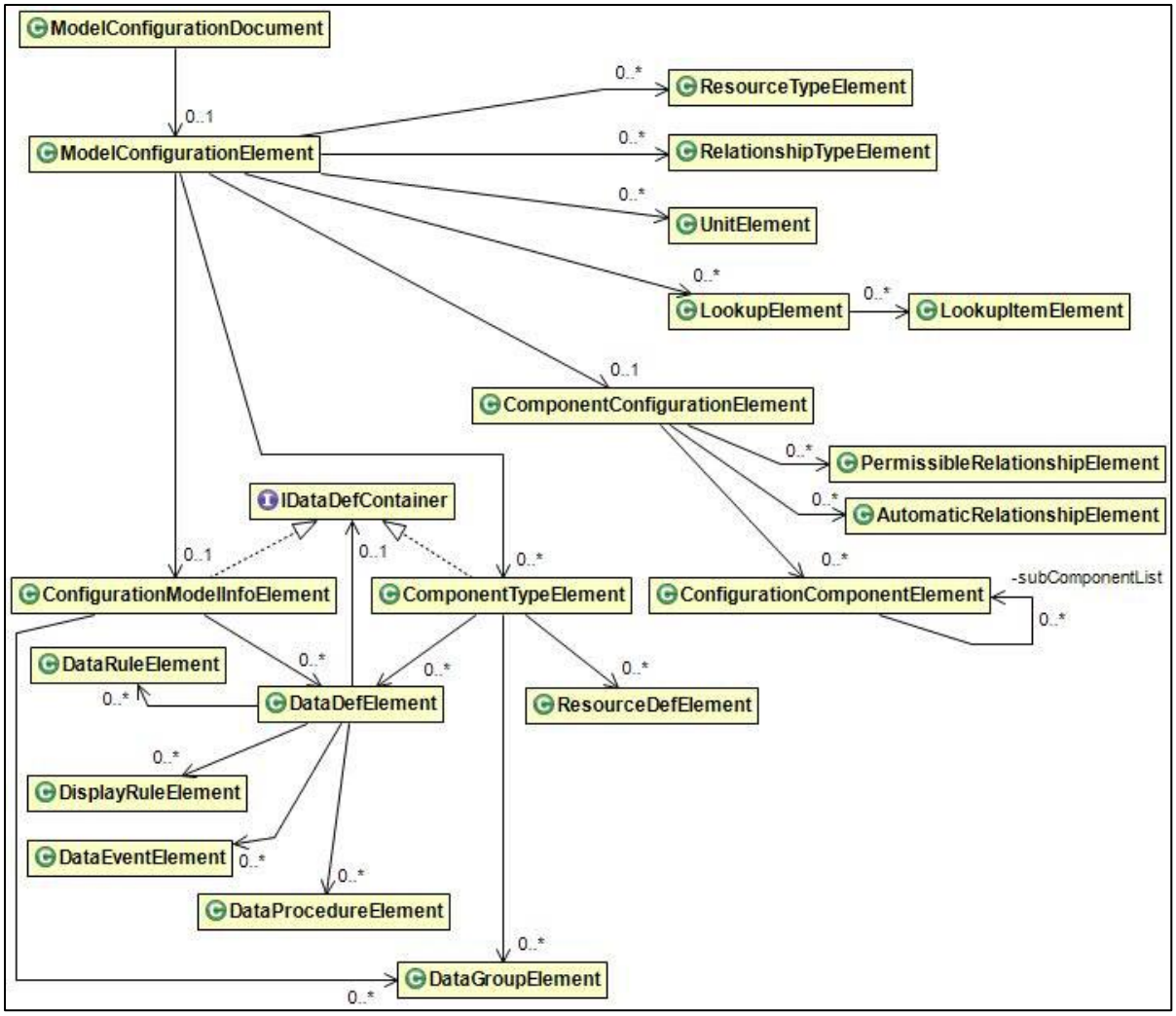

Figure 8-32 Simplified UML diagram of the *XmlModelFiles.ModelConfiguration* package

As described in Section 8.5.2.9, the *RuleSourceFile* element in the *ModelConfiguration* schema stores the name of text file containing the source 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 *XmlModelFiles.ModelRules* package contains several utility classes used to compile and execute the code for the data and display rules stored in a rule file. These classes were not developed as part of this PhD study, but were consolidated into this library as they had previously formed part of the Configuration Editor.

8.5.4 *ModelDataAccess* Library

The *ACRU* 3 version of the model provided three different *ACRU* specific text formats by which time series data could be provided as input to the model, namely the Single format, the Composite format and the CompositeY2K format (same as the Composite format but with four digit numbers for years). These three formats were not easy to create using text editors or spreadsheets and both the Composite format and the CompositeY2K format could only contain specific *ACRU* input variables. Time series of simulated output data from the *ACRU* 3 version of the model was written to an *ACRU* specific binary format.

The *ACRU* 2000 version of the model also enabled the selection of the dBase IV (dbf) format for model output time series data. The *ACRU* 4 version of the model, through the XML model input files, enabled more flexible configuration of which file and file format each model variable read data from or wrote data to. As part of the development of the SPATSIM-HDSF modelling framework, code was written to enable the *ACRU* 4 version of the model to read from and write to SPATSIM-HDSF databases in Microsoft Access (Clark *et al.*, 2009; Clark *et al.*, 2012a). In addition, a software library named *ModelDataAccess* was developed for the .Net platform, containing a set of classes to read and write data from and to the existing *ACRU* input and output formats, where each of these reader/writer classes implements a common interface named *IDataReaderWriter* to enable easier integration of new formats into the system (Clark, 2013). The Java code to read and write the existing formats was previously part of the *ACRU* model code but was subsequently consolidated into a Java version of the *ModelDataAccess* library as a set of reader/writer classes.

Subsequently, as part of this study, two additional data formats were included into the *ModelDataAccess* libraries: (i) an *ACRU* specific comma separated value (CSV) format, and (ii) the Delft-FEWS PI XML format. The *ACRU* CSV format can contain time series data for any variable, enables easy data visualisation and editing in text editors and spreadsheets, and can include information such as units of measure and data type. The Delft-FEWS software (Werner *et al.*, 2013; Deltares, 2018) is an open platform through which data and models can be flexibly integrated to construct operational forecasting systems. The recent addition of the Delft-FEWS PI XML format, as described briefly in Appendix 8.5.5, enables the *ACRU* model to be run as a model in the Delft-FEWS framework using the Delft-FEWS *General Adapter*. Thus, *ACRU* can be run as part of a Delft-FEWS setup using historical or forecast time series data stored by Delft-FEWS as input and saving *ACRU* simulated outputs back into Delft-FEWS for analysis and visualisation or as input to another model.

8.5.5 Integration of the *ACRU* Model With Delft-FEWS

The Delft Flood Early Warning System (Delft-FEWS), developed by Deltares in the Netherlands, is described as an "*Expert data handling and model integration software for flood forecasting, drought and seasonal forecasting, and real-time water resources management*" (Deltares, 2018). More information is available in Werner *et al.* (2013) and at http://oss.deltares.nl/web/delft-fews/. Delft-FEWS is a system that facilitates data handing and model integration enabling users to build their own custom modelling systems. For each application of Delft-FEWS, users decide which model or models need to be implemented and the datasets required to run these models. External models are usually linked into and executed from DELFT-FEWS using the *General Adapter* module. Prior to running a model the user would execute one or more Delft-FEWS *Workflows* to import the data required for modelling into a Delft-FEWS database. For each model implementation a *Workflow* in Delft-FEWS would be used to execute an instance of the *General Adapter* for the model. Delft-FEWS already contains pre-built adapters for a variety of models and also adapters to read and write a variety of commonly used data formats. To integrate other models into Delft-FEWS, where the model user does not have access to source code for a model, it would be necessary to develop a pre-adapter and a post-adapter to enable the transfer of data between Delft-FEWS and the model.

In the case of the *ACRU* model, it was possible to omit the pre-adaption and post-adaption phases of linking and executing the model by adding functionality to the *ACRU* model to read from and write to the Delft-FEWS Published Interface XML (PI XML) data exchange files directly. Two new classes were developed and incorporated into the *ModelDataAccess* library for the *ACRU4* and *ACRU* 5 versions of the model: (i) the *ADelftFewsPiXmlFileReader* class which reads a PI XML file, containing daily time series input data, into *ACRU* and, (ii) the *ADelftFewsPiXmlFileWriter* which writes daily time series of *ACRU* simulation results to a PI XML file. These two new classes are shown in the Unified Modelling Language (UML) class diagram in Figure 8-33. The linking and execution of the *ACRU* model in Delft-FEWS in shown in Figure 8-34. A simple Delft-FEWS *Workflow* can be used to run the *ACRU* model, using historical time series of rainfall data imported into Delft-FEWS, and then to import simulated streamflow time series from *ACRU* back into Delft-FEWS.

The successful development of these reader/writer classes for the *ACRU* model to read and write PI XML files, enables the *ACRU* model be used as a model within Delft-FEWS. However, further investigation is required into the use of *ACRU* together with Delft-FEWS to

provide agrohydrological forecasts. One important point to note is that, although two or more models may be run from Delft-FEWS such that the output from one model is used as input to another model, these models are run in series. The implication of this is that any feedbacks, between different components of the modelled system represented by different models, cannot be modelled unless each model is run one timestep at a time.

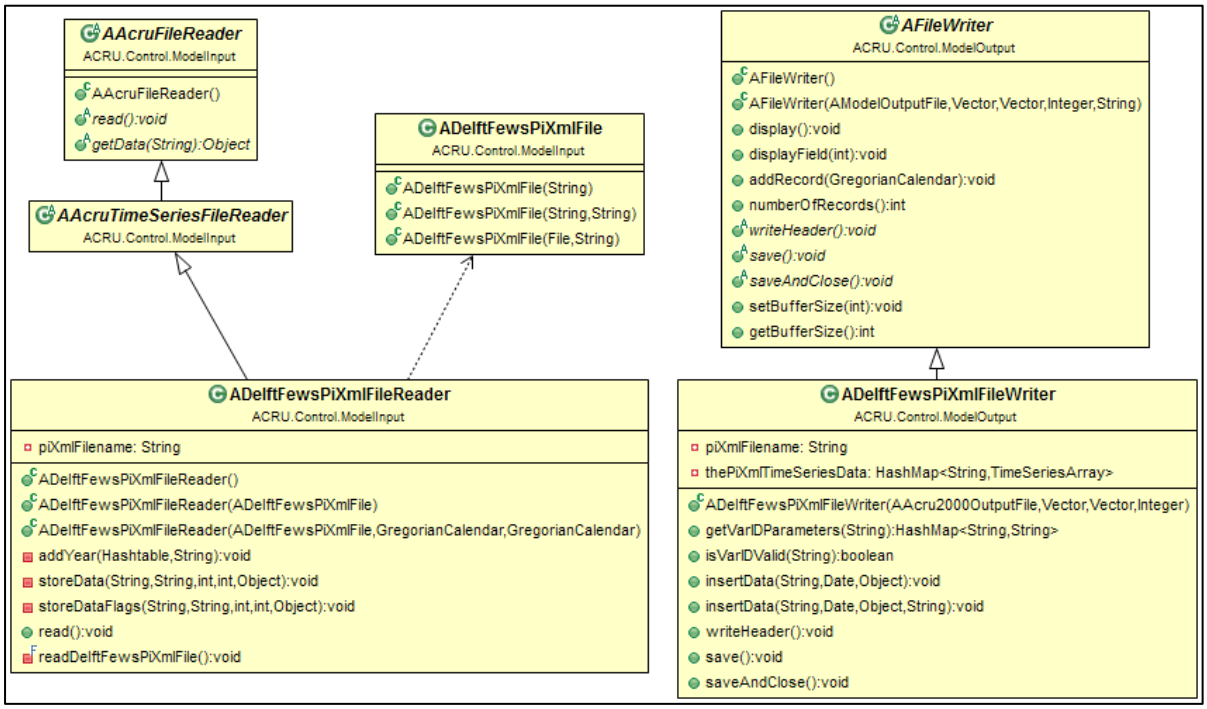

Figure 8-33 UML diagram of new *ACRU* classes developed to read and write PI XML files

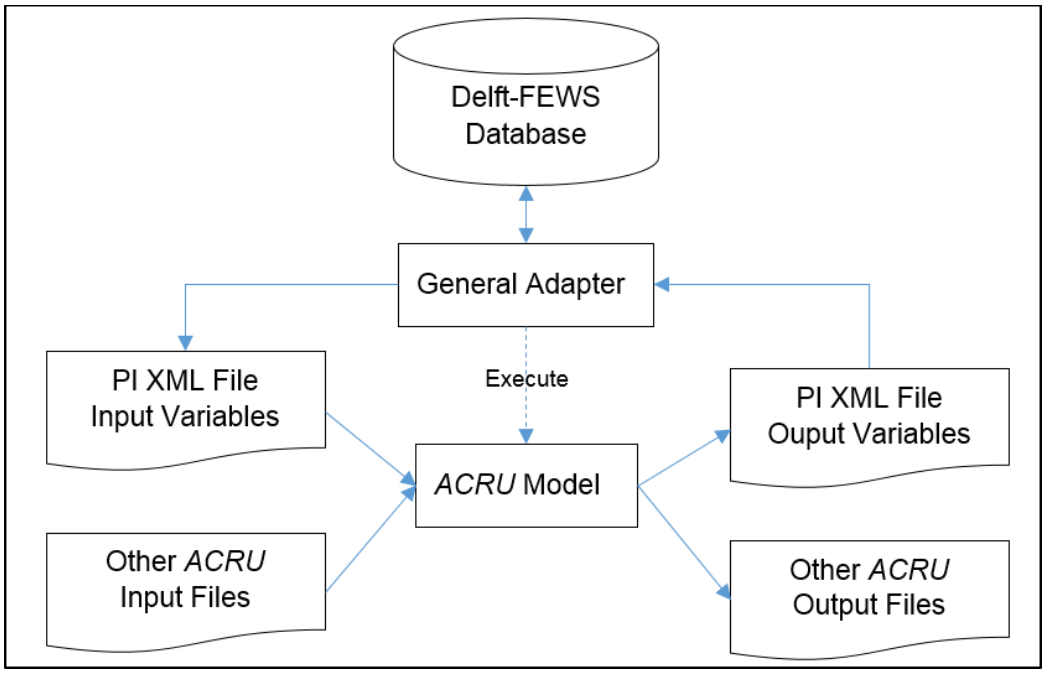

Figure 8-34 Linking and execution of the *ACRU* model in Delft-FEWS

8.6 Development of OpenMI Composition Tools

In OpenMI two or more linkable components can be connected to create an OpenMI 'composition'. One of the tools developed in the FluidEarth project was Pipistrelle, shown in Figure 8-35, which is a graphical user interface that can be used to create and then run compositions of linked models. To be able to add a linkable component to a composition an OMI-file (*.omi) must be created. An OMI-file is a standard OpenMI file in XML format which contains information about (OpenMI Association, 2010d): (i) the linkable component class for the model to be instantiated, (ii) the compiled library containing the linkable component class, and (iii) model specific arguments providing information required to configure the model and initialise the linkable component. The linkable components are then added to Pipistrelle using these model specific OMI-files which are displayed as shown in the example in Figure 8-35. The connections and associated output adapters between the two linkable components are then configured using the tools provided in Pipistrelle. Finally a 'trigger' or 'pull' variable is selected for one of the models (the *eWater Source – uMngeni_Upper* model in this example) through which the linked simulation will be initiated. A Pipistrelle composition may be saved to a composition file (*.chi) which stores the composition information in an XML format.

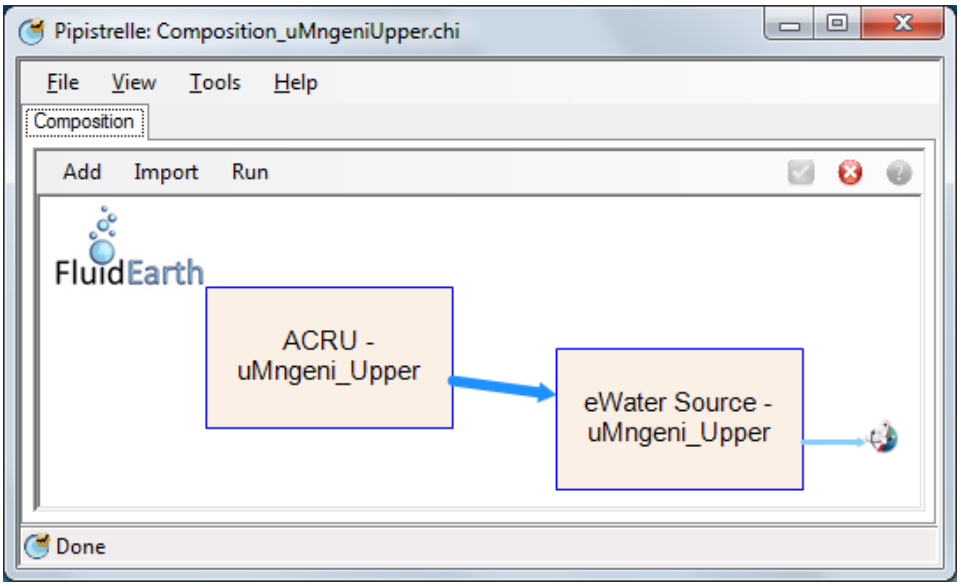

Figure 8-35 The Pipistrelle tool for creating compositions of linked models

Pipistrelle is a useful tool for creating and editing compositions with a relatively small number of connections and adapters between the models. However, this would be a tedious exercise if there were a large number of connections and adapters to be configured, as was the case with the case study in the upper uMngeni Catchment (Chapter 5) for the linked

ACRU and eWater Source models. Initially it seemed that it would be best to write some code to automate the population of a Pipistrelle composition file. However, these files were found to be difficult to work with. This lead to the development of what were termed OpenMI 'composition information' files OCI-files (*.oci) for use in this study. These OCI-files are also in XML format but are simpler than the CHI-files and easier to work with. A XML schema diagram for these OCI-files is shown in Figure 8-36. The OCI-files, which contain a small set of XML elements and attributes, are used to store information about linkable components, adapter factories, adapters, connections and the pull variable. A corresponding set of C# classes, shown in Figure 8-37, were created to programmatically create and edit these OCI-files. The main *CompositionInfo* class also contains code that enables a CHI-file to be generated. The *CompositionInfo_OMI* class, which is an alternative to the *CompositionInfo* class, enables existing OMI-files to be used to store information about the linkable components.

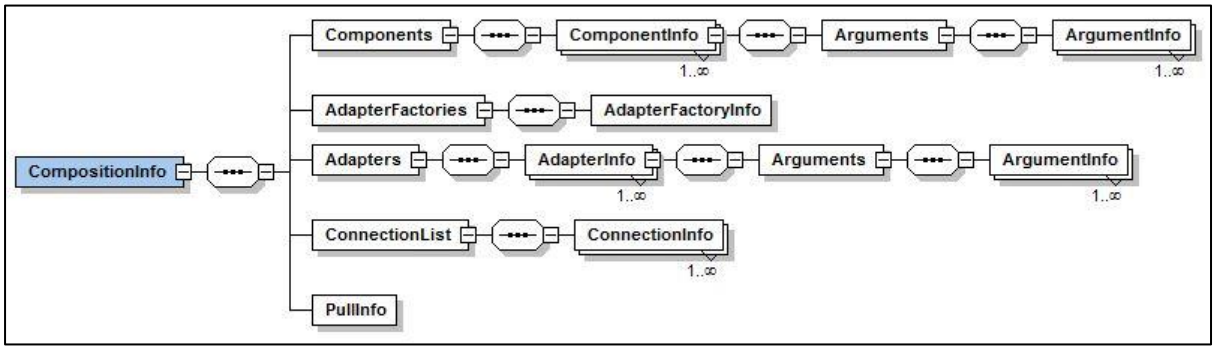

Figure 8-36 XML schema diagram for the OpenMI composition information files

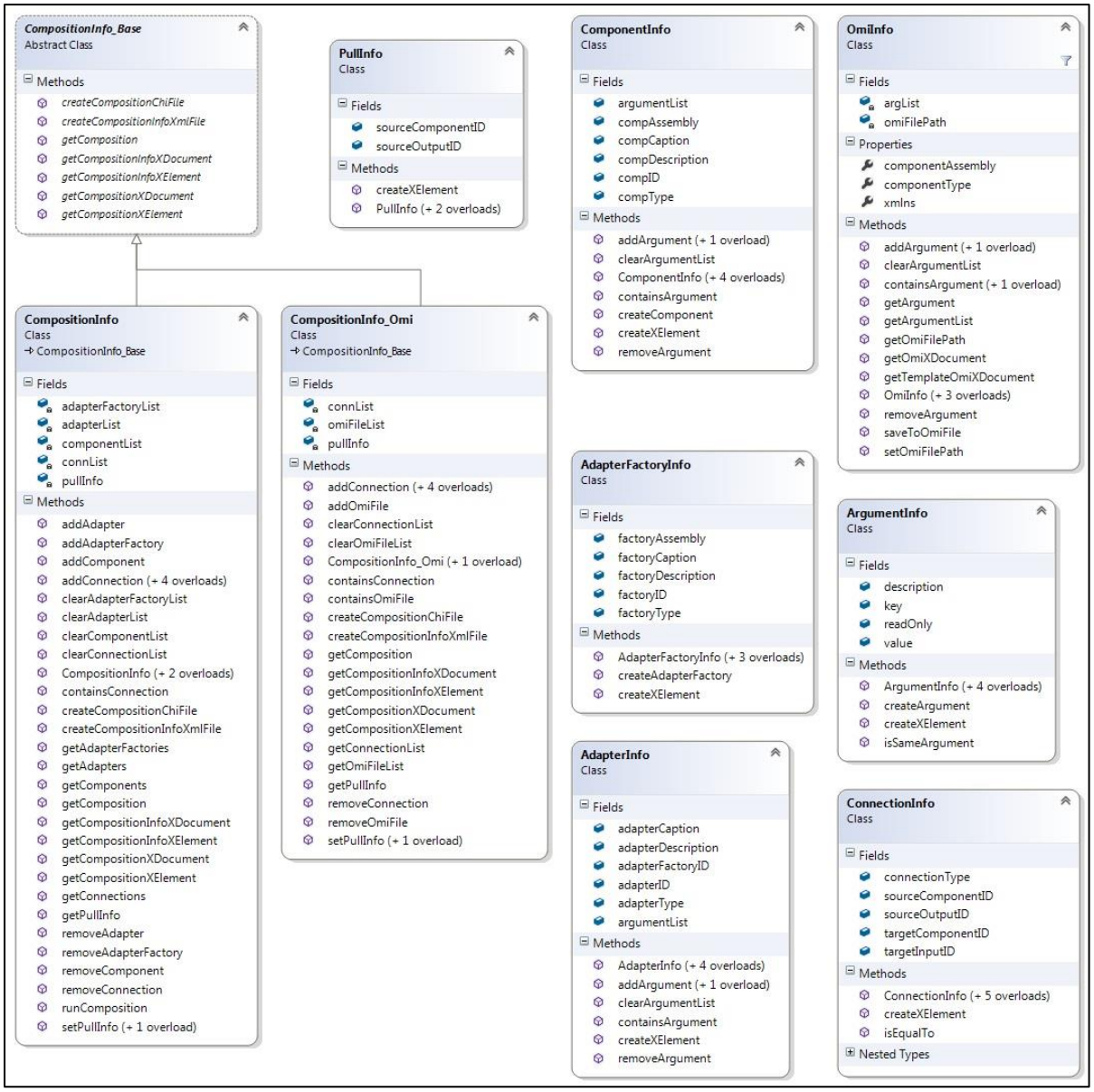

Figure 8-37 UML class diagram of classes created to work with the OpenMI composition information files

8.7 Development of Tools to Configure the Linked *ACRU* – eWater Source Models

The graphical user interface for eWater Source, shown in Figure 8-38, provides tools that enable model users to set up a flow network consisting of different types of nodes connected by links, and then to configure these nodes and links with variable and parameter values. Most of the flow network information required to configure eWater Source could be obtained from an *ACRU* model input file. Thus, several new C# classes and forms were developed to: (i) create a new eWater Source scenario, (ii) create the flow network within this scenario, (iii) configure the network nodes and links, and (iv) to run the linked models.

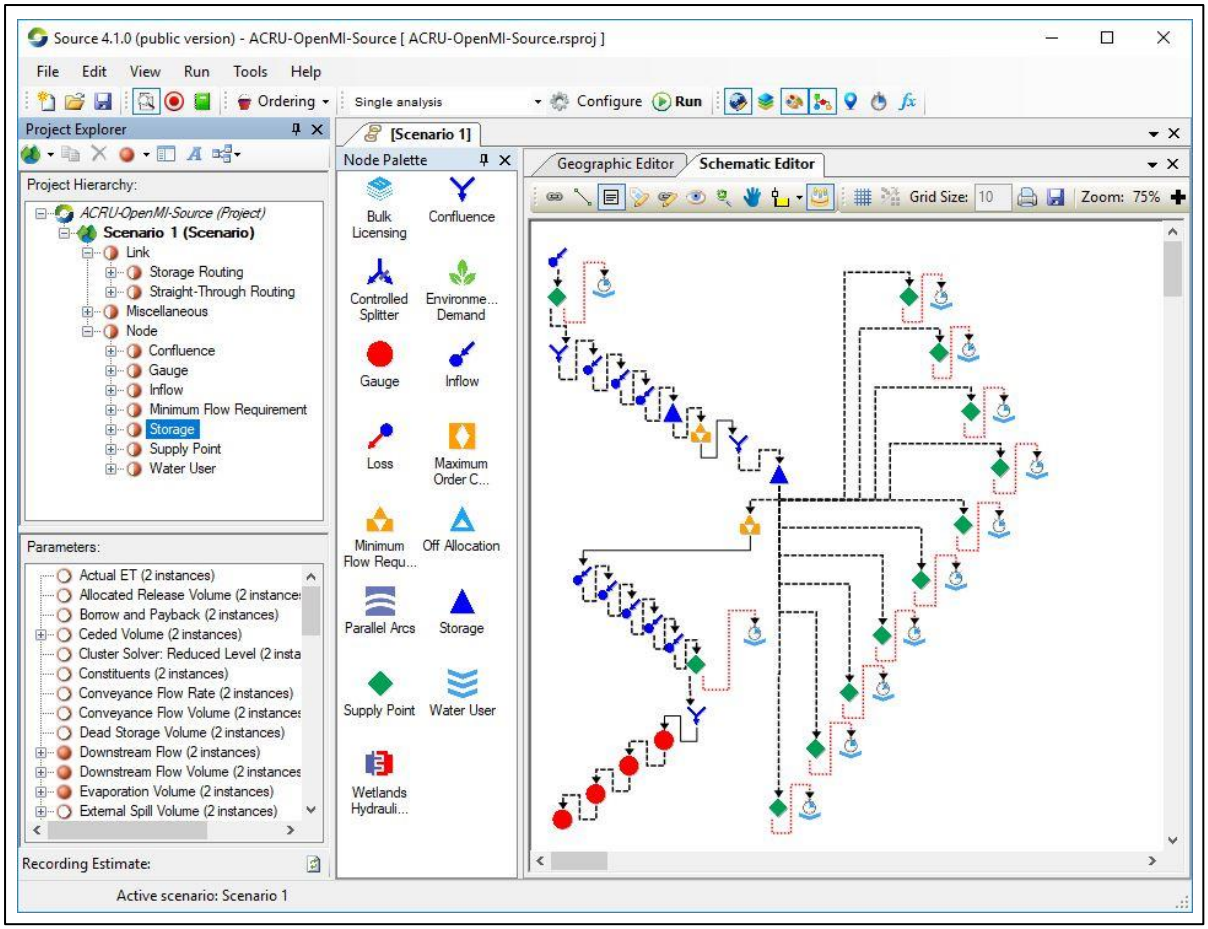

Figure 8-38 The eWater Source graphical user interface (eWater CRC, 2017)

When a new eWater Source project is created the project creation form enables a scenario creator to be selected. A new scenario creator named *ACRU Scenario Creator* was developed to create a scenario based on information imported from an *ACRU* input file. This new form implements the *RiverSystem.Controls.Interfaces.IScenarioCreation* interface, and can be included in eWater Source as one of the scenario creation options. The new *ACRU Scenario Creator* form is shown superimposed on the eWater Source project creation form in Figure 8-39. The form enables the user to select the *ACRU* input file and model run to be imported, and also whether eWater Source is to be run at a daily or an hourly timestep.

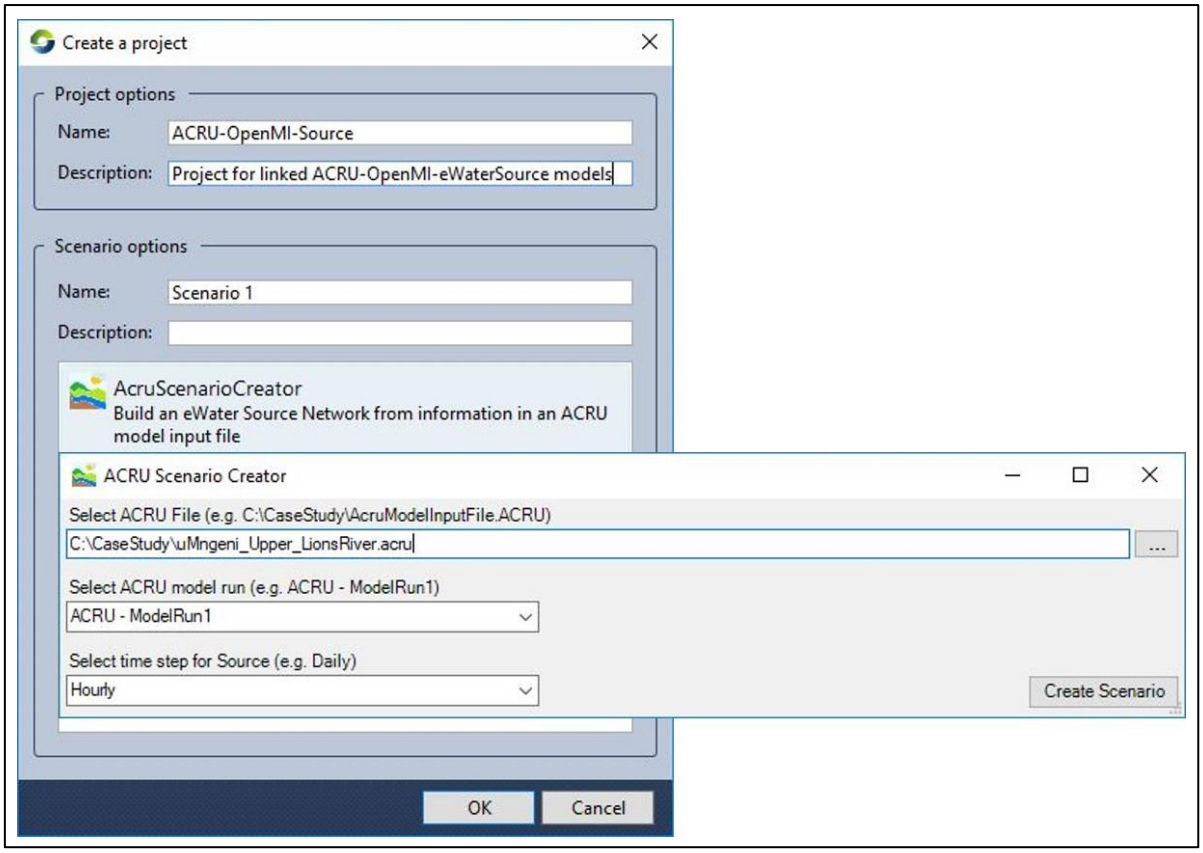

Figure 8-39 The *ACRU* Scenario Creator form in eWater Source

A class named *AcruNetworkScenarioImporter* was created to read a specified *ACRU* input file and configure a new eWater Source scenario. The configuration of eWater Source included the creation of all the necessary *Inflow* nodes, *Storage* nodes (dams), *Minimum Requirement* nodes, *Supply Point* nodes, *Time Series Demand* water user nodes, interconnecting links without flow routing (*Straight-Through Routing* links) and river links with flow routing (*Storage Routing* links). eWater Source *Functions* were created for the *Inflow* nodes and *Time Series Demand* nodes to facilitate OpenMI connections with *ACRU*. All the information required to configure the dam *Storage* nodes was obtained from the *ACRU* input file, including dam geometry, spillway rating curve, seepage losses and flow releases. The dam seepage losses and flow releases from dams were modelled using *Minimum Requirement* nodes. Similarly many of the river reach characteristics required for flow routing were obtained from the *ACRU* input file, which for the upper uMngeni case study, were estimated as described in Appendix 8.9.3. As part of the import process an OCI-file was automatically generated with the *ACRU* and eWater Source models as linkable components and all the necessary connections and output adapters. As part of the import it was necessary to add some of the *ACRU* rainfall and reference potential evaporation time series data files as data sources in eWater Source. Thus, two additional classes named *AcruCsvIO* and *AcruCompositeY2kIO* were created by extending the

TIME.DataType.IO.MultiTimeSeriesIO class to enable eWater Source to read the data from the *ACRU* time series data files. A UML class diagram for these three classes is shown in Figure 8-40.

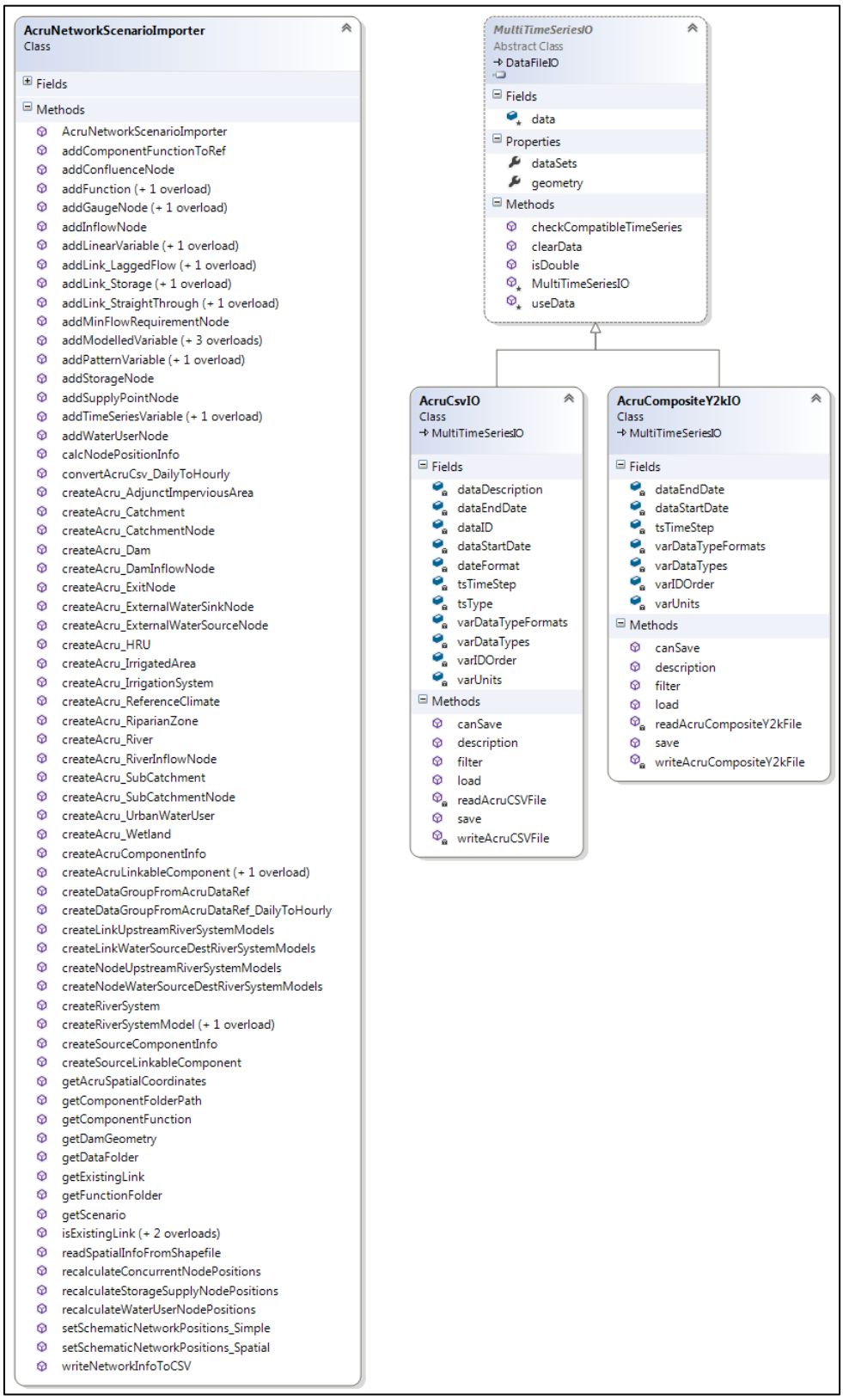

Figure 8-40 UML class diagram for the *ACRU* – eWater Source configuration tools

The *OpenMI Composition Tools* form, shown in Figure 8-41, was developed as a plugin for eWater Source. This form enables users to select an OpenMI composition specified in an OCI-file and then run the OpenMI composition. There is also an option on this form to create a CHI-file for the selected OpenMI composition.

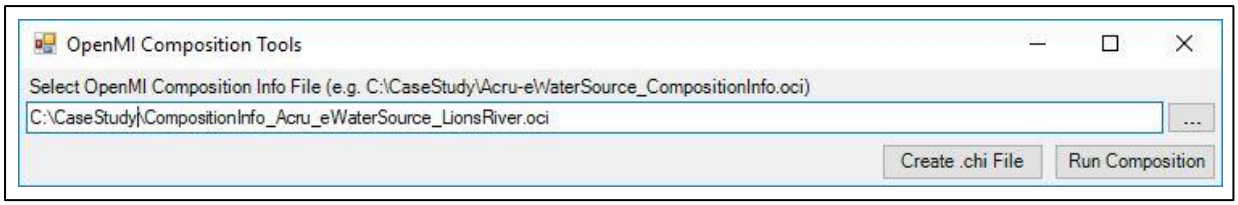

Figure 8-41 The OpenMI Composition Tools form in eWater Source

8.8 Development of a Water Use Quantification and Accounting System for South Africa

In a research project titled "*Development And Assessment Of An Integrated Water Resources Accounting Methodology For South Africa*" (Clark, 2015a) a methodology was developed to produce annual catchment-scale water resource accounts for South Africa. The methodology is described in Clark (2015b), but an overview is provided in this section as the configuration of *ACRU* for the case study in the upper uMngeni Catchment in Chapter 5 was based on this methodology.

The compilation of water resource accounts is data intensive and much of the data are variable in both space and time. Some of the data are available from ground-based measurements such as rainfall, reference potential evaporation and streamflow. Water resources modelling is also data intensive, but can also be an invaluable tool for estimating water resource quantities in ungauged catchments and for estimating water resources variables that cannot be easily measured. An investigation into the water resource related datasets available in South Africa, and a review of water use quantification methodologies previously applied in South Africa helped to guide the development of the methodology. The water resource related datasets investigated included catchment boundaries, altitude, climate (rainfall, evaporation, air temperature), land cover/use, soil hydrological characteristics, dams, rivers, measured streamflow, abstractions, return flows, transfers and reserved flows.

The following key decisions guided the development of the methodology:

- The WA+ water accounting system would be used, with an initial focus on the Resource Base Sheet, and only after that the Utilized Flows Sheet.
- A hydrological modelling approach using the *ACRU* agrohydrological model would be the most suitable. The hydrological modelling approach was selected as there are many components of the water resource accounts which cannot be easily measured, either directly or by remote sensing. A daily physical conceptual model, such as *ACRU*, enables the natural daily fluctuations in the water balance of the climate/plant/soil continuum to be represented and ensures internal consistency through the modelled feed-forwards and feedbacks between the various components of the hydrological system.
- Remotely sensed data products would be considered as potential sources of data for hydrological modelling.
- To produce annual water resource accounts at a Quaternary Catchment scale, although the hydrological modelling should be done at a suitable finer spatial scale to represent variations in climate and sectoral water use within a Quaternary Catchment.
- The methodology should make it possible to aggregate accounts up from finer to coarser spatial and temporal scales.
- Consistent with WA+, the methodology should have a strong land cover/use basis, to enable assessment of sectoral water use.
- To initially focus on rainfall and total evaporation estimates at a catchment scale.
- To initially focus on water quantity, but to anticipate that water quality and economic aspects of water resources would be important additional components of the accounts in the future.

As stated, the methodology was intended to have a strong land cover/use focus. There are various land cover/use datasets available for different regions and points in time and these all use different land cover/use classifications. Thus, some means was required to provide consistency in the application of these various datasets and ensure compatibility between water resource accounts compiled using different datasets. This led to the development of a standard hierarchy of land cover/use classes and an associated database of land cover/use classes containing information describing the hydrological characteristics of these classes. The land cover/use classes and hierarchy were used in a Python script that was developed to determine HRUs based on catchment boundaries, land cover/use, previously existing natural vegetation and soils datasets.

The poor spatial representation and poor availability of rain gauge data led to the investigation of remotely sensed rainfall datasets. Four remotely sensed daily rainfall datasets (CMORPH, FEWS ARC 2.0, FEWS RFE 2.0 and TRMM) were compared with rain gauge data and the simulated streamflow resulting from the use of these rainfall datasets was compared with measured streamflow. The investigation of remotely sensed rainfall datasets is discussed further in Appendix 8.9.9.2, Clark (2015b), Clark (2016) and Clark (2018). Although remotely sensed rainfall offers advantages in spatial representation and availability, the coarse resolution and bias in rainfall quantities may be a problem in accurately estimating rainfall at sub-Quaternary scale for use in water resource accounts.

Naturally vegetated, cultivated and water body land cover/use classes together typically cover the largest portion of a catchment and are the easiest to represent in a hydrological model for a large number of catchments. Datasets quantifying water use and return flows for urban and mining land cover/use classes are often harder to access and more difficult to model. Urban residential water use was estimated in a simple manner based on population and assumptions regarding per capita water use and return flows. Industrial, commercial and mining water use was not taken into account.

As part of the methodology a project database spreadsheet was developed, in which the spatial configuration of catchments, subcatchments, HRUs, river flow network, dams and other water infrastructure is specified. This spreadsheet acts as a structured source of information from which the *ACRU* model, and potentially other hydrological models can be configured. This project database also aims to make catchment configuration more transparent, editable and reproducible, though implementation by individual models will require different model specific assumptions. A library of Python scripts was developed to process datasets and to populate the project database spreadsheet. Java code was also developed to use the information contained in the project database spreadsheet and associated datasets to configure the *ACRU* model. The workflow to compile water resource accounts using the methodology includes the following main components: (i) processing of datasets, (ii) compilation of a project database spreadsheet containing catchment configuration information, (iii) configuration of *ACRU* using the project database and associated datasets, and (iv) simulation using *ACRU* and compilation of accounts.

The *ACRU* model was further developed to compile the modified WA+ Resource Base Sheets and store the information required to populate the Land And Water Use Summary table. The new source code is contained in the *ACRU.Processes.Accounting* package and includes classes to: (i) store monthly and annual volumes of water stocks and flows for each

catchment, subcatchment, HRU, dam, river and water user, and (ii) the *PWaterAccount* class which creates instances of the Resource Base Sheet for each month and year for each subcatchment and catchment modelled. It was initially intended that this source code which was added to *ACRU* to compile the modified WA+ Resource Base Sheets would be a prototype from which a standalone tool, outside of *ACRU*, would be developed. However, it was found that although the stocks and flows within a catchment were relatively easy to keep track of, the flows between catchments were more difficult to keep track of, especially when aggregating up to bigger catchments. Thus, it was concluded that it would be easier to keep this functionality within *ACRU* which, partly as a result of the object-oriented structure, makes it easier to determine the source and destination catchments to include transfers into and out of catchments in the accounts. The Resource Base Sheet accounts were also found to be a very useful summary of the simulated model output from *ACRU*, as they show not only the simulated streamflow at the exit of the catchment, but also the inflows, consumption and changes in storage within the catchment.

8.9 Case Study - Configuration of the *ACRU* and eWater Source Models

This section describes the data and information used to configure the *ACRU* and eWater Source models for the upper uMngeni Catchment. The methodology and datasets described in Clark (2015b) and Clark (2015d) were applied, with a few enhancements, as described below.

8.9.1 Catchment and Subcatchment Boundaries

The geographical region of South Africa, Lesotho and Swaziland has been divided up by the Department of Water and Sanitation (DWS) into a hierarchical system of catchments, composed of 22 Primary Catchments containing, Secondary, Tertiary and Quaternary Catchments. The Quaternary Catchments are widely used in South Africa for water resources assessments. The upper uMngeni Catchment consists of the U20A (Mpendle), U20B (Lions River), U20C (Midmar), U20D (Karkloof) and U20E (Albert Falls) Quaternary Catchments, as shown in Figure 5-1.

For the purpose of the case study it was decided that the sub-Quaternary catchment boundaries used by Umgeni Water and also (Warburton, 2011) should be used. These sub-Quaternary catchment boundaries did not match the new DWS Quaternary Catchment (SLIM, 2014) boundaries exactly and had to be adjusted slightly or split into two. The sub-Quaternary catchments used for the upper uMngeni Catchment are shown in Figure 8-42.

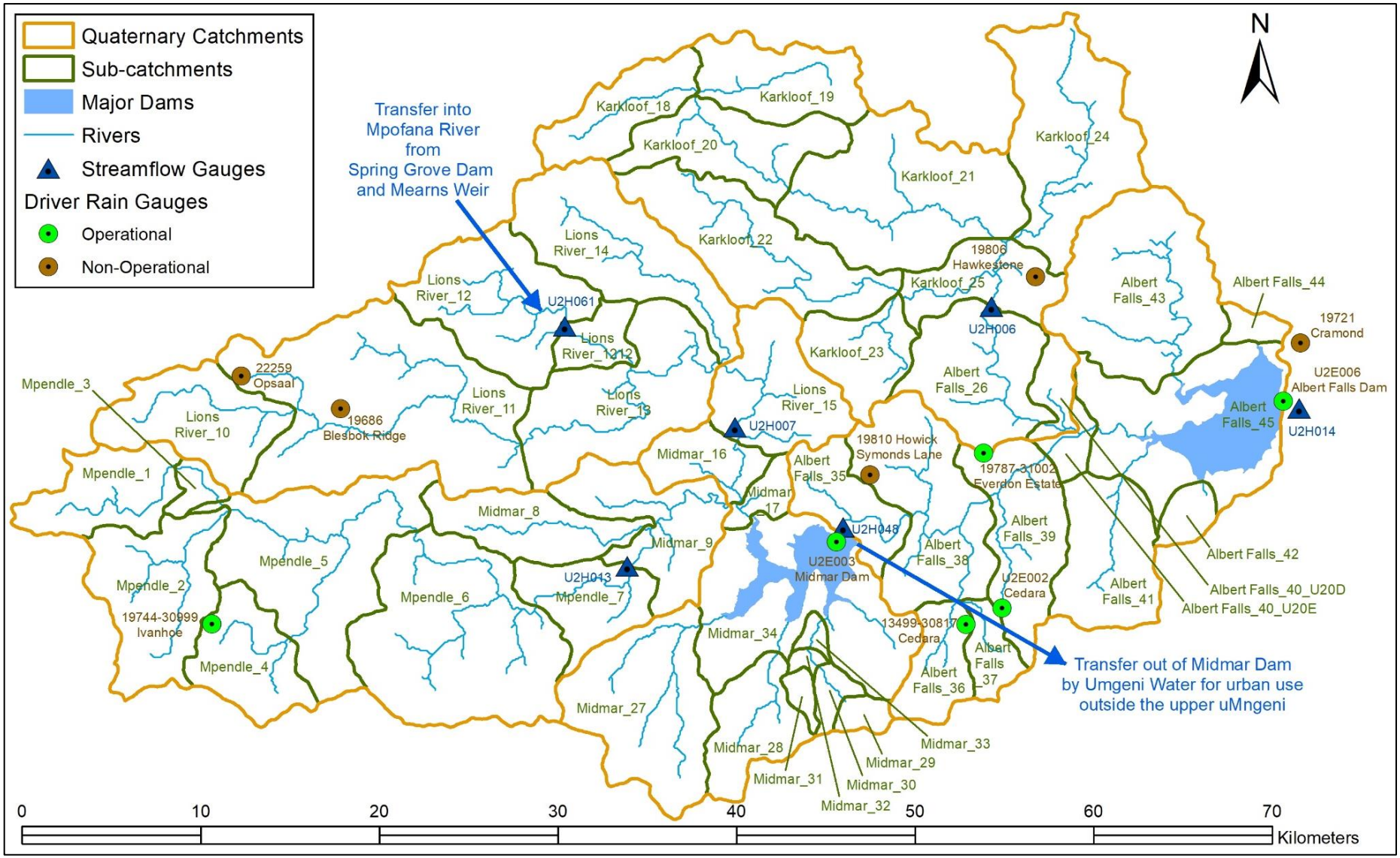

Figure 8-42 Catchments, rivers, major dams, streamflow gauges and rain gauges in the upper uMngeni Catchment

8.9.2 Altitude

The Digital Elevation Model (DEM) (Weepener *et al.*, 2011e) with 90 m resolution described by Weepener *et al.* (2011a) was used to determine the mean altitude for each sub-Quaternary catchment. The DEM altitudes are shown in Figure 8-43. The altitude ranges from 2064 m in the Lions River Catchment in the West to 656 m at Albert Falls Dam in the East.

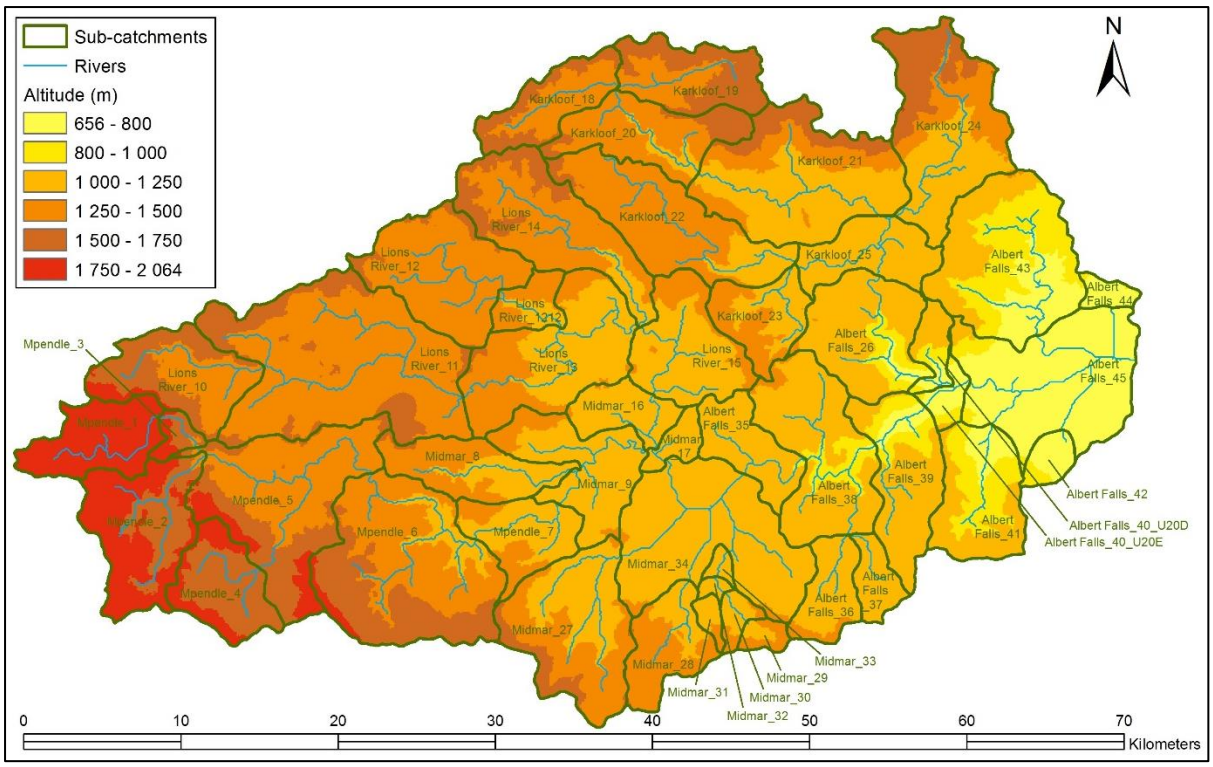

Figure 8-43 DEM altitudes for the upper uMngeni Catchment (after Weepener *et al.*, 2011e)

8.9.3 Rivers and River Nodes

The river features shown in Figure 8-42 were clipped from a vector dataset developed by Weepener *et al.* (2011d) in WRC project K5/1908 using the Shuttle Radar Topography Mission (SRTM) 90m DEM as described by Weepener *et al.* (2011a). As described in Clark (2015d), the clipped rivers dataset together with the upper uMngeni sub-Quaternary Catchment boundaries dataset was used to manually create a point shapefile of river nodes using ArcMap. A river node was created where each sub-Quaternary catchment boundary intersected a river segment and where a confluence of river reaches occurred between

these points. For each node, attributes were set specifying the downstream node and whether the node was at the exit of a sub-Quaternary catchment.

Typically, in *ACRU*, river reaches are modelled as simple conduits of water, thus: (i) they are assumed to have no surface area within the catchment through which they flow, and (ii) hydraulic characteristics such as length, slope, friction and cross-sectional area are not required. However, although the rivers in the upper uMngeni Catchment are relatively small, the land cover/use dataset used (as discussed in Appendix 8.9.7) included some pixels of the class *Water (natural)* which were identified as being open water sections of river reaches for which an area could be calculated. Based on the river features dataset shown in Figure 8-42, a new vector dataset was created containing only the main river reaches within each sub-Quaternary catchment. Using this new dataset the reach lengths were measured and the average reach slopes were calculated using the start and end elevation of each reach, taking into account Howick Falls (95 m) and Karkloof Falls (98 m).

Information describing the hydraulic characteristics for the main river reaches was required for Muskingum flow routing (McCarthy, 1938) in the eWater Source river network model. As there were no known measured reach cross-section data available for the study catchment, the generalised equations proposed by Allen *et al.* (1994) based on measurements at 674 stations in the United states, were used as described by Schulze *et al.* (2005). These equations relate channel width, depth and velocity to flow rate using equations of the form $y = a\,Q^b$ (where: y = dependent variable [m or m/s], Q = flow rate [m^3/s], a = constant, b = exponent). A Muskingum X (weighting factor [dimensionless]) value of 0.2 was assumed (Smithers and Caldecott, 1995) and Muskingum K (storage time constant [s]) values were calculated for a range of flow rates for each main river reach as eWater Source provides the facility to estimate the K values dynamically based on a user specified table of corresponding flow rates and K values.

8.9.4 Streamflow Gauges

There are six operational streamflow gauges operated by DWS in the catchment, and the location of these is shown in Figure 8-42. The measured primary flow data was downloaded from the DWS website [http://www.dwa.gov.za/Hydrology/Verified/hymain.aspx]. The measured primary flow data (instantaneous flow rates, typically at 12 minute intervals) were converted to 24 hour flow volumes for an 8:00 am to 8:00 am day to be compatible with the modelled daily streamflow outputs which were based on 24 rainfall totals for an 8:00 am to 8:00 am rainfall day. The measured daily streamflow data for gauge U2H048 were provided

by Umgeni Water. Gauge U2H013 is on the upper reaches of the uMngeni River in Quaternary Catchment U20A. Gauge U2H061 is a relatively new gauging station operational (since March 2013) on the Mpofana River a short distance downstream of the outfall of the inter-catchment transfer pipeline from Mearns Weir and Spring Grove Dam. The catchment area upstream of gauge U2H061 is relatively small (50.6 km^2) and thus flows are substantially altered by the inter-catchment transfer flows. Further downstream, gauge U2H007 provides a measure of flows in the lower portion of the Lions River. Gauge U2H006, in Quaternary Catchment U20D, provides a measure of flows in the Karkloof River. The other two gauges, U2H048 just downstream of Midmar Dam and U2H014 just downstream of Albert Falls Dam, provide a measure of the combined spillway and controlled release flows from these dams. The measured flow data from these six streamflow gauges were used to verify the streamflows simulated by the *ACRU* model.

8.9.5 Dams

In addition to the two large dams, Midmar and Albert Falls, the upper uMngeni Catchment has a large number of smaller farm dams of various sizes used for irrigation, stock watering and recreational fishing. These dams can have a significant effect on the hydrology within a catchment due to their impact on water flows and due to the evaporation from their open water surfaces. However, the lack of good datasets characterising these dams, and the need to simplify the representation of these dams for modelling purposes, provided challenges for the configuration of the model and for the simulations. The following potential sources of information on dams were identified:

- The DWS Dam Safety Office (DSO) database of registered dams (DSO, 2016) is a database of dams with a storage capacity of more than 50000 m^3 and a wall height of more than 5 m. This database includes information on the latitude and longitude of the dam wall, full surface area and full storage capacity of each dam.
- The DWS Water Authorisation Registration and Management System (WARMS) database (Anderson *et al.*, 2008) includes information on all dams reported by water users as part of the water use registration and licencing process, including latitude-longitude location, full surface area, full storage capacity and purpose.
- The 1:50000 scale topographical maps from the Surveyor General includes information on surface area and location in the form of polygons.
- The Ezemvelo KwaZulu-Natal Wildlife land cover/use dataset for 2011 (Ezemvelo KZN Wildlife and GeoTerraImage, 2013) includes 20 m raster pixels classed as *Water (dams)*.

In the upper uMngeni Catchment there are a total of 103 registered dams. In Clark (2017a) a small investigation for the upper uMngeni Catchment compared the DWS DSO database, the DWS WARMS database and dams included on the 1:50000 topographic maps. The DWS WARMS database, the 1:50000 topographic maps and Google Earth were used to verify and correct, where necessary, the full surface area of the dams in the DWS DSO database. The DWS WARMS database together with the empirical relationship $A = 7.2\ S_v^{0.77}$ (where: A = surface area [m^2], S_v = storage volume [m^3]), developed by Maaren and Moolman (1985), were used to verify and correct, where necessary, the full storage capacity of the dams in the DWS DSO database. However, the lack of a reliable identifier of individual dams that is common to all three datasets made cross-checking difficult and there were substantial differences between the datasets with respect to the individual dams contained in each dataset and their location.

The methodology described in Clark (2015b) and modified by Clark (2016) was used to determine the configuration of dams within each sub-Quaternary Catchment to be used in the *ACRU* model. The land cover/use raster dataset (Ezemvelo KZN Wildlife and GeoTerraImage, 2013), discussed in Appendix 8.9.7, was used to create a vector dataset of dam polygons, which was used in conjunction with the DWS DSO database to identify unregistered dams. For each unregistered dam the estimated full surface area was used with the Maaren and Moolman (1985) equation to estimate the full storage capacity. The unregistered dams within each sub-Quaternary Catchment were then lumped together, by summing surface areas and storage capacities and calculating a composite area:capacity relationship, to create a single lumped unregistered dam per catchment. Midmar Dam and Albert Falls Dam were modelled as individual dams. All other registered dams within each sub-Quaternary Catchment were then lumped together, by summing surface areas and storage capacities and calculating a composite area:capacity relationship, to create a single lumped registered dam per catchment. Thus, in most subcatchments, where necessary, two lumped dams were modelled: (i) a lumped dam representing unregistered dams, and (ii) a lumped dam representing registered dams. The area:capacity relationship of each individual registered dam was estimated by adjusting the exponent of the Maaren and Moolman (1985) equation to fit the surface area and storage capacity when full, which was the only known point on the curve. More accurate area:capacity relationships for Midmar Dam and Albert Falls Dam were derived from information provided by the Durban Regional Office of DWS.

A common simplifying assumption when configuring the *ACRU* model is to assume a single lumped dam at the downstream outlet of each catchment. This can affect simulated streamflows, especially when small dams are almost empty at the start of the rainy season.

In the upper uMngeni Catchment it was observed that this was not a good assumption in most catchments, due the large number and spatial distribution of the dams with most not being on the main river reaches. A Python script, described in Clark (2017b), was developed to use the land cover/use dataset (Ezemvelo KZN Wildlife and GeoTerraImage, 2013), the DEM dataset (Weepener *et al.*, 2011c) and the flow direction dataset (Weepener *et al.*, 2011b) to determine the region within each sub-Quaternary Catchment that is upstream of farm dams. The dams and their contributing regions are shown in Figure 8-44. In each catchment the runoff into dams and outflow from dams were configured as follows:

- HRUs within the region upstream of farm dams contribute runoff to the lumped unregistered dam, if it exists, otherwise to the lumped registered dam, if it exists;
- the lumped unregistered dam, if it exists, then flows into the lumped registered dam, if it exists;
- the lumped registered dam, if it exists, then flows to an individual large registered dam, if it exists, otherwise to the outlet of the catchment;
- HRUs within the region downstream of farm dams contribute runoff to an individual large registered dam, if it exists, otherwise to the outlet of the catchment.

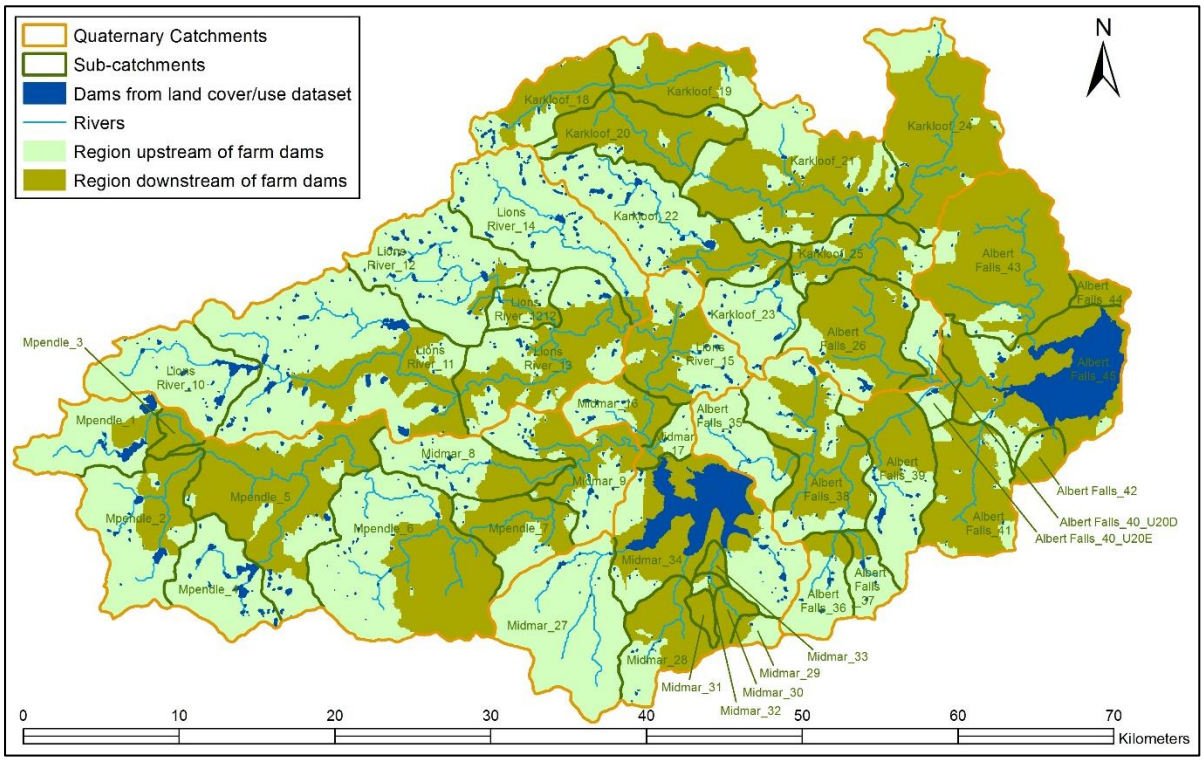

Figure 8-44 Dams and regions upstream or downstream of farm dams in the upper uMngeni Catchment

For Midmar Dam and Albert Falls Dam it was necessary to estimate the daily flow release volumes. Flow releases occur from Albert Falls Dam to make water available to Nagle Dam downstream from which Umgeni Water pump water for use to supply eThekwini Municipality with bulk water for use in parts of the greater Durban area. These flow releases from Albert Falls Dam were estimated by subtracting the daily dam spillway flow volumes, measured at DWS gauge U2R003, from daily flow volumes at the downstream weir U2H014. The same method did not work for Midmar Dam, possibly due to a problem with the measurement of flows at the downstream weir U2H048. The flow releases from Midmar Dam were estimated from daily flow volumes at the downstream weir U2H048, but during dam spill periods the flow releases were estimated by interpolating between daily flow volumes at U2H048 just before and just after the spill period.

The *ACRU* model accounts for seepage from dams. Seepage from Midmar Dam and Albert Falls Dam was assumed to be zero as, if there is any seepage, it would be accounted for in the estimation of flow releases described above. For the farm dams, a seepage rate of 0.067 % of dam full storage capacity per day was assumed, based on the recommendation in Smithers and Schulze (1995). The *ACRU* 3 version assumes that this seepage rate is constant. However, a small improvement was made to the *ACRU* 5 version by using the area:capacity relationship for a dam to estimate the depth based on the current volume of water stored in the dam, and the seepage rate was then varied in proportion to the depth of water relative to the depth at full capacity, i.e. seepage reduces as the depth of water in the dam decreases.

The flow routing option in the *ACRU* 3 and *ACRU* 2000 versions of the model substantially increases the execution time of the model and has not yet been included in the *ACRU* 5 version. When running *ACRU* without the flow routing option turned on, all water in a dam exceeding the full storage volume at the end of the daily timestep is transferred downstream via the spillway, thus the hydraulic characteristics of the spillway are not required. The estimation of the spillway hydraulic characteristics for dams was required for flow routing in eWater Source. The spillway rating curves for Midmar Dam and Albert Falls Dam were obtained from DWS. However, no spillway rating curves or even spillway dimensions were available for the farm dams, including the registered dams in the DWS DSO database. The spillway dimensions for these farm dams were estimated using recommendations from the Food and Agriculture Organisation's (FAO) "Manual on small earth dams" (Stephens, 2010) together with design flood peak flow rates (Smithers, 2014) for the catchment areas upstream of these dams.

The storage volumes of the farm dams were initialised to 50 % of full storage capacity at the start of the simulation and *ACRU* was run for a warm-up period of one full hydrological year prior to the start date of the required simulation period. The storage volumes of Midmar Dam and Albert Falls Dam were initialised using measured values recorded by DWS for the start of the warm-up period (1 October 2007).

8.9.6 Transfers, Abstractions and Return Flows

The available water supply in the uMngeni Catchment has been augmented by transfers from Mearns Weir since 1983 and from Spring Grove Dam since 2016. The transfer from Mearns Weir started as an emergency scheme during a severe drought but was recommissioned for regular use in 1993 (DWAF and Umgeni Water, 2004). Mearns Weir, situated on the Mooi River just downstream of the Little Mooi tributary, does not provide much storage, thus the scheme, which has a pumped of capacity of 3.2 m^3/s, is operated on a run-of-river basis (DWAF and Umgeni Water, 2004). Spring Grove Dam, with a storage capacity of 139.5 million m^3 was completed in 2013 as part of Phase 2 of the Mooi-Mgeni River Transfer Scheme (DSO, 2016). The transfer from Spring Grove Dam has a pumped capacity of 4.5 m^3/s, which is the capacity of the transfer infrastructure and the receiving stream (DWAF and Umgeni Water, 2004). Measured flow values from Mearns Weir and Spring Grove Dam were obtained from Umgeni Water and were used to create a daily time series of flow volumes into the Lions River_12 sub-Quaternary catchment for use in the *ACRU* model.

Umgeni Water extracts water from Midmar Dam to supply bulk water to municipalities, both within and outside the upper uMngeni Catchment. The measured daily primary flow rate data for gauging station U2H049 was downloaded from the DWS website [http://www.dwa.gov.za/Hydrology/Verified/hymain.aspx]. Water extracted from Midmar Dam is treated at the Midmar Water Treatment Works (WTW) and DV Harris WTW. Based on information included in Umgeni Water (2016) it was established that for the years 2012, 2013, 2014 and 2015 the water supplied from these two WTW to Howick, Mpophomeni and Albert Falls (within the upper Umgeni Catchment) was almost constant with an average of 3.8 % of the total water abstraction from Midmar Dam. Thus, the daily flow volumes at U2H049 were multiplied by 0.962 to create an estimate a daily time series of flow volumes from Midmar Dam out of the study catchment for use in the *ACRU* model. The data for U2H049 showed that for the hydrological years 2007/2008 to 2014/2015 the average annual abstraction volume from Midmar increased by about 5.4% but due to drought induced restrictions put in place by Umgeni Water abstractions reduced from 130 Million m^3/annum in

2014/2015 to 120 million m^3/annum in 2015/2016. Thus, it is important to be able to represent these variations over time in the *ACRU* model inputs.

8.9.7 Land Cover/Use

Land cover/use datasets are compiled for different purposes by different people and organisations, thus the classification system used varies. This led to the development of a standard classification of land cover/use for the water use quantification and accounting methodology developed by Clark (2015b). This standard classification makes it easier to compare results from studies based on different land cover/use datasets, for different time periods or for different catchments. For each land cover/use dataset it is necessary to map each of the dataset classes to one of the standard classes. To represent the different land cover/use classes, for the purpose of configuring the *ACRU* model, a database of land cover/use class information was developed. For each class, relevant information is stored, such as: vegetation characteristics (canopy interception, crop coefficient, sensitivity to stress, root distribution), coefficient of initial abstraction, dryland or irrigated, annual or perennial, pervious and impervious area fractions.

In addition, and as described in more detail by Clark (2015b), the standard land cover/use classes were organised into a hierarchy of categories of land cover/use. This hierarchical structure provides a means of grouping similar land covers and uses so that they can be summarised with different degrees of detail in water accounts. The most specific categories are at the bottom of the hierarchy, within increasingly more general categories, ending with the most general categories at the top of the hierarchy. The following five categories form the top of the hierarchy:

- *Natural* - areas covered with natural vegetation or uncultivated bare ground,
- Cultivated - areas covered with agricultural crops or production forest plantations,
- Urban/Built-up - urban and other built-up areas including residential, commercial and industrial areas,
- Mines and Quarries - areas characterised by quarries, subsurface and surface mining features, and
- Waterbodies - open bodies of water and wetland areas with aquatic vegetation cover.

As described in Clark (2015b), the determination of HRUs for modelling in *ACRU* were primarily based on land cover and use. For this case study the Ezemvelo KwaZulu-Natal Wildlife raster land cover/use dataset for the province of KwaZulu-Natal for the year 2011

(Ezemvelo KZN Wildlife and GeoTerraImage, 2013) with resolution of 20 m was used. As shown in Figure 8-45, in the upper uMngeni Catchment, the remaining natural land cover is predominantly grassland with some bush and dense bush in the East, and some patches of indigenous forest especially in the Karkloof area in the North-East. There are extensive cultivated areas including production forest plantations, commercial agriculture (both dryland and irrigated), and some sugarcane in the warmer area in the East. The catchment includes the urban areas of Howick, Mpophomeni and part of Hilton. In addition to the dams discussed in Appendix 8.9.5, there are several wetlands distributed through the western and central parts of the catchment.

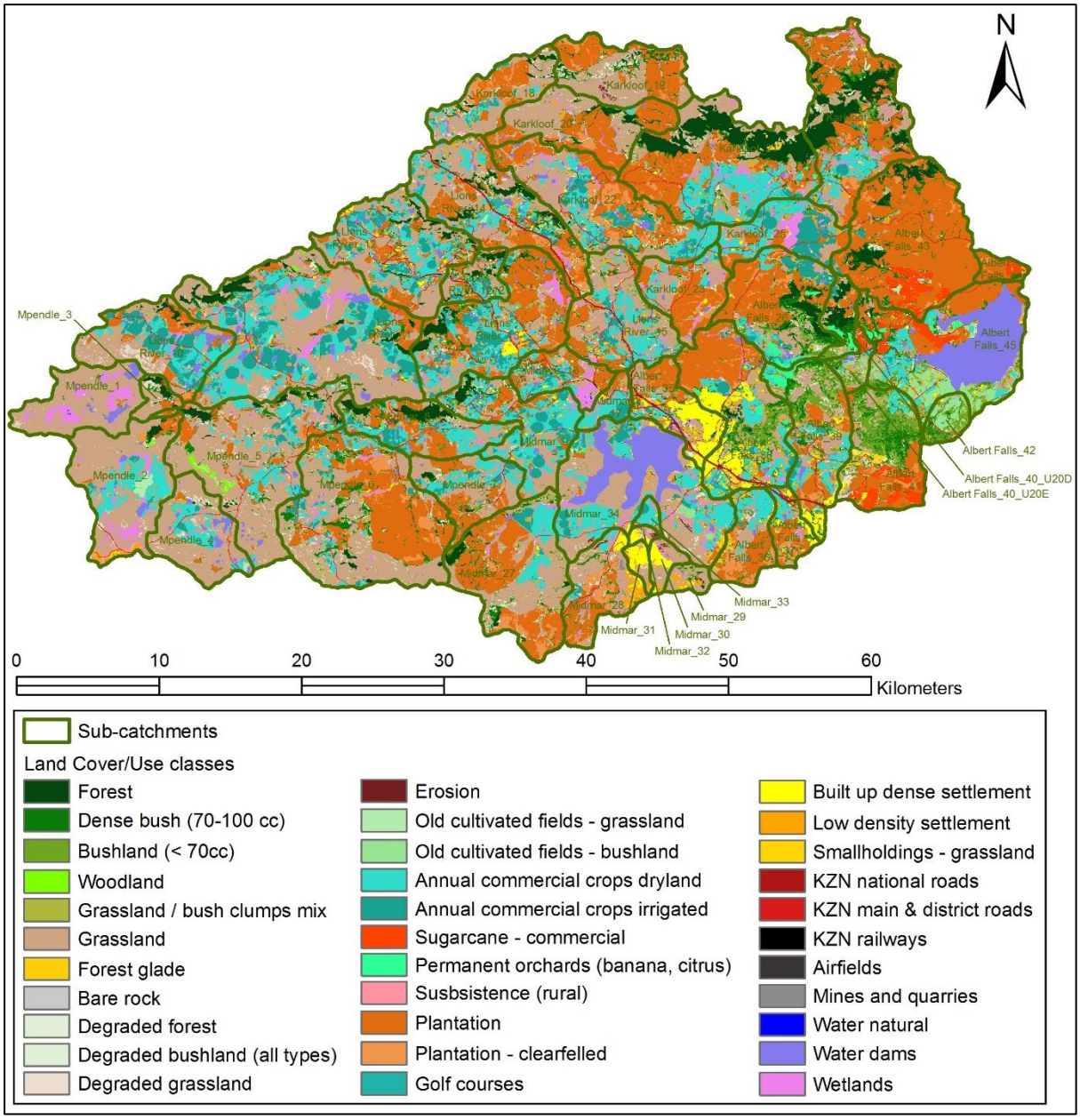

Figure 8-45 Land cover/use classes in the upper uMngeni Catchment (after Ezemvelo KZN Wildlife and GeoTerraImage, 2013)

In most of the land cover/use datasets, natural vegetation is classified as either natural vegetation or degraded natural vegetation with a few very general classes for each. In order to be able to represent natural vegetation in more detail, Clark (2015b) describes the use of the Acocks Veld Types (Acocks, 1988) dataset together with the hydrological characteristics derived by Schulze *et al.* (2004). If the natural vegetation is degraded then the hydrological characteristics are adjusted based on recommendations by Schulze (2013). The Acocks Veld Types in the upper uMngeni Catchment are shown in Figure 8-46.

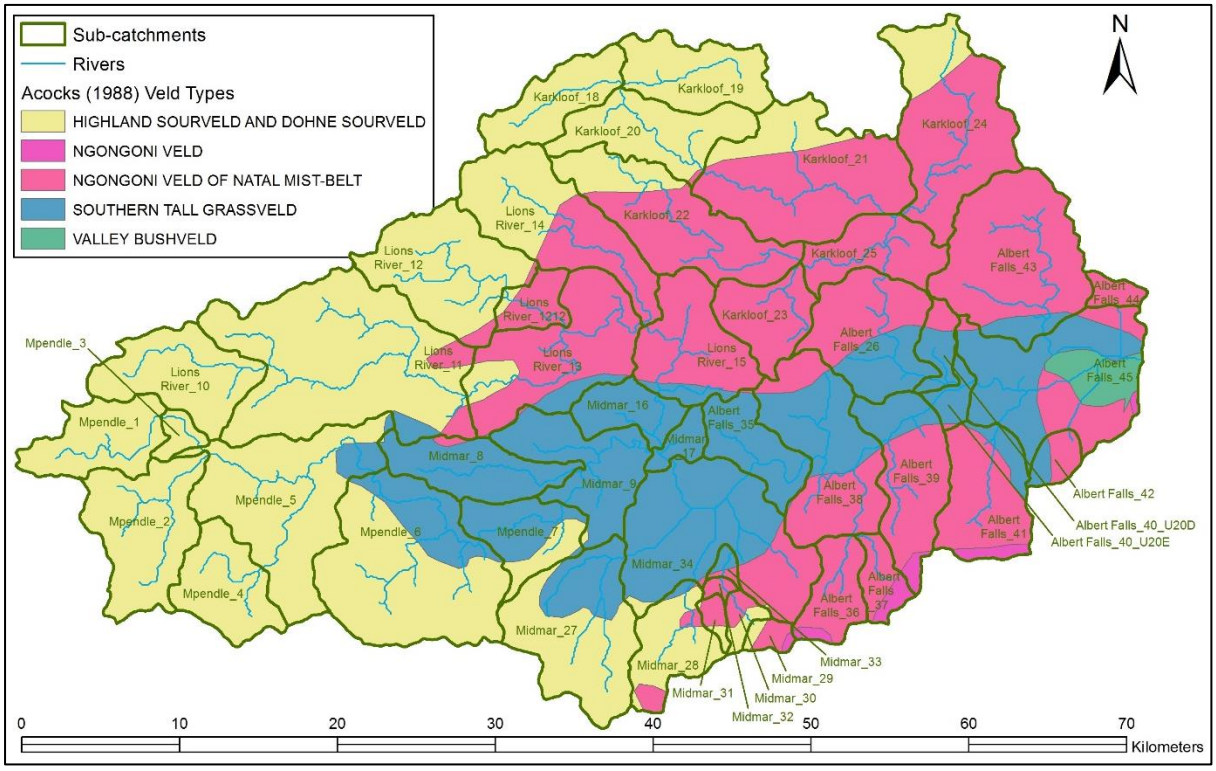

Figure 8-46 Acocks Veld Types in the upper uMngeni Catchment (after Acocks, 1988)

Water requirements for irrigation are estimated in *ACRU* which was configured to use a soil water deficit scheduling method. *ACRU* was configured such that in each sub-Quaternary Catchment the lumped registered dam, if one exists, was assumed to be the water source for irrigation, otherwise irrigation was assumed to be from run-of-river on the main river reach within the catchment.

Urban areas were represented in *ACRU*, as described in Clark (2015b), using a combination of pervious vegetated areas (irrigated in some instances), disjunct impervious areas (representing roofs) and adjunct impervious areas (representing roads and other infrastructure connected directly to some form of storm drainage). The proportion of these different areas varied for the different classes of urban area. Residential water

requirements, as described in Clark (2015b), were determined using: (i) population estimates from the CSIR's functional typology population dataset (CSIR, 2013), based on the 2011 population census, and (ii) estimated daily water requirements for the different classes of urban area from CSIR (2003). Midmar Dam was assumed to be the source of water for all residential water requirements for the main urban areas. For the low density rural urban areas the lumped registered dam in the local sub-Quaternary Catchment, if one exists, was assumed to be the water source, otherwise from run-of-river on the main river reach within the catchment. Industrial water use was not included as no specific data was available and industrial water use within the upper uMngeni was not considered to be significant.

8.9.8 Soils

The soils dataset developed and described by Schulze and Horan (2008) was used in this case study. The soil hydrological properties included in the dataset are the depth, porosity, drained upper limit and wilting point for each of the A and B soil horizons, and also the saturated drainage rate from the A to the B soil horizon. Using the methodology described in Clark (2015b) the dominant soil type for each land cover/use based HRU within each sub-Quaternary catchment was used to determine the hydrological characteristics required by the *ACRU* hydrological model for each HRU. The soil moisture stores were initialised to 50% of plant available water (PAW) and then the *ACRU* model was run for a warm-up period of one full hydrological year prior to the start date of the required simulation period.

8.9.9 Climate

The *ACRU* model requires daily rainfall time series for each sub-Quaternary Catchment as an input. *ACRU* also requires daily reference potential evaporation time series, or other climate variables from which it can be calculated, for each sub-Quaternary Catchment as an input. Other climate variables required by *ACRU* are: (i) MAP, for the estimation of catchment lag using the Schmidt/Schulze equation (Schmidt and Schulze, 1987), which is used in calculating peak discharge, and (ii) air temperature, which is used to adjust crop coefficients following periods of water stress.

8.9.9.1 Mean annual precipitation (MAP)

The MAP dataset developed by Lynch (2004) was used in this case study. The MAP for the upper uMngeni Catchment is shown in Figure 8-47. The upper uMngeni Catchment has a relatively high rainfall, but there is significant spatial variation across the catchment, with the

Karkloof Catchment and the area just to the south of it having the highest rainfall. The Lynch (2004) dataset was used to create a shapefile containing area weighted mean MAP values for each sub-Quaternary catchment which were then used to configure the *ACRU* model.

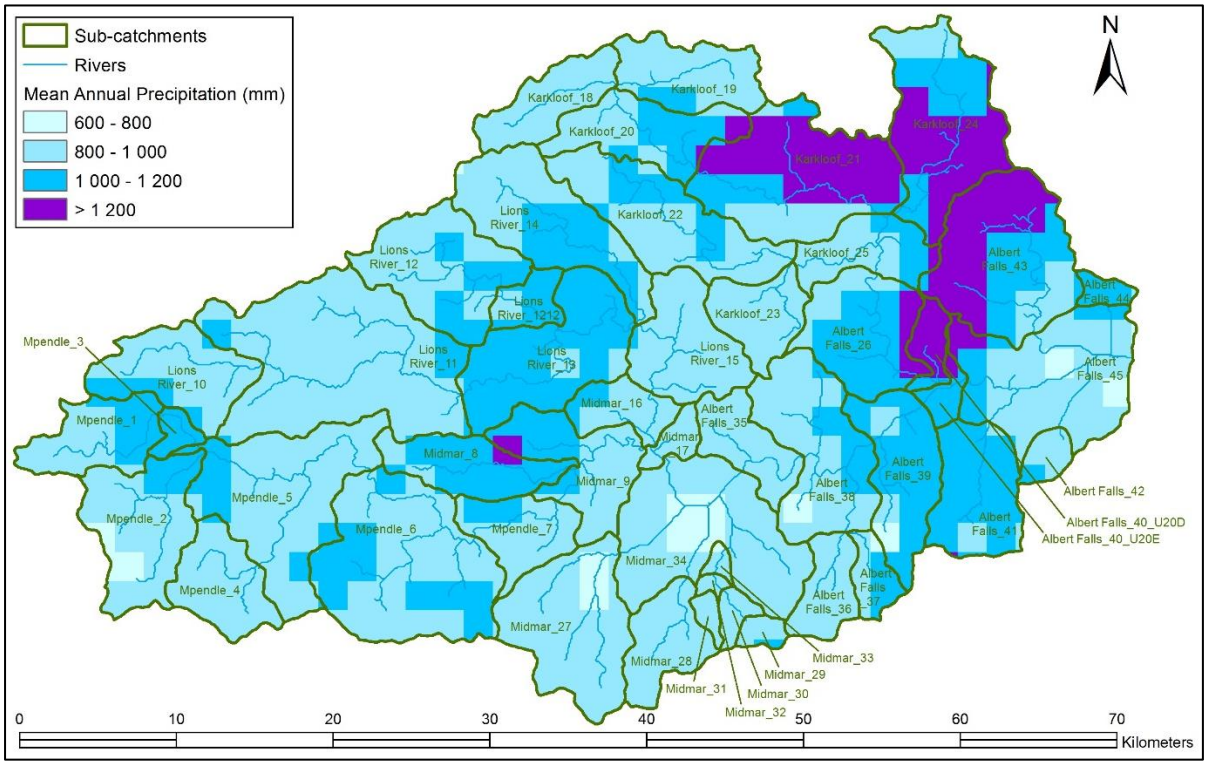

Figure 8-47 MAP in the upper uMngeni Catchment (after Lynch, 2004)

8.9.9.2 Daily rainfall

Accurate estimation of areal rainfall is important as it is the one of the key inputs required for hydrological modelling. Rain gauge networks are crucial, yet expensive to establish and maintain, and there has been a general decline in the number of rain gauges in South Africa (Pegram *et al.*, 2016). This decline is also evident in the upper uMngeni Catchment, as shown in Figure 8-42, which is a catchment that is part of an economically important region of KwaZulu-Natal and which provides water to millions of people. In addition to poor spatial representation, other potential problems associated with using rain gauge data for modelling include: (i) missing and poor quality data, (ii) poor accessibility to data from the institutions that measured it, and (iii) the lag between data measurement and when it is made accessible. Remotely sensed rainfall datasets from satellites offer some potential advantages over using rain gauge data for modelling, including: (i) better spatial representation, despite the coarse resolution, in areas with a sparse rain gauge network, (ii) some datasets are available in near-real time, (iii) many datasets are freely available and can be downloaded over the internet. Although investigating remotely sensed satellite

rainfall was not one of the objectives of this study, it formed an important part in attempting to improve the performance of hydrological modelling in the case study catchment by improving the catchment rainfall estimates using satellite remotely sensed rainfall estimates asa relatively new source of rainfall data.

Clark (2015b) reported on an initial investigation into four remotely sensed rainfall datasets: (i) CMORPH (Joyce *et al.*, 2004), (ii) FEWS RFE 2.0 (Novella and Thiaw, 2012), (iii) FEWS ARC 2.0 (Novella and Thiaw, 2012), and (iv) TRMM 3B42 Kummerow *et al.* (2000). These remotely sensed datasets compared favourably with rain gauge data in the uMngeni Catchment but performed poorly in the Sabie-Sand Catchment. Conversely, when these datasets were used to model streamflow using *ACRU* and the results were compared with measured streamflow, the results in the Sabie-Sand were more encouraging than the results for the uMngeni catchment. The requirement for some form of adjustment for localised bias was evident. Clark (2015b) concluded that although remotely sensed rainfall datasets offer advantages in spatial representation and availability, the coarse resolution and bias in rainfall quantities are a problem in accurately estimating rainfall at sub-Quaternary Catchment scale and that further investigation was required into methods for downscaling and adjusting to reduced localised biases.

Further investigation into some simple methods for adjusting remotely sensed rainfall estimates to reduce local biases are reported in Clark (2016). Part of the investigation by Clark (2016) was repeated by Clark (2018) using an additional year of data and the refined *ACRU* configuration described in this section. A summary of the previous investigations by Clark (2015b) and Clark (2016), descriptions of the datasets used and the results of the extended investigation are included in Clark (2018). The outcome of the investigation was that an adjustment method based on the accumulative frequency distributions of the remotely sensed and rain gauge datasets, was found to be effective in reducing the differences between the means and the standard deviations of the measured and estimated streamflow datasets, but was not effective in improving the goodness-of-fit indicated by the R^2 and NSE values, possibly as a result of the timing of modelled streamflows. The adjusted FEWS ARC 2.0 and FEWS RFE 2.0 rainfall datasets resulted in better verifications against modelled streamflow than the TRMM 3B42 and CMORPH datasets. Thus, based on the extended investigation, reported in Clark (2018), a rainfall dataset based on the FEWS RFE 2.0 satellite remotely sensed daily rainfall product (Novella and Thiaw, 2012), but adjusted using measured rainfall data from rain gauges to reduce localised bias, was selected for use in this case study. The area weighted monthly rainfall depths for the upper uMngeni Catchment are shown in Figure 8-48.

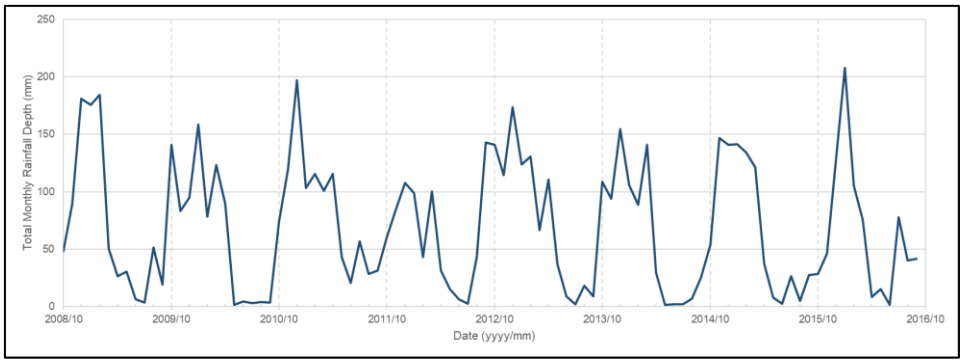

Figure 8-48 Area weighted monthly rainfall depths for the upper uMngeni Catchment

8.9.9.3 Reference potential evaporation

A reference potential evaporation (ET_0) dataset was developed using the Penman-Monteith equations (Allen *et al.*, 1998), together with forecast climate data from the South African Weather Service (SAWS) version of the Unified Model and remotely sensed radiation data (Pegram *et al.*, 2010; Sinclair and Pegram, 2010; Sinclair and Pegram, 2013). The ET_0 dataset of hourly values at 0.11° spatial resolution is available on the Satellite Applications Hydrology Group (SAHG) website (http://sahg.ukzn.ac.za/soil_moisture/et) for the period October 2007 to February 2017.

The SAHG ET_0 dataset was used together with a shapefile dataset of sub-Quaternary catchment boundaries to create a time series of area weighted daily ET_0 data values for each sub-Quaternary catchment. The area weighted monthly ET_0 depths for the upper uMngeni Catchment are shown in Figure 8-49. For the eight years shown there appears to be a trend of increasing maximum monthly total ET_0 values each year. The *ACRU* model was originally developed to use A-pan reference potential evaporation together with associated crop factors. Therefore, an adjustment factor of 1.2 (Shuttleworth, 2010) was applied to the ET_0 values in *ACRU* to estimate A-pan equivalent daily ET_0 values.

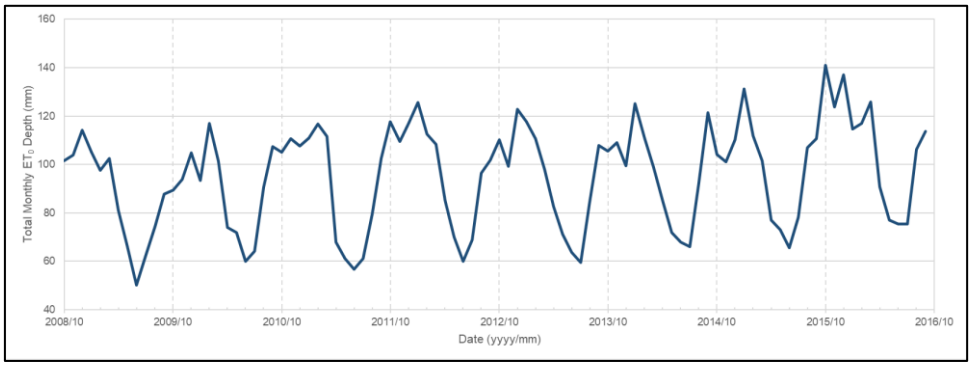

Figure 8-49 Area weighted monthly ET_0 depths for the upper uMngeni Catchment

8.9.9.4 Air Temperature

The datasets of long-term mean month-of-year maximum and minimum daily air temperature developed by Schulze and Maharaj (2008a) and Schulze and Maharaj (2008b) were used in this case study. For each of the 12 month-of-year maximum air temperature raster datasets and the 12 month-of-year minimum air temperature raster datasets an areal mean value was calculated for each sub-Quaternary catchment to produce a shapefile dataset of month-of-year maximum and minimum daily temperature values. The values were then used to configure the *ACRU* model.

8.10 Case Study - Verification of the Simulations

The *ACRU* 5 version of the model was configured for the upper uMngeni Catchment and the model was run for the period 1 October 2007 to 30 September 2016, a total of nine hydrological years. The simulated streamflow results were compared to measured streamflow at the six streamflow gauges described in Appendix 8.9.4. The first year of simulated streamflow was regarded as a warmup period for the model and excluded from the statistical analysis, resulting in an analysis of eight hydrological years. Any periods where there was missing data in each measured streamflow dataset, were also excluded from the statistical analysis. At streamflow gauge U2H061 only three hydrological years, starting in October 2013, were included in the comparison as measurements only started in 2013.

The goodness-of-fit statistics comparing the daily measured and simulated streamflow time series are shown in

Table 8-5, and for monthly time series in Table 8-6. The verification of streamflow at gauges U2H006, U2H007 and U2H013 was of primary interest, as: (i) gauges U2H014 and U2H048 are immediately downstream of large dams and thus have additional uncertainties associated with them, such as errors in upstream flow estimates, abstractions and releases, and (ii) U2H061 is just downstream of the inter-catchment transfer from Mearns and Spring Grove Dam, with the transfer flows being substantially greater than runoff from the catchment.

Table 8-5 Statistics describing daily measured and simulated streamflow depths

Streamflow Gauge	U2H061	U2H007	U2H013	U2H048	U2H006	U2H014
Total measured flows (mm)	3266.113	1512.445	1583.883	615.484	1482.716	712.368
Total simulated flows (mm)	3223.737	1505.795	1687.224	346.374	1510.905	686.417
Mean measured flows (mm/day)	3.125	0.521	0.544	0.211	0.550	0.301
Mean simulated flows (mm/day)	3.085	0.519	0.579	0.119	0.560	0.290
% Difference between means	-1.297	-0.440	6.525	-43.723	1.901	-3.643
Std. Deviation of measured flows (mm)	2.936	0.516	0.903	0.603	0.840	0.390
Std. Deviation of simulated flows (mm)	3.190	0.605	0.765	0.287	0.949	0.382
% Difference between Std. Deviations	8.658	17.155	-15.294	-52.385	12.883	-2.101
Regression Coefficient (slope)	1.057	0.603	0.436	0.138	0.577	0.419
Regression Intercept	-0.218	0.204	0.342	0.089	0.243	0.164
Correlation Coefficient: Pearson's R	0.973	0.515	0.515	0.290	0.512	0.428
Coefficient of Determination: R^2	0.946	0.265	0.265	0.084	0.262	0.183
Nash-Sutcliffe Efficiency	0.936	-0.158	0.154	0.026	-0.081	0.018

Table 8-6 Statistics describing monthly measured and simulated streamflow depths

Streamflow Gauge	U2H061	U2H007	U2H013	U2H048	U2H006	U2H014
Total measured flows (mm)	3033.569	1487.356	1581.441	612.941	1405.551	687.206
Total simulated flows (mm)	2983.849	1462.367	1681.683	343.831	1392.890	661.280
Mean measured flows (mm/month)	94.799	15.823	16.647	6.452	16.934	9.287
Mean simulated flows (mm/month)	93.245	15.557	17.702	3.619	16.782	8.936
% Difference between means	-1.639	-1.680	6.339	-43.905	-0.901	-3.773
Std. Deviation of measured flows (mm)	79.968	13.364	20.574	14.173	19.953	7.933
Std. Deviation of simulated flows (mm)	85.480	13.329	14.601	4.473	16.057	6.233
% Difference between Std. Deviations	6.893	-0.257	-29.030	-68.438	-19.524	-21.433
Regression Coefficient (slope)	1.049	0.654	0.501	0.190	0.660	0.595
Regression Intercept	-6.219	5.207	9.362	2.396	5.613	3.409
Correlation Coefficient: Pearson's R	0.982	0.656	0.706	0.601	0.820	0.758
Coefficient of Determination: R^2	0.963	0.430	0.498	0.361	0.672	0.574
Nash-Sutcliffe Efficiency	0.962	0.333	0.499	0.241	0.705	0.695

The total and mean flow depths were simulated well when considered over the full comparison period, with the exception of gauge U2H048 which is located just downstream of Midmar Dam. The percentage difference between the means of measured and simulated daily streamflows was good (less than 10%) at most gauges. The reason for the poor simulation at gauge U2H048 is not immediately apparent, as estimates at the upstream gauges (U2H007 and U2H0013) are good. The estimation of runoff in the subcatchments immediately surrounding Midmar Dam are also expected to be good as the same driver rain gauge was used to adjust the remotely sensed rainfall in most of Quaternary Catchments

U20A, U20B and U20C. The flows at gauge U2H014 below Albert Falls Dam do not appear to have been substantially impacted by the poor estimation upstream at gauge U2H048. The percentage difference between the standard deviation of measured and simulated daily streamflows was also satisfactory (less than 15%) at most gauges. However, these conservation statistics only tell part of the story. As the flow releases from both Midmar and Albert Falls Dams were estimated based on measured flows at the measurement weirs immediately downstream, the errors in the simulated flows were all due to excessive spill flows or to the dams not spilling when they should have.

The regression statistics for the daily streamflow data, shown in

Table 8-5, indicate that the pattern and magnitude of the daily flows was not simulated well at most of the gauges. Only at gauge U2H061 was there a high degree of association between the measured and simulated daily flows. Gauge U2H061 has a relatively small catchment area (50 km^2) and flows are often dominated by the measured inter-catchment transfer from Mearns Weir and Spring Grove Dam. The regression statistics for the monthly streamflow data, shown in Table 8-6, indicate a better association between the measured and simulated monthly flows, though at most gauges the association was still not good.

To provide further insight into the simulation results the time series of monthly rainfall, measured streamflow and simulated streamflow were plotted for each of the six streamflow gauges as shown in Figure 8-50 to Figure 8-53 and Figure 8-55 to Figure 8-56. The monthly rainfall depths plotted for each streamflow gauge were calculated by area weighting the monthly rainfall depths for all the catchments contributing to the streamflow gauge. In addition the mean monthly storage, as a percent of full capacity, is shown for Midmar Dam in Figure 8-54 and for Albert Falls Dam in Figure 8-57.

The monthly time series of streamflow for gauge U2H061 are shown in Figure 8-50, in which the monthly time series of inflows from the inter-catchment transfer are also shown. It can be seen that the transfer scheme is used every year to supplement the water supply in the uMngeni supply system. Due to the drought the transfer was in operation continuously from September 2015, and the transfer flows increased when the pipeline directly from Spring Grove Dam came into operation in mid-2016. The simulated streamflows closely follow the transfer flows, with runoff from the Lions River_12 catchment having only a small effect on these flows. One area of concern is noted in the two periods in 2014 and 2015 where the measured streamflow is less than the transfer flow. This indicates a possible error in the measurement of one or both of these flows, or that part of the transfer flow is being lost or utilised somewhere between the source and the outfall in the Lions River_12 catchment. However, the measured streamflow is greater than the transfer flow during the 2013/2014 and 2016 transfer periods.

The monthly time series of streamflow for gauge U2H007 on the Lions River downstream of gauge U2H061 are shown in Figure 8-51. The seasonal trends in the measured streamflow are represented well, but there are significant over or underestimations in the magnitude of the flow depths in some seasons. The statistics did not indicate the transfer flows as having a strong influence on the flows at weir U2H007 downstream. However, from 2013 when recording of flows started at gauge U2H061, there appears to be a relationship between the over and underestimated flow periods at these two gauges. This may indicate that the

measured transfer flows, which were used to model the transfer, were incorrect, and at least partly contributed to the poor association between measured and simulated flows at gauge U2H007.

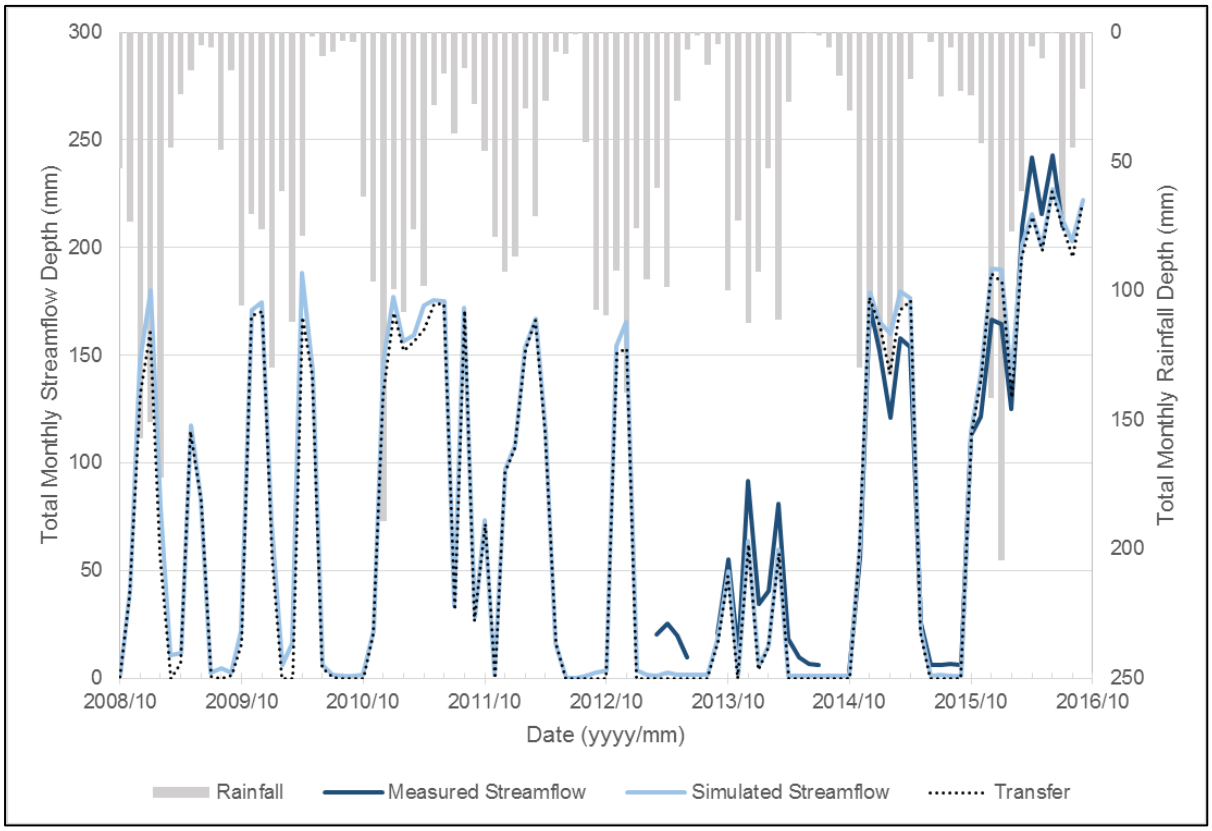

Figure 8-50 Total monthly rainfall and streamflow depths at gauge U2H061

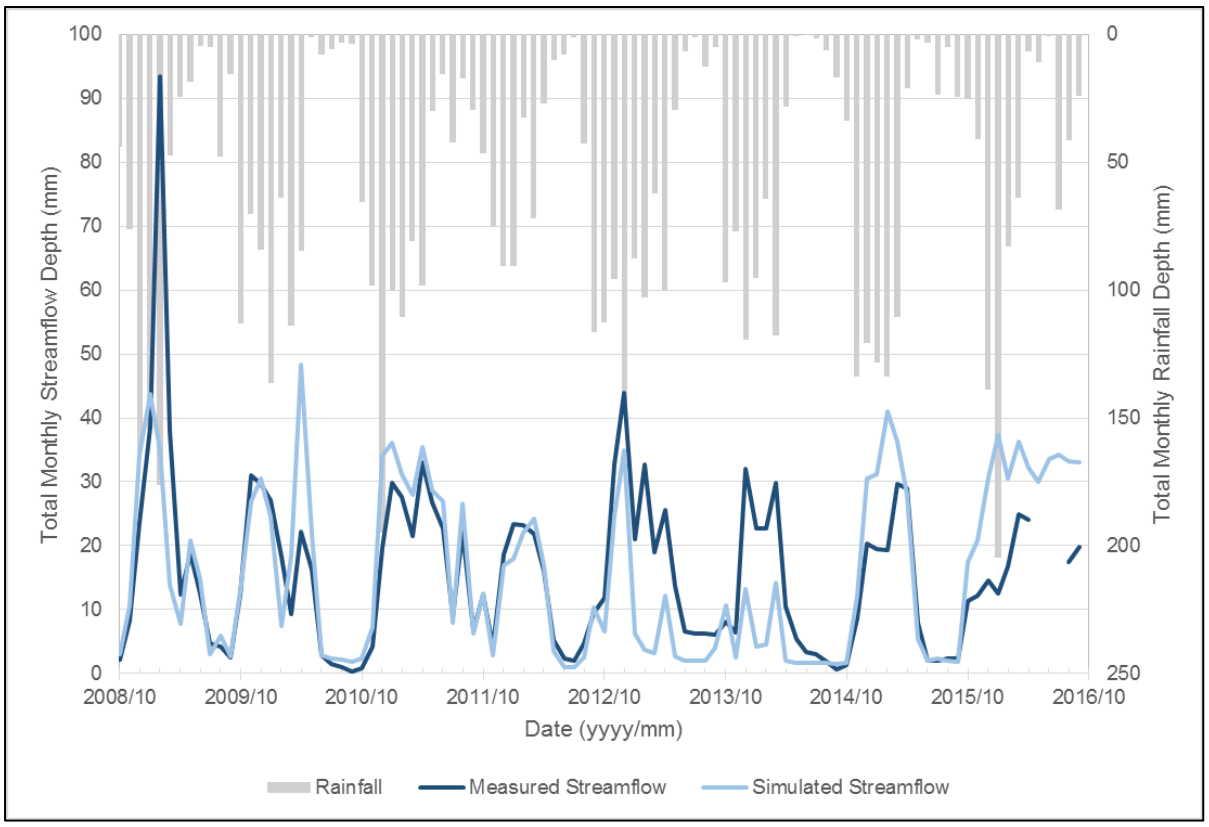

Figure 8-51 Total monthly rainfall and streamflow depths at gauge U2H007 (Lions River)

The monthly time series of streamflow for gauge U2H013 on the uMngeni River in the Mpendle WMA are shown in Figure 8-52. Again, the seasonal trends in the measured streamflow are represented well, but with significant over or underestimations in the magnitude of the flow in some seasons. The trends in the simulated flows appear to be associated with the trends in the estimated rainfall.

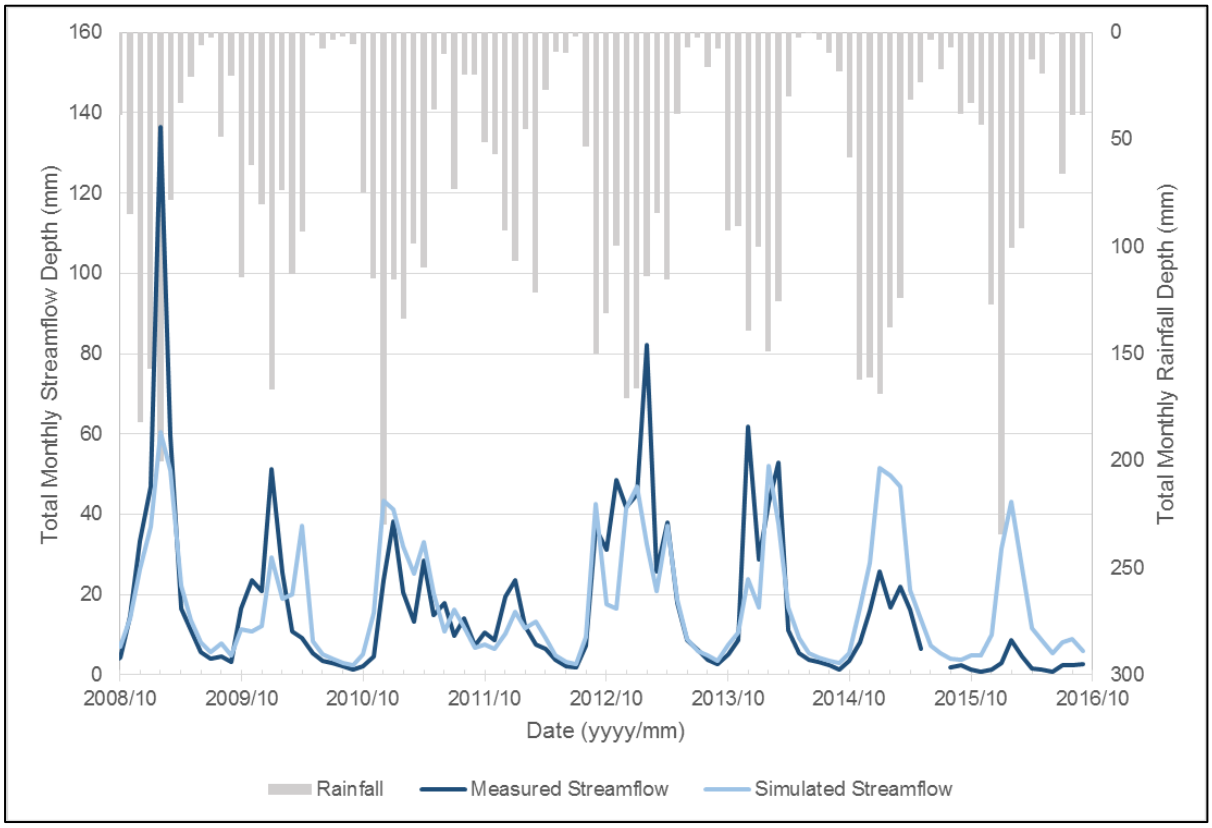

Figure 8-52 Total monthly rainfall and streamflow depths at gauge U2H013 (Mpendle)

At all of gauges U2H007, U2H013 and U2H006 there is an overestimation of streamflow during the drought years 2014/2015 and 2015/2016. Except for simulated streamflow, there is no real way to validate the spatial rainfall estimates. However, a quick comparison of the adjusted FEWS RFE 2.0 rainfall value at the driver rain gauges used to adjust the remotely sensed data showed: (i) an overestimation of rainfall in these two years at the 19744-30999_IvanhoeImpendhle rain gauge, (ii) no clear overestimation at the U2E003_MidmarDam rain gauge, and (iii) no comparison was possible at the 19806_HawkestoneHowick rain gauge as the record stopped in 2011. The high ET_0 values in those two years, as shown in Figure 8-49, seem to indicate that there was not a general underestimation of ET_0 during this drought period. It was concluded that the over simulation of streamflow in these two years was most likely due to the spatial rainfall estimates, though this would need to be investigated further outside of this study.

The monthly time series of streamflow for gauge U2H048 on the uMngeni River just downstream of Midmar Dam are shown in Figure 8-53. As anticipated from the goodness-of-fit statistics the flows at this gauge are poorly simulated. The over and under simulation of flows in the different years corresponds closely to over and under simulation at the two upstream gauges U2H007 and U2H013. Further verification, indicating that it is not just an error with the gauge, is provided in the comparison of measured and simulated monthly average storage values for Midmar Dam, shown in Figure 8-54. In the 2010/2011 hydrological year the simulated spill from Midmar was earlier and bigger than the measured spill. Although the 2011/2012 year was simulated well at the upstream gauges, with a small under simulation at U2H013 early in the year, the dam continued to be drawn down during the summer rainfall season instead of recovering to a small spill as shown in the measured data. Under simulations in the following two years resulted in the simulated storage being drawn down further instead of spilling as it should have. This was followed by two years of over simulated upstream flows, during which the storage over recovered.

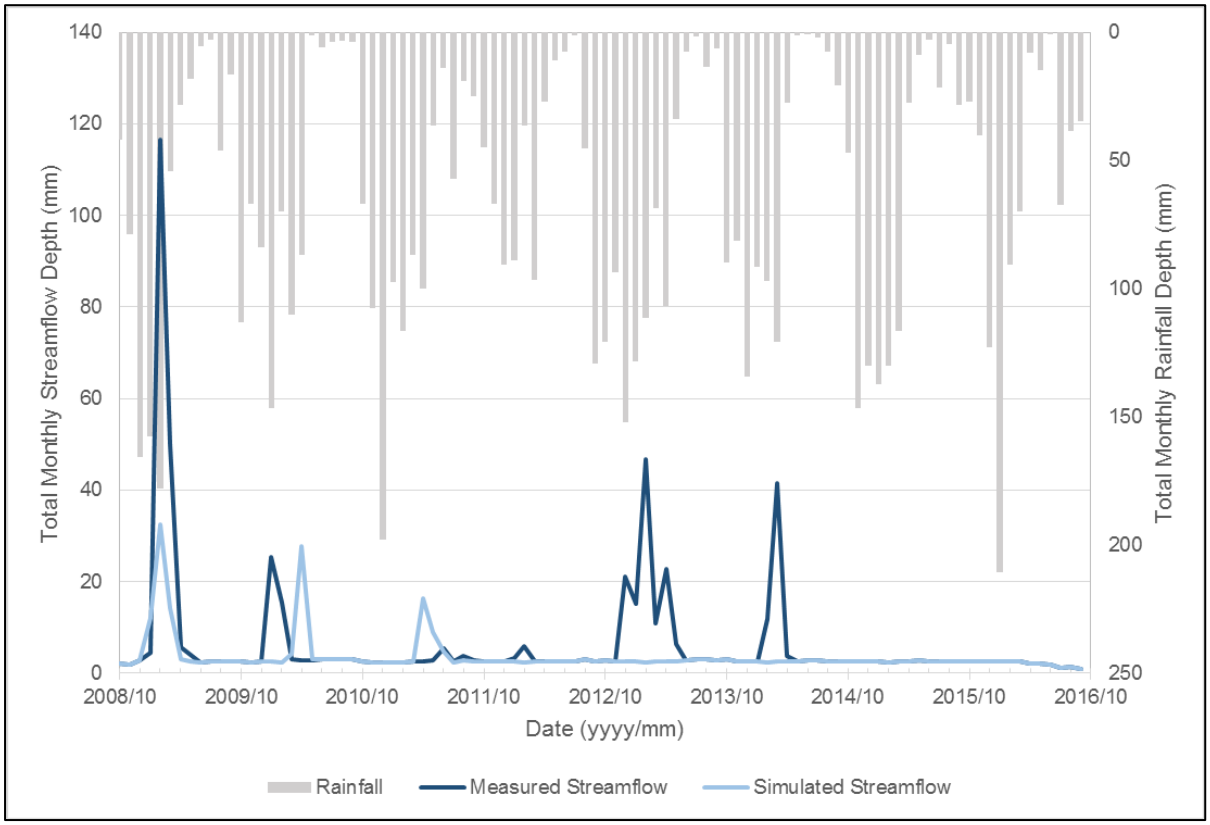

Figure 8-53 Total monthly rainfall and streamflow depths at gauge U2H048 (Midmar)

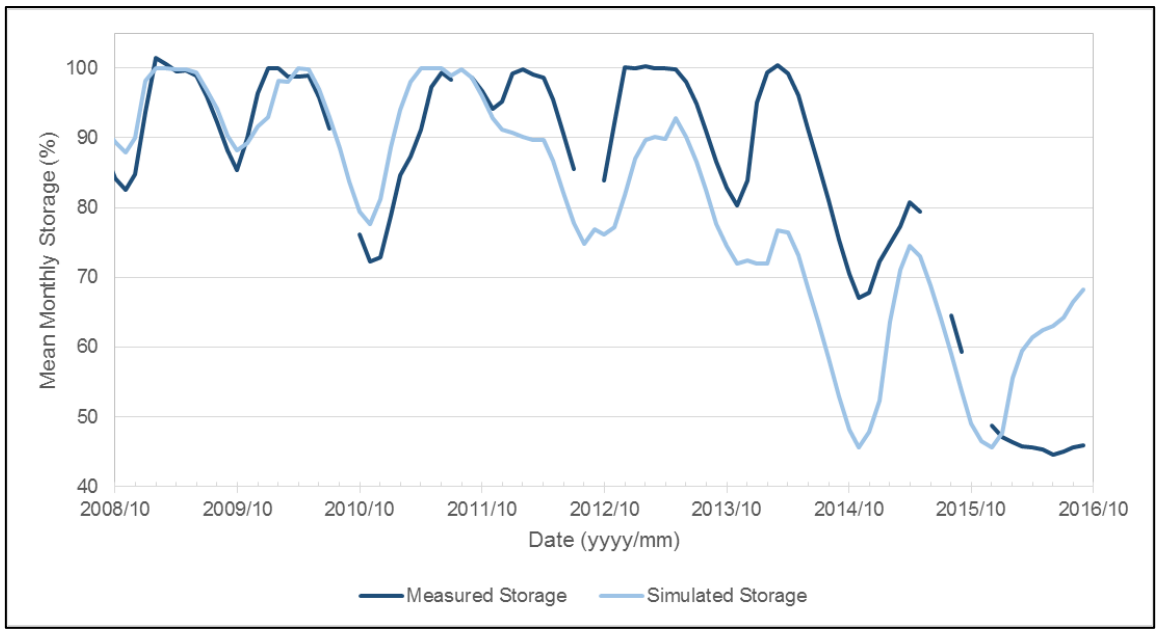

Figure 8-54 Mean monthly storage in Midmar Dam

The monthly time series of streamflow for gauge U2H006 on the Karkloof River are shown in Figure 8-55. The seasonal trends in the measured streamflow are represented well, but with some over- or underestimations in the magnitude of the flow depths in some seasons. The trends in the simulated flows appear to be associated with the trends in estimated rainfall.

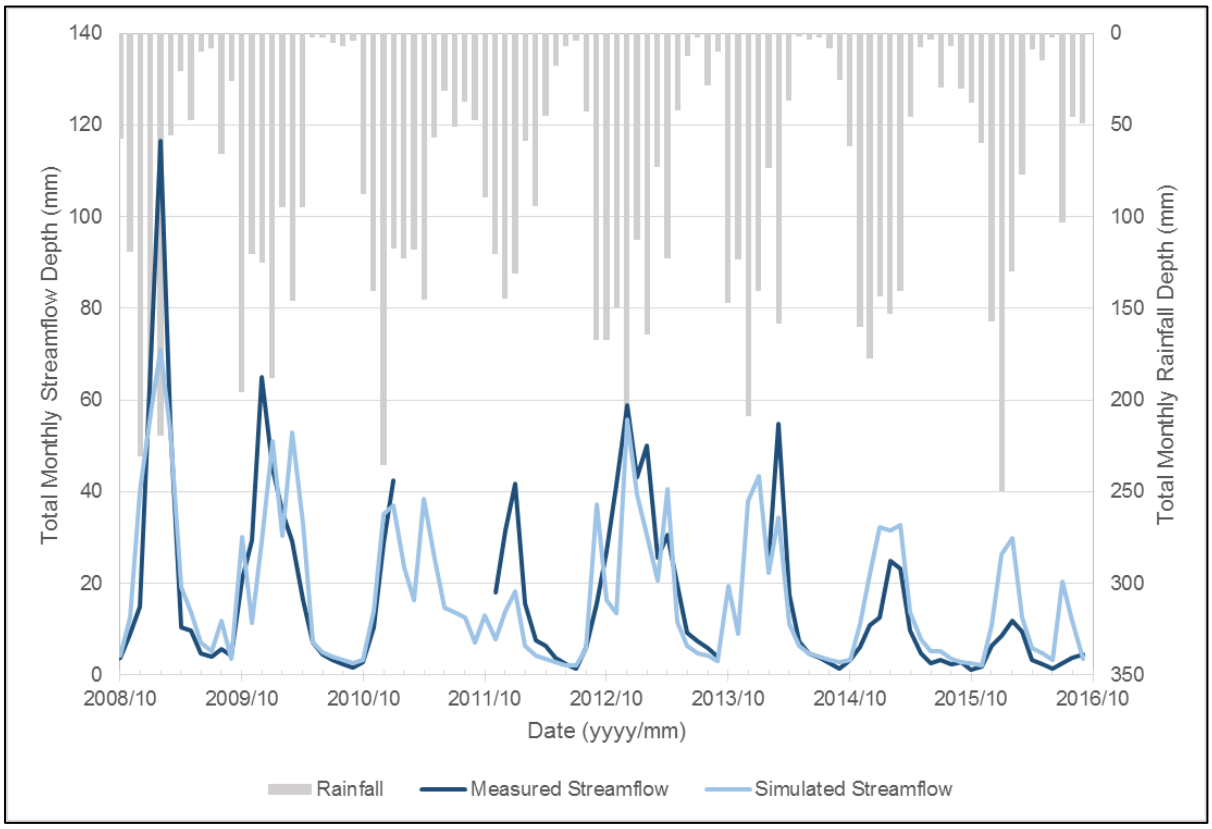

Figure 8-55 Total monthly rainfall and streamflow depths at gauge U2H006 (Karkloof)

The monthly time series of streamflow for gauge U2H014 on the uMngeni River just downstream of Albert Falls Dam are shown in Figure 8-56, and the monthly time series of measured and simulated monthly average storage for Albert Falls Dam are shown in Figure 8-57. The trends in the measured streamflow are represented well. As expected the measured flow releases from Albert Falls Dam result in good simulations during the non-spill periods. The dam spills in the 2008/2009 and 2009/2010 years despite an under simulation in the 2007/2008 warmup year. The dam spills later than it should in the 2009/2010 year due to the timing of flows from both gauge U2H006 and U2H048 upstream. The measured storage data indicates a substantial overestimation of inflows to the dam during the 2010/2011 year, possibly related to flows at U2H048 upstream being over simulated during this period, and unfortunately there was almost a whole year of measured flow records missing at gauge U2H006. The storage during the 2012/2013 and 2013/2014 years were well simulated despite under simulation of inflows and flows at U2H014. However the severe drawdown during the subsequent drought years is poorly represented due to over simulation of inflows from upstream.

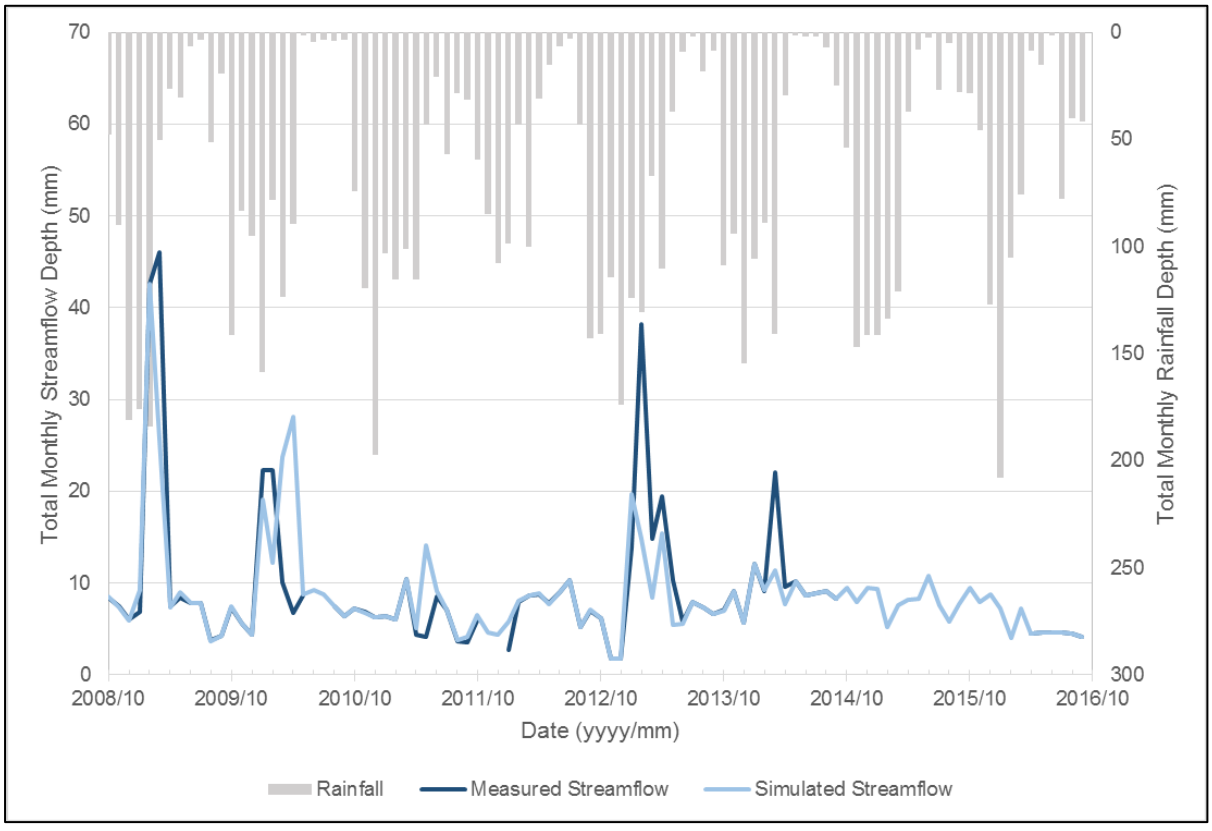

Figure 8-56 Total monthly rainfall and streamflow depths at gauge U2H014 (Albert Falls)

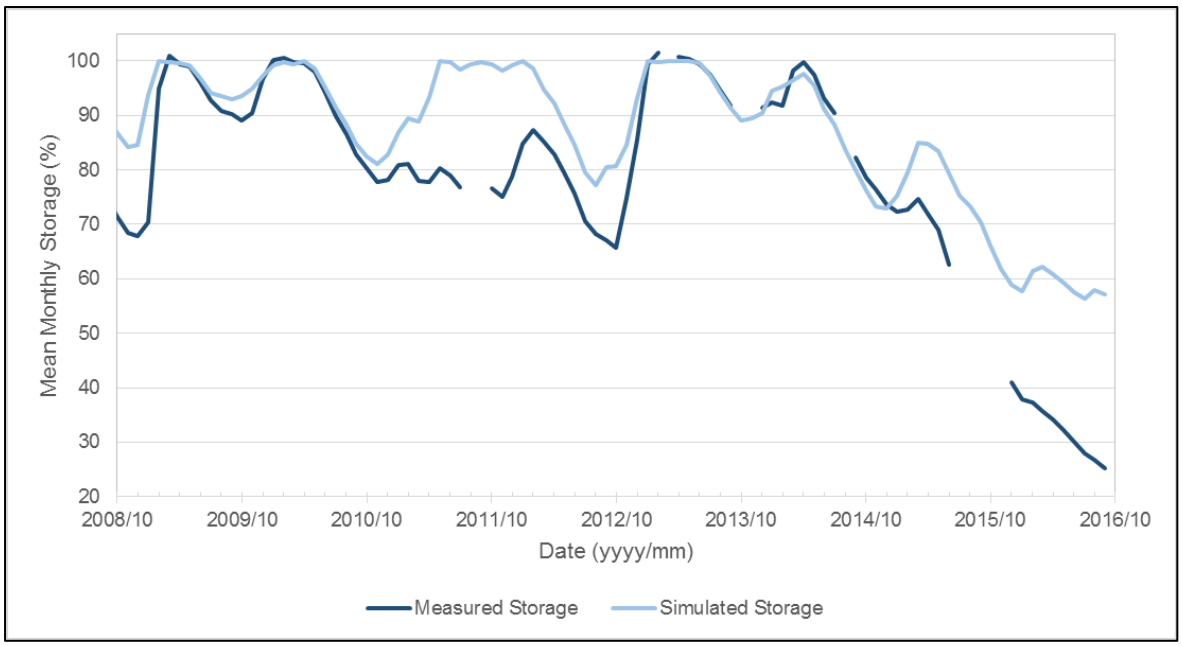

Figure 8-57 Mean monthly storage in Albert Falls Dam

The daily measured and simulated flows at gauges U2H007, U2H013 and U2H006 were also compared visually by graphing the time series to discern whether there were additional possible causes for the poor regression statistics. It was observed that when significant runoff producing daily rainfall occurred, the simulated daily streamflow values peaked on the same day as the rainfall event occurred (as expected from the *ACRU* runoff algorithms), but that the measured daily streamflow values usually peaked the following day with lower flow values. Examples of this are shown in Figure 8-58 (U2H007), Figure 8-59 (U2H013) and Figure 8-60 (U2H006) for the 2008/2009 summer rainfall season. The measured flows are represented by a dark blue line and the flow simulated in *ACRU* by a light blue line. It was concluded that the poor degree of statistical association between the simulated and the measured streamflow was most likely to be due to: (i) differences in actual and estimated rainfall volumes, and (ii) the mismatch in the timing of flows. It was also concluded that the mismatch in the timing of flows is possibly partly due the *ACRU* model not lagging and attenuating flows as they proceed down river reaches and through dams.

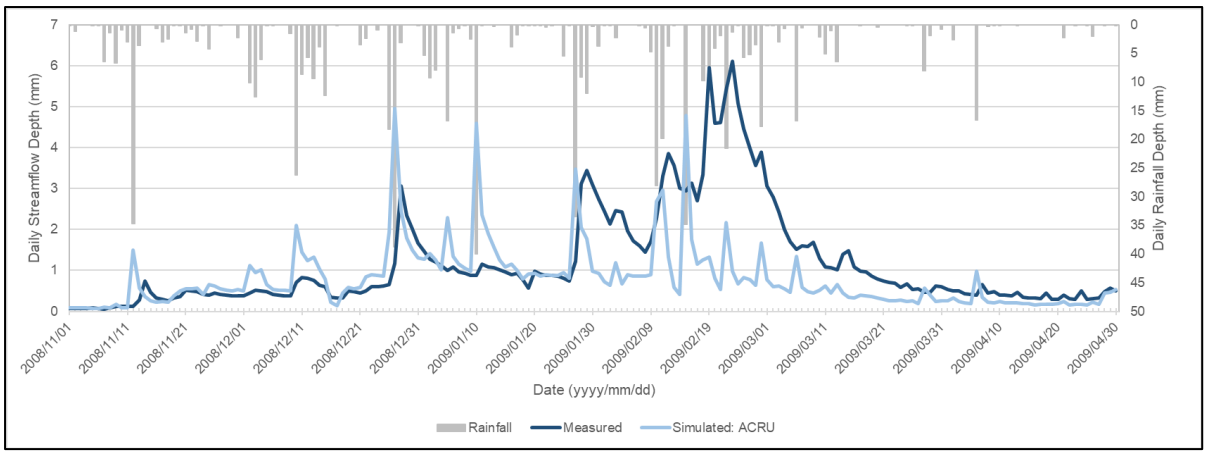

Figure 8-58 Daily rainfall and streamflow at gauge U2H007 (Lions River) for 2008/2009

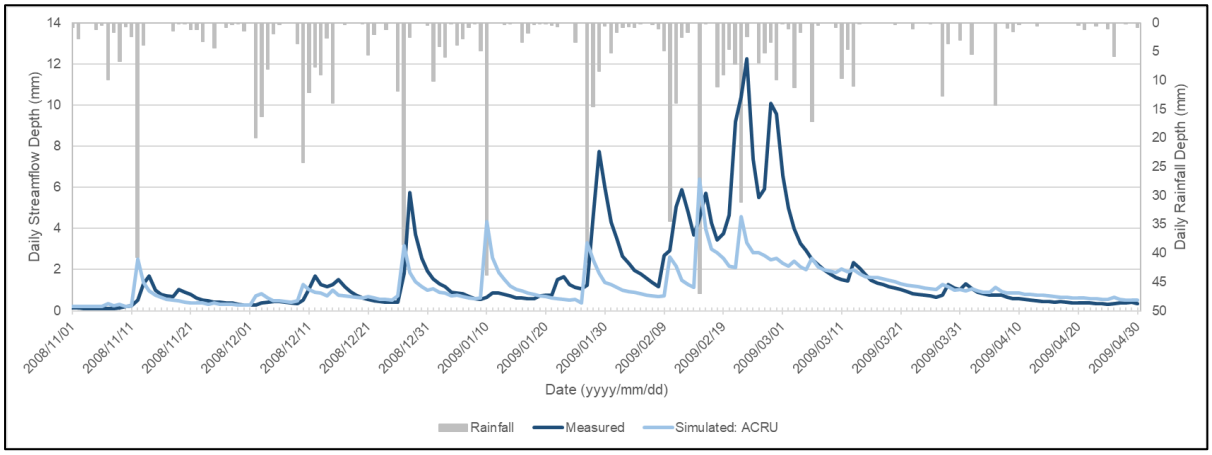

Figure 8-59 Daily rainfall and streamflow at gauge U2H013 (Mpendle) for 2008/2009

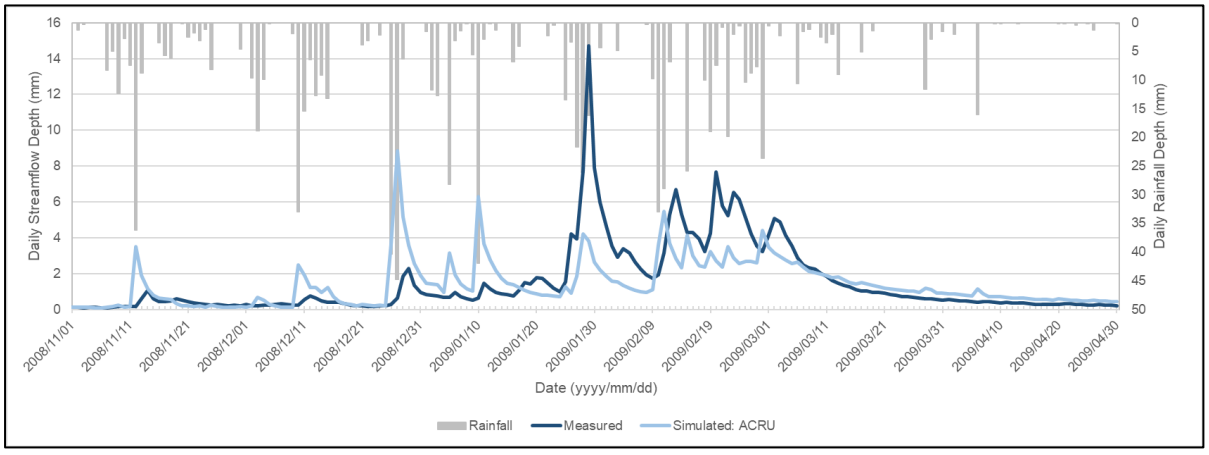

Figure 8-60 Daily rainfall and streamflow at gauge U2H006 (Karkloof) for 2008/2009

Configuration of the *ACRU* model for the upper uMngeni catchment was done at a sub-Quaternary Catchment scale, with detailed representation of the different land cover/use classes and detailed representation of dams with their contributing areas. All model parameters were based on recommended values resulting from the application of *ACRU* over many years in many different catchments. Although the urban areas within the upper uMngeni Catchment are not extensive the estimation of urban water use is one source of uncertainty due to possible inaccuracies in the estimation of population in each catchment and per capita water use. Industrial water use was not represented due to unavailability of data. There is a substantial amount of irrigated agriculture in the catchment, which is another source of uncertainty, as many assumptions had to be made regarding irrigation water sources and scheduling. Therefore, though it is possible that the model configuration may be improved, it is not expected to be the main source of the poor simulation results. The daily time series variables quantifying the inter-catchment transfer, abstractions from Midmar Dam and flow releases from both Midmar and Albert Falls Dams, were based on measured flow data. Therefore, apart from the uncertainties related to the measurement and processing of these datasets, these engineered flows are judged to have been accurately represented.

Poor estimation of the meteorological driver variables, is most likely to be the main cause of the poor simulation results. The reference potential evaporation (ET_0) data was not verified in this study, but the limited verification by Pegram *et al.* (2010) of the SAHG ET_0 estimates indicated good correlation with estimates based on measured meteorological forcing variables. In the relatively high rainfall upper uMngeni Catchment, with a strong seasonal variation in rainfall, and rainfall frequently occurring in the form of high intensity storms, rainfall is the primary driver of hydrological responses. A sparse rain gauge network makes it difficult to estimate catchment rainfall accurately. The application of remotely sensed rainfall is possible but also requires rain gauge data to do localised corrections However, verification of these spatial rainfall estimates can only be done indirectly through hydrological modelling with which there are associated many other uncertainties.

8.11 Description of Items in Resource Base Sheet of Water Resource Account

The individual items of the water resource accounts, in the form of the modified WA+ Resource Base Sheet described in Clark (2015b), are briefly described in Table 8-7.

Table 8-7 Description of items in the Resource Base Sheet

Account Item		**Description**
Precipitation	[1]	Precipitation as an inflow to the catchment
$Q_{in\ SW}$	[2]	Surface water inflow (e.g. from upstream catchment)
$Q_{in\ GW}$	[3]	Groundwater inflow (e.g. from neighbouring catchment)
$Q_{in\ Transfers}$	[4]	Inflow to a catchment as inter-catchment transfers
Gross Inflow	[5]	Gross inflow to the catchment [1] + [2] + [3] + [4]
$\Delta S_{f\ SW}$	[6]	Decrease in surface water store (e.g. in dams)
$\Delta S_{f\ SoilM}$	[7]	Decrease in soil moisture store
$\Delta S_{f\ GW}$	[8]	Decrease in groundwater store
Net Inflow	[9]	Net inflow to the catchment accounting for change in storage within the catchment [5] + [6] + [7] + [8]
Landscape ET	[10]	Evaporation of naturally occurring water from the landscape
Exploitable Water	[11]	Water that could be exploited [9] – [10]
Reserved Outflow	[12]	The portion of utilizable flow that is reserved as outflow from the catchment, for example, to meet environment requirements or downstream requirements
Available Water	[13]	Exploitable water – reserved outflow [11] – [12]
Incremental ET	[14]	Evaporation of water that would not naturally occur (e.g. irrigated water)
Non-recoverable Flow	[15]	Flow that is utilised but can't be recovered because, for example, it is polluted
Utilized Flow	[16]	Portion of available water that is utilized [14] + [15]
Utilizable Outflow	[17]	The portion of utilizable outflow that was not utilized, but could have been utilized, and will flow out of the catchment [13] – [16]
Consumed Water	[18]	The water that was consumed (depleted) within the catchment and is not available for re-use [10] + [14] + [15] or [10] + [16]
Total Evaporation	[19]	Total evaporation within the catchment [10] + [14]
Interception	[20]	The precipitation and irrigated water that has been intercepted by vegetation and other surfaces and has subsequently evaporated
Transpiration	[21]	The water transpired by vegetation
Soil Water Evaporation	[22]	The water evaporated from the soil
Open Water Evaporation	[23]	The evaporation from open water surfaces (e.g. dams)
Outflow	[24]	Water flowing out of a catchment = net inflow – consumed water [9] – [18] or [12] + [17]
$Q_{Out\ SW}$	[25]	The water that flows out of the catchment as surface water (e.g. to a downstream catchment)
$Q_{Out\ GW}$	[26]	The water that flows out of the catchment in the form of groundwater (e.g. to a neighbouring catchment)
$Q_{Out\ Transfers}$	[27]	Outflow from a catchment as inter-catchment transfers