Framework for Interoperable Freshwater Models: Testing and Recommendations

Prepared for Ministry of Business, Innovation and Employment

February 2014
Figure 3: Illustration of how OGC Web Processing Services, (WPS), catalogue services (SC/W) and Web Feature/Coverage services (WFS/WCS) could be used to mediate between model computation components and data sources.

Tables
Table 1: List of 12 candidate frameworks screened out after preliminary evaluation.
Table 2: List of six candidate frameworks selected for detailed screening.

Reviewed by
G. McBride

Approved for release by
C. Howard-Williams

Formatting checked by:
LAB
Executive summary

Purpose and scope

This report summarises the three-year MBIE-funded project Framework for Interoperable Freshwater Models (FIFM) which investigated flexible computer-based frameworks for integrating freshwater models. Key results and findings are presented and recommendations for further work are made.

The project addressed the need for integrated freshwater modelling to support freshwater policy and management, as expressed by the Land and Water Forum, the Ministry for the Environment Water Research Strategy, and the Freshwater Reforms. The project focussed on frameworks for interoperability, that is, software infrastructure that enables various models of water quality and quantity to interoperate. An interoperable freshwater modelling framework would enable the co-ordinated, integrated, and flexible coupling and re-use of a diverse range of models and associated data sources, leading to improved integrated modelling and freshwater management. Such interoperability contrasts with the current situation in New Zealand, where there are disparate freshwater models with only occasional inflexible integration.

The intended end-point of this project was a plan for future development of an interoperability framework. That point was reached by conducting preliminary proof-of-concept testing (i.e., demonstrating that the framework could work in practice) to assess the difficulties and advantages of a linking framework in relation to end-user requirements, and for the range of data sources and models used in New Zealand. Subsequent stages of funding would be required to develop or modify the framework to achieve a fully functional or useful implementation.

We purposefully did not embark on design of a new framework. Rather, it was preferable select an existing framework and trial it in relation to end-user needs and in the context of New Zealand models and data sources. The scope was restricted to models of freshwater quality and quantity and associated data, to ensure that the project was tractable. This focus does not preclude extension to other domains such as ecology, economics or planning in the future.

Methodology

The project involved the following components:

- Developing a comprehensive publicly-accessible wiki (https://teamwork.niwa.co.nz/display/IFM) to disseminate information and findings from this project.

- Holding an end-user workshop to identify user needs for interoperability, and documentation of the findings in a report (Snow et al. 2011).

- Compilation of an inventory of freshwater models and environmental data currently used in New Zealand, categorised in relation to a common set of attributes relevant to coupling. This information is available in a structured database on the project wiki.
- Development of a tool, ModelVis, to identify and display potential linkages between current freshwater models and data sources (see Elliott et al. 2012 and the project wiki).

- Reviewing a range of existing interoperability frameworks and screening them in relation to user needs, leading to the selection of a framework for more detailed hands-on assessment. This led to a second project report (Elliott et al. 2012).

- Testing the preferred framework, OMS3, for its ability to link freshwater models commonly used in New Zealand (summarised in Section 3). This involved setting up the hydrology model WATYIELD in OMS3, converting it to a spatio-temporal model, linking to climate time-series provided by standard web services, obtaining spatial data from standard geospatial data sources, manipulating data with geo-processing routines, and displaying the model results as a time series. We also set up the farm nutrient loss model OVERSEER as an OMS3 component (either a simplified version of the engine run as a dll, or as a web service) with a simple user interface to modify rainfall. We linked the OVERSEER component to the network aggregation and routing components of the catchment model CLUES. We also attempted to set up components of the agricultural point-scale simulation system APSIM in OMS3.

- Results of initial testing were presented at an end-user webinar, and comments from this webinar were incorporated into final recommendations.

- Developing recommendations for future framework implementation, taking into account results of testing, end-user feedback, and further consideration of alternatives and recent developments in the literature (Sections 4 and 5) (this report).

**Key results**

End-user needs assessment confirmed the strong perceived benefits of an interoperable modelling system, such as improving integration of models across freshwater domains, re-using model components, and making better use of the increased availability of environmental datasets and associated standards.

The review of models and datasets identified a large number of freshwater models and associated datasets in New Zealand. Documenting these models and data sources in a structured database enabled better understanding of the attributes of the models, especially the potential for linking them. The ModelVis tool enables interactive searching for models by a limited set of attributes, along with graphical display of the potential linkages between models.

Over 18 existing environmental modelling frameworks were identified from a literature review. Subsequent screening identified a shortlist of six existing frameworks that met many of the criteria identified by end-users and the project team. Considering that several frameworks had promise, it was considered best to conduct hands-on testing of one of the most promising frameworks, rather than developing a new framework. The report on this phase of the project (Elliott et al. 2012) also summarised key technologies and concepts in the interoperability arena, such as web services and associated standards for providing data over the web.
The framework selected for testing was OMS3¹ (David et al. 2013). That framework was developed by the US Department of Agriculture (USDA) for component-based model and simulation development on multiple computer platforms. It was developed mainly for simulations of effects of agricultural systems on water quantity and quality, although broader hydrological modelling is also done with this framework and the framework design is fairly general. A key aspect of OMS3 was the desire of the USDA to foster both new model development and re-use along with integration of existing legacy models.

From the standpoint of the project, OMS3 met most of the key and high importance user needs identified at the end-user workshop including:

- Is open source.
- Receives substantial investment from the USDA as part of the large Conservation Delivery Streamlining Initiative (CDSI) programme and therefore has a high likelihood for continuity and longevity.
- Is supported by integrated development environments (IDE) for developing testing and running interoperable models.
- Has its own integrated user interface for ease of use.
- Has a considerable and active international user community.
- The primary developer was open and responsive to questions and queries.

Our trialling has demonstrated the feasibility of using OMS3 to couple freshwater models together in a variety of ways, and also of coupling models to input environmental datasets. All of the tests were successful, except for setting up APSIM. For example, we were able to set up components from a variety of sources, link and run them, develop simple user interfaces, visualise the results, and access data and simple models provided as web services. In some cases, especially where the user interface was coupled closely to the calculations, it was best to re-write the model in Java rather than attempting to work with the existing code. It was necessary to have competent Java programmers involved in setting up OMS3 components, writing the scripts that link files, and writing user interfaces. Overall, we demonstrated that OMS3 has many of the desired building blocks for a modelling framework for New Zealand.

Despite these positive signals and successful testing, users are not yet in a position to fully embrace OMS3 for several reasons:

- Funding is needed to support transferring models to the framework, maintaining the models and components, and contributing to international development communities.
- There are some weaknesses in the framework, including difficulty in setting up models written using Microsoft .NET, there is no current publicly-available repository of components, documentation is patchy, user interfaces need to be constructed from scratch, there is no core support for many data structures commonly used in hydrology (such as networks), there is little core functionality

¹ [http://www.javaforge.com/project/oms](http://www.javaforge.com/project/oms)
for visualising results, and there is no core geospatial awareness. Many of these shortcomings could be addressed by building new components, especially ones using third-party libraries such as GDAL, and over time more components and utilities will become available. The extra work required to build such utility components presents a barrier to adoption by a wide user base.

- The ‘indirect users’ have called for demonstrating of a polished showcase demonstration project, to demonstrate complex arrangements of model components and a polished user interface and visualisation.
- We have not yet tested OMS3 for computationally-demanding applications.
- The user base is currently small, and there is reliance on continued funding from the USDA.
- Recent adoption of OpenMI as an OGC standard for interoperability has created some confusion about the best future pathway. We do not propose shifting to OpenMI at this stage, for reasons outlined below.

Given these reservations, we re-considered alternatives to OMS3, taking into account recent literature and development.

An important development since we undertook framework screening was the ratification of OpenMI as a standard by the Open Geospatial Consortium (OGC). Potentially this could lead to wider uptake of OpenMI, especially considering the rapidly-increasing uptake of OGC standards for time series and geospatial data (including within New Zealand). Also, there are several recent papers on integrating OpenMI with other frameworks or data sources (e.g., Castronova et al. 2013b). Despite these developments, we do not currently recommend adoption of OpenMI standards in New Zealand for several reasons: there is little uptake of the OpenMI 2.0 standard by other groups so far; there are currently no open-source software development environments to aid implementation of models in the OpenMI standard; and OMS3 could be made OpenMI compliant, which would mean that OMS3 would not need to be abandoned. We recommend undertaking a periodic review of progress with OpenMI standards as part of continued development of an interoperable modelling framework for New Zealand.

A further development in interoperability is the approach of running models as a web service, building on successful initiatives in the data provisioning area. Models as a service is a promising and active area of development, and several recent papers have pointed to this approach as an important future direction. We have already successfully tested some simple models provided as web services in OMS3, and we consider that web services could overcome institutional, IP, and technical barriers to interoperability. However, the technology for running linked models as web services is young, and there are some inherent pitfalls such as delays due to transferring data over the web, so we proposed a staged process to adoption of web services.

**Recommendations for further work**

As a result of this project, the following further steps are recommended:
1) Develop a showcase integrated model built with OMS3, scaling up from our experience to date. This showcase will demonstrate the capabilities of OMS3 to the range of end-users, as well as provide a deeper and more demanding level of testing (for more complex and computationally-demanding models and more elaborate user interfaces).

2) Further exploration of using models as a web service. This exploration would follow a staged process, initially adopting a set of standards for New Zealand data, then setting up OMS3 components to access these data through standard web services such as SOS and WPS, setting up web-based models as web services linked to OMS3, setting up computational components as web services using OMS3, and finally moving to more web-centric linking and co-ordination technologies as they become available. This will provide early benefits (enhanced data provision to models) with a longer-term pathway to frameworks that use models as a web service, with associated benefits such as institutions being able to maintain control of their models and being less bound by programming language and platform dependencies.

3) Establish baseline funding and institutional arrangements for an interoperability framework to co-ordinate efforts across and within institutions, build consensus on standards adoption, keep abreast of developments (such as OpenMI), remain part of international communities, assist with setting up models in the framework, and provide education and training. Ideally, the local activities would be funded from a dedicated MBIE or other central government funding stream, as other funding streams are largely committed, and individual programmes are usually more interested in developing data or models in their immediate domain of interest. An alternative is to build interoperability as a project within the National Science Challenge or Centre of Research Excellence programmes.
1 Introduction

1.1 The Need

Both the Land and Water Forum (LAWF\textsuperscript{2}) and the MfE Water Research Strategy\textsuperscript{3} identify the need for robust models to assist the development and implementation of freshwater policy and management, and especially the need for models to interoperate. Interoperability is the ability for models, data sources and user interfaces to exchange data in a co-ordinated and integrated manner. The goal of interoperability includes the efficient reuse of existing models and coupling of new models, the minimisation of repetitious and error-prone manual data input and exchange between models, more efficient combination of models to provide systems-level environmental predictions. Interoperability makes model development and usage more efficient by reducing needed resources and also has the potential to make the modelling process more robust by providing provision for auditing of the exchange of data between models. Key items in the relevant reports are:

LAWF Third Report Recommendation 63
“...All parties, central and local government, industry and science providers should continue investment in the development of models (including development and prioritisation of a limited number of interoperable models) and measurement-based monitoring systems for practical application to water quality management. Investment should be based around partnerships and guided by a national strategy that ensures co-ordination of available resources. This should include clear guidance and protocols on how models, monitoring systems and their output data, should be used in the development, implementation and enforcement of water quality policy.”

“...Capability, capacity and the use of information will be critical issues in implementing changes to water management, particularly in the period when objectives and limits, and the methods and tools to achieve them are being developed. Investment is required to speed the development of a small number of interoperable models and efforts are required to improve the communication of science to lay audiences and the integration of Mātauranga Māori into our decision-making processes. All sectors need to invest in the development and implementation of extension programmes to ensure continued and accelerated uptake of good management practice.”

These imperatives are echoed in the proposed Freshwater Reforms\textsuperscript{4}, where integrated modelling plays an important role in catchment deliberation processes.

1.2 The project

The MBIE-funded project “Framework for Interoperable Freshwater Models” (FIFM) aimed to test to proof-of-concept stage a computer-based framework that would allow models used in

\textsuperscript{2} www.landandwater.org.nz/includes/download.aspx?ID=124767  
\textsuperscript{3} mfe.govt.nz/publications/water/water-research-strategy  
\textsuperscript{4} http://www.mfe.govt.nz/publications/water/freshwater-reform-2013/
the freshwater arena to interoperate by sharing data, communicate with each other, and make use of shared visualisation or mapping tools.

The intention was that the resulting framework would ultimately:

- enable various models of water quality and quantity used in New Zealand, to interoperate, that is, to work together in a co-ordinated and integrated fashion and link with data sources
- adapt or re-use existing frameworks and software tools, preferably open-access ones
- be freely available
- be sufficiently easy to use that the effort of using the framework is significantly less than the benefits accrued from using it, and
- make use of current and emerging computer technologies.

The project addressed general frameworks for interoperability, rather than implementing a system to integrate a specific set of models. Rather than focusing on coupling a selected few models, the project investigated more flexible and general approaches. A model framework is a software system that allows various components such as models, data sources, user interfaces, and data views to be linked and function in a co-ordinated and integrated way. Through the provision of standards and common core functions such as data plotting, frameworks provide for re-use of models and utility components, thereby reducing the software development effort required for model integration and system maintenance.

This three-year project provided an initial foray into the use of environmental modelling frameworks for New Zealand. To ensure that the project was tractable and focused, the scope was restricted to models of freshwater quality and quantity and associated data. This focus does not preclude extension to other domains such as ecological, economic or management aspects in the future.

The end-point of FIFM was, as originally intended, a plan for future development of an interoperability framework. That point was reached by conducting preliminary proof-of-concept testing to assess the difficulties and advantages of such a linking framework in relation to end-user requirements and the range of data sources and models used in New Zealand. Subsequent stages of funding would be required to develop or modify the framework to achieve a functional or useful implementation.

We purposefully did not embark on design of a new framework. Rather, the project team considered that it is preferable to assess selected frameworks against the needs by trialling them. This is because existing frameworks of considerable sophistication are available and designing and implementing a new system involves a large input of resources.

This project has:

- Developed a public wiki https://teamwork.niwa.co.nz/display/IFM to disseminate information and findings from this project.
- Compiled an inventory of freshwater models and environmental data currently used in New Zealand, categorised in relation to a common set of attributes relevant to coupling.

- Held an end-user workshop to identify user needs and desires for interoperability.

- Developed a tool, ModelVis, to identify and display potential linkages between current freshwater models and data sources.

- Compiled an inventory of interoperability frameworks and assessing them against user needs.

- Tested the preferred framework for functionality for linking freshwater models commonly used in New Zealand.

- Developed a recommendation for future development to meet identified needs for an interoperable freshwater modelling framework.

This report summarises key aspects of the project.
2 Early phases of the project

2.1 Phase 1: Assessment of End-User Needs

In June 2011 the FIFM project held a workshop with end-users to both inform potential end users about the project and to elicit their needs and requirements for linking frameworks that would operate in the freshwater modelling arena (Snow et al. 2011). The workshop was attended by approximately 45 participants from a range of government, industry, research and iwi organisations.

The workshop addressed four key aims that were necessary to plan the next phase of the project:

1. Define the types or categories of end users for a FIFM.
2. Understand the range of needs from the end users and their expectations for use.
3. Understand the potential benefits that users could get from a FIFM.
4. Identify key requirements that might impact on technical decisions to be made when developing a suitable framework.

The workshop, and preliminary planning before the workshop, succeeded in partially addressing these aims. Key findings are presented below.

2.1.1 Types of end users for a FIFM

Potential end-users for an interoperable modelling framework will always span a wide range of interests and classification into discrete categories will be problematic. Before the workshop we hypothesised that attendees would be able to loosely identify themselves on a spectrum from indirect user to interpreter to direct user to a developer. Following is a description of each of the four user groups:

- **Indirect Users** – policy makers, resource managers, and others who want to ask relevant questions of freshwater models and apply the results to assist with policy development and resource management but who by and large do not want to use or operate the models themselves.

- **Interpreters** – users who sit “between” the indirect users and direct users to help formulate appropriate questions (e.g., scenarios), analyse results, and generate suitable outputs (e.g., maps, charts, other materials) to aid indirect users and the broader community of interest.

- **Direct Users** – modelling analysts with the scientific and technical knowledge to apply models competently and with confidence, and modellers who undertake research to develop, test and apply new models.

- **Developers** – informatics researchers, programmers and software engineers who help design, implement and maintain the FIFM framework, and join new models and data into the framework.
2.1.2 Needs of End Users

Overall there was broad agreement among the different groups of end users, although different aspects were of more importance to some groups than others. The common points, and points of potential disagreement, follow below.

All end-users agreed that a FIFM must be publically available, not be ‘in house’, and usage and development must not be restricted to a small group of users. This was seen as important to ensure robustness and transparency and also to ensure that indirect users were not locked into using a small group of experts, but could instead pick the right team for the task at hand. However there was a range of views, which are not necessarily in conflict, regarding the degree of openness and extensibility of the framework.

Some, primarily indirect users, desired that the framework be thoroughly reviewed with scientifically approved combinations of models. We believe that desire was primarily driven by the needs of end users who are not themselves freshwater experts. They require assurance via appropriate processes (e.g., independent review) that a particular application of the FIFM would be appropriate and withstand independent scientific scrutiny.

Another group of end users, more predominantly at the developer end, emphasised that the FIFM should make it easy to add new models and databases.

These points are not necessarily in conflict if the framework includes something akin to an optional certification process. Certified models and databases would have undergone a defined process of scientific review and testing. Other models or combinations of models could be listed as experimental until they achieve certification, and some might never go through the certification process. While these points might not necessarily place restrictions on the FIFM itself they do have implications for the design and implementation of the FIFM, especially any required metadata and other information and the processes to update this information.

The need for the ability to audit the FIFM usage was mentioned in more than one context and was strongly linked with the desire to develop and explore scenarios, especially by indirect users. At one extreme, ‘audit’ was taken to mean that a particular usage of the FIFM could be passed to an independent expert and reviewed for fitness-for-purpose. Other end users desired auditability to provide a complete record of the choices made, data used, etc., and was primarily for their own information and for repeatability. This need was also related to the expressed desire to assess multiple scenarios with an interface that would clearly identify the differences between the scenario assumptions and projected outcomes. There was an expressed desire that the FIFM include good user interfaces and this might also be related to the points about auditability and scenario testing.

More than one group expressed the desire that the framework would clearly identify the sources of uncertainty, the assumptions made, and the confidence bounds on the outputs. Technologies for quantifying the uncertainty in individual model outputs due to input and parameter uncertainty are reasonably well established and the primary technical challenge is one of processing speed. Technologies for assessing the impact of model conceptualisation and model uncertainty on the outputs are developing. More work is required to understand how to calculate, visualise, and use confidence intervals in several, often correlated, model outputs.
All groups expressed the need for a good collaborative governance structure around the FIFM. The governance role should include the scientific review / accreditation processes as well as ensuring that the IFM was continually developing to keep pace with new IT infrastructure as well as new models that should be linked into the framework.

### 2.1.3 Potential Benefits from a FIFM

The primary benefits identified were increased efficiency and more robust model estimates. All groups assumed that if there was increased transparency about the models and fewer ‘in-house’ models then the energy invested in arguments about the models and their results would decrease. In turn it was assumed that this would lead to better and more timely policy and regulatory decisions. Some groups identified that the cost of provision of modelling services might decrease. Other benefits include the ability to connect models to answer more complex questions for integrated decision-making.

### 2.1.4 Key Technical Requirements of a Suitable Framework

Given the need to accommodate a broad range of users with varying degrees of knowledge and the limited time for the workshop, it was not practicable to elicit many specific technical requirements or specifications for the framework. However, some requirements emerged from the discussions throughout the day, which are summarised as follows:

- Databases and other input data must be made readily available.
- A wide range of models should be included and the FIFM should be designed to ensure extensibility (the ability to add different models, and models of different types).
- A scenario manager and tools for visualising data and results would be highly desirable.
- Proper document methodologies are needed to provide an ‘audit-trail’ capturing the full range of choices made by direct users and/or interpreters. Models run previously should be able to be re-run.
- The FIFM must be open-source.

There was no discussion recorded about the relative benefits of frameworks developed mainly for freshwater modelling compared to more all-embracing frameworks. However the benefits of those different approaches were considered during the review of possible frameworks for trialling in the FIFM project.

### 2.2 Phase 2: FIFM Wiki

In Phase 2 the project team established a project Wiki to facilitate the sharing of ideas, data, and foster “interoperability” on a practical level (https://teamwork.niwa.co.nz/display/IFM/Framework+for+Interoperable+Freshwater+Models) (Figure 1). The Wiki was made publically accessible to meet the key attribute of openness as expressed by end-users at the earlier project workshop. The project team made data and information available for public inspection and downloading as the project progressed. Project team members also had access to protected access to change the Wiki contents and to access private areas where shared documents were developed in collaboration.
2.3 Phase 3: Inventory of Models and Data

Phase 3 involved an inventory of data and models relevant to freshwater modelling in New Zealand. Over 40 models were inventoried. Information about the models is presented and available on the FIFM Wiki at https://teamwork.niwa.co.nz/display/IFM/Compilation+of+models+and+their+attributes. The inventory includes information about the scope, availability, purpose, inputs, outputs, history cost and IP as well as several other more technical categories of information.

The Wiki also identifies other model summaries that might be useful to users – see https://teamwork.niwa.co.nz/display/IFM/Other+NZ+inventories.

2.4 Phase 4: ModelVis Development

During consultation, potential users identified that they faced significant issues in understanding the range of models that were available and used in New Zealand, how those models related to each other (e.g., were two models competitive or might one provide input data for the other?) and what the data requirements of models were. In many respects the Inventory of Models (Section 2.3 and also the Wiki at https://teamwork.niwa.co.nz/display/IFM/Compilation+of+models+and+their+attributes) together provided the requisite information, but they do not show potential interrelationships between models easily. The tool ModelVis is an alternative way of viewing the same information but concentrating on a visualisation of the interrelationships between models rather than details about the individual models.
ModelVis allows users to search for models with particular attributes (e.g., that work at a particular scale or that produce certain outputs), how a particular model might interrelate with others, and where to find additional information about a selected model. These features are shown in summary form in Figure 2. ModelVis is available to users at https://teamwork.niwa.co.nz/display/IFM/Relationships+between+models+-+the+ModelVis+tool. The key method of identifying potential linkages was to identify items where a model output or data type would match the inputs needs of another model. This required categorising the data items and model inputs and outputs according to a common set of attributes and categories (for example, whether a model uses or outputs flow time series).

![Figure 2: The ModelVis screen showing the model criteria and list, linkage diagrams and summary information.](image)

### 2.5 Phase 5: Screening of Frameworks

#### 2.5.1 Inventory of Frameworks

Information was gathered on a number of existing frameworks of relevance to freshwater modelling. The search was based on published journal articles and conference proceedings, web searches, discussions with developers of frameworks, and personal knowledge. Information on each framework was gathered based on readily-accessible information (published on the web), and information was collated in a common format. The list of frameworks was not exhaustive. In some cases a framework was investigated but attributes were not collected or only partial information was collected if it was clear that the framework was not relevant for our needs, e.g., it was no longer under active development or use.
For each framework an extensive collection of attributes and information was compiled under the broad areas of general information, scope, development history, cost and intellectual property considerations, applications, technical considerations, user information, other information, relevant internet links and references.

The initial search identified a total of 18 potential frameworks for further investigation.

### 2.5.2 Screening of frameworks

Screening involved assessing the information gathered for each of the 18 frameworks against a set of 35 criteria organised into 9 broad themes based on the end-users needs identified at the June 2011 workshop. Each criterion was ranked for importance (key, high, and medium). Any framework that did not meet a high proportion of key criteria was screened out from further consideration.

As a result of the screening, 12 frameworks were screened out (Table 1). Further information on the frameworks and the screening can be found in Elliott et al. (2012).

Table 1: List of 12 candidate frameworks screened out after preliminary evaluation.

<table>
<thead>
<tr>
<th>Name</th>
<th>Primary reason for screening out (as of 2012)</th>
</tr>
</thead>
<tbody>
<tr>
<td>APSIM framework</td>
<td>Not spatial.</td>
</tr>
<tr>
<td>Bespoke Framework Generator (BFG)</td>
<td>Prototype only, somewhat experimental, difficult to understand, unsure of longevity.</td>
</tr>
<tr>
<td>Delft-FEWS</td>
<td>Targeted at flood prediction, not easily extensible to other problem domains.</td>
</tr>
<tr>
<td>EnSym</td>
<td>Not flexible and is proprietary.</td>
</tr>
<tr>
<td>ESMF</td>
<td>Targeted at grid-based global/continental earth system modelling using high-performance computing.</td>
</tr>
<tr>
<td>GME (Generic Modelling Environment)</td>
<td>Not spatially-oriented. Seems to be sophisticated and flexible, though. Probably not well suited for integrating legacy models. Ongoing support uncertain.</td>
</tr>
<tr>
<td>GME (Geospatial Modelling Environment)</td>
<td>Currently more of an environment for manipulating and analysing geospatial data, rather than a modelling framework.</td>
</tr>
<tr>
<td>Hydrologists Workbench</td>
<td>More of a workflow tool for operation of rainfall-runoff models. Prototype only.</td>
</tr>
<tr>
<td>Hydromodeler</td>
<td>The uptake and support for this tool is uncertain. Support for spatial data is unclear. The Hydromodeler data access and visualisation component is more mature and supported. Hydromodeler did not accept OpenMI and includes a visualisation environment, so review this decision if OpenMI is promising.</td>
</tr>
<tr>
<td>ICMS</td>
<td>No recent development activity.</td>
</tr>
<tr>
<td>OASIS</td>
<td>System for running complex climate-ocean models on high-performance computing. Difficult to use for smaller freshwater models.</td>
</tr>
<tr>
<td>Seamless</td>
<td>OpenMI is the underlying model framework technology, so focus further assessment on OpenMI.</td>
</tr>
</tbody>
</table>
2.5.3 Short-Listed Frameworks

The remaining six frameworks were assessed against key criteria (Table 2). While high- and medium-priority criteria could also affect eventual selection of frameworks, at this stage the key-priority items provided sufficient differentiation for selection. The short-listed frameworks are listed below, and the assessment matrix is shown in Appendix A.

Table 2: List of six candidate frameworks selected for detailed screening.

<table>
<thead>
<tr>
<th>Framework</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSDMS Modelling Tool (CMT)</td>
<td>Framework software with models for simulation of earth surface processes such as sediment dynamics and hydrology, running on a High Performance Computing platform.</td>
</tr>
<tr>
<td>(Community Surface Dynamics Modelling System Modelling Tool)</td>
<td></td>
</tr>
<tr>
<td>OMS3 (David et al. 2013)</td>
<td>A framework developed by the US Department of Agriculture for component-based model and simulation development on multiple platforms. It is intended mainly for simulations of effects of agricultural systems on water quantity and quality. It includes a set of standard annotations to describe models, a model configuration editor, and a console for launching and monitoring models.</td>
</tr>
<tr>
<td>Object Modelling System v3</td>
<td></td>
</tr>
<tr>
<td>OpenMI</td>
<td>A specification for model linking that allows components running within a framework to be exchangeable. OpenMI consists of the OpenMI standard, and tools such as an SDK (development kit) and simple GUI based model configuration editors (for running one or more compliant models)</td>
</tr>
<tr>
<td>OpenPALM</td>
<td>Framework aimed at dynamic coupling of numerical models with high performance computing. The package provides a graphical user interface for model coupling, scheduling and parallelisation.</td>
</tr>
<tr>
<td>(Deelman et al. 2005)</td>
<td></td>
</tr>
<tr>
<td>Pegasus</td>
<td>Framework for scientific workflows. It allow users to set up and run multi-step computations, for example retrieve data from a database, reformat the data, and run an analysis. It allows for applications that run in a number of different environments including desktops, clusters, grids, and clouds.</td>
</tr>
<tr>
<td>(The Invisible Modelling Environment)</td>
<td>A model framework for developing, testing, linking, and calibrating environmental simulation models. End-user applications with user interfaces can be developed using TIME. The framework was developed by CSIRO Land and Water mainly for water quantity and quality models. TIME includes base data types commonly used in hydrologic modelling such as time series, grids, polygons, and networks, and visualisation.</td>
</tr>
</tbody>
</table>

5 http://csdms.colorado.edu/wiki/Main_Page
6 http://www.javaforge.com/project/oms
7 http://www.openmi.org/
8 http://www.cerfacs.fr/globc/PALM_WEB/
2.5.4 Selection of framework for full testing and evaluation

The assessment of shortlisted frameworks against criteria showed that many of the frameworks provide many of the desired features, but most had some shortcomings and none was clearly superior. Considering that there is sufficient promise in the existing frameworks, the project team considered that it was appropriate to continue to a deeper evaluation of a small number of frameworks by hands-on testing.

The team considered it most appropriate to focus on the most promising frameworks first, rather than trialling all the shortlisted frameworks. The following frameworks were not considered suitable for further testing for the following reasons:

- CSDMS and OpenPalm are mainly intended for running complex large-scale models on high-performance computers. There is not an immediate demand for such facilities for freshwater models used in New Zealand. Provision of high-performance capabilities comes at the cost of difficulty of programming models for the interface and difficulties in making the models accessible to end-users who just wish to run the assemblage of models with a user interface.

- There are concerns about the longevity of TIME. Development on the framework seems to have ceased, and the current funding programmes have finished. The championing organisation, CSIRO, has been recently more committed to developing sophisticated end-user applications such as Source. This situation could change in the future, but at present adoption of TIME would be risky.

- Pegasus is attractive in the sense that it provides workflow functionality, and hence might be able to provide provenance and auditing information better than other frameworks. However, Pegasus is mainly intended for large-scale scientific workflows, rather than for running small collections of freshwater models. It relies on models being set up as executable files and relies largely on file-based data transfer, so it is not very amenable to setting up interfaces for end-user assemblages.

- OpenMI was similar to OMS3 in capabilities and pedigree, having been developed primarily to link existing hydrologic and hydraulic models or model components. However at the time of the evaluation, the status of OpenMI was in question because the projects that originally funded development had ended and support was transitioning to an unfunded contribution basis. Further, at the time there was little software available to make use of the OpenMI standard. However since this phase of the project, there have been additional developments regarding OpenMI that are discussed further in Section 4.2.1.

Based on all the considerations above, the project team decided to only test OMS3. OMS3 is a framework for environmental model development developed by the US Department of Agriculture and Colorado State University (David et al. 2013). OMS3 is programmed in Java and was intentionally designed to facilitate transfer of technology from research to end-users. The intent of the frameworks is to provide
“a consistent and efficient way to create science components, build, calibrate, and evaluate models and adjust them as science advances, in addition to re-purposing models for emerging customer requirements.” (David et al. 2010).

A key aspect of OMS3 was the desire to foster both new model development and re-use and integration of existing legacy models through a minimalistic annotation approach, as well as allow for the possibility to scale-up modelling efforts with minimal technical overhead.

From the standpoint of the project, OMS3 met most of the key and high importance attributes identified at the end-user workshop including:

- Is open source.
- Receives substantial investment from the US Department of Agriculture as part of the large Conservation Delivery Streamlining Initiative (CDSI) programme and therefore has a high likelihood for continuity and longevity.
- Is supported by integrated development environments (IDE) for developing testing and running interoperable models.
- Has its own integrated graphical user interface for ease of use.
- Has a considerable and active international user community.
- The primary developer was open and responsive to questions and queries.
- Is platform independent, as it is based on the Java programming language.
3 Object Modelling System v3 (OMS3) Testing

3.1 Approach and rationale for testing

Our approach for testing was centred on assessing OMS3 in relation to key requirements of frameworks including: 1) auditability; 2) ease of use including ease of implementing legacy models of varying complexity; 3) support for different types of modelling; 4) capacity for complex simulations; 5) support model calibration and sensitivity analysis; 6) likely longevity; 7) cross-platform capability; and 8) openness and use of standards.

We performed hands-on testing of OMS3 using New Zealand data and models, to get direct experience of how difficult it is to use the framework, to obtain a deeper understanding of the framework, and to test whether the framework meets specific key requirements.

We did not have the resources for hands-on testing of all aspects of the framework relating to the requirements (for example, setting up a complex spatio-temporal model with a user interface), so some aspects have been assessed based on reading OMS3 documentation or from discussions with the OMS3 developers.

The hands-on testing centred on implementing three models both individually and exploring coupling them to create simple integrated models:

1. WATYIELD. This is a simple point-scale time-stepping water balance model developed by Landcare Research. This model was used as a centre for the following questions:
   - How difficult is it to set up a legacy model written in a language other than Java?
   - Can the framework be used to work with time-stepping models operating at a point scale?
   - Can time-series results be plotted?
   - Can a time-series model component be linked to climate time-series input data obtained via an SOS (Sensor Observation Service) web service?
   - Can a spatial dynamic model with a physics component be set up in OMS3?
   - Can spatial data be imported using web-based services to be used with a spatial model?

2. OVERSEER and SPARROW

OVERSEER\(^9\) is a model for predicting nutrient leaching from farms. SPARROW\(^10\) estimates nutrient loadings in surface waters based on statistical modelling of relationships between nutrient leaching from different land uses and observed nutrient loadings. A simple version of OVERSEER is coupled to the SPARROW

---

\(^9\) OVERSEER.org.nz
\(^10\) http://water.usgs.gov/nawqa/sparrow/
component in the CLUES catchment model (Semadeni-Davies et al. 2011). We used OVERSEER and the SPARROW code to examine the following questions:

- Can a simple model provided as a “black box” executable model (i.e., only the compiled version of the model is available) be implemented as an OMS3 component via “wrapping” (i.e., new software code written in the OMS3 framework to call the executable model and pass input and output data appropriately)? This test was attempted with a) the OVERSEER Version 5 dynamic link library (= dll or executable) used in CLUES, which reads and outputs a small set of parameters stored in memory and b) the OVERSEER Version 6.0 dll which reads and outputs XML-based files.

- Can a simple user interface be developed as an OMS3 component to manipulate an XML-based input file and then run a model (OVERSEER6) and display the results?

- Can results of two runs be compared with a simple plot?

- Can OMS3 be used to run a model that (OVERSEER) that has been set up as a web service?

- Can an existing model written in the Fortran language (SPARROW predictive model from CLUES) be set up as an OMS3 component without translating the code into the Java language?

- Can two different models be linked together (OVERSEER linked to SPARROW)?

3. APSIM

APSIM (Keating et al. 2003) is a simulation framework that uses the Common Modelling Protocol (Moore et al. 2007) to link a large range of dynamic simulation models and enables process-based simulation of a large range of agricultural systems. We tested whether existing time-stepping soil modelling components from APSIM developed in the Microsoft .NET software framework could be set up as OMS3 components.

These tests, and the degree of success with them, are discussed in more detail in Appendix B. In short, all the tests were successful except for the APSIM test, although work-arounds to accommodate models provided as dll’s were required. The results are also mentioned in the next section which presents our findings about the suitability of OMS3.

3.2 Results of assessment

In this section we provide results of our assessment of the suitability of OMS3 in relation to requirements identified by end-users and the project team, in a Question and Answer (Q&A) format. The assessment is informed by hands-on testing, documentation of OMS3, and discussions with the developers, as described in Section 3.1.
3.2.1 Does the framework allow for auditability?

A requirement that arose from the workshop, especially from the indirect users, was that an interoperable modelling framework must allow for repeatability and auditability of model runs, provision of metadata, and information on model assumptions.

These needs would be best met by a framework that is specifically targeted at scientific workflows, such as those used in operational flood forecasting systems. OMS3 is not targeted specifically at scientific workflows, but it has several features which help in terms of auditability.

First, OMS3 provides several meta-data annotation types (such as version numbers) that can be included in code for the individual components or parameter files, and a facility to provide reporting on the components based on these annotations. Second, OMS3 uses scripts to define the assemblage of components, how they are run, and values of parameters (and file names), to allow for repeatability and documentation of runs. In this sense the simulations are self-documenting. The system does not provide specifically for recording the provenance of files provided by external parties, though. Third, simulation results can be logged. Finally, all the resources for a simulation can be packaged as a simulation executable file, including a digital signature.

It would be very unusual for a modelling framework to provide specifically for documentation of model assumptions, or assumptions made during a particular application of a model. Information about the formulation and assumption of a model would generally be left to the base documentation of the models or model components, while information about the specific application (for example, reasons for choosing a certain discretisation of stream reaches in a particular study area) would need to be documented separately.

3.2.2 How easy is it to use the framework?

Our hands-on testing gave us direct experience with setting up model components of various types and complexity and linking them. Generally, OMS3 was found to be versatile and straightforward to work with, although for complex models and chained simulations the assistance of a systems developer would be required.

Making a model OMS3-compliant is achieved by inserting special OMS3 annotations into the computer code (usually Java) for the component. The components along with their linkages, specification of input and parameter files, and post-processing steps are specified in a script file. The script uses various high-level commands (which are called Domain Specific Language elements, or DLE, in OMS3) and can be edited in a console graphical user interface. The simulation can be run from the interface, or can be called externally from other applications.

Ease of setting up components

The ease or difficulty with which a model can be made OMS3 compliant is model-specific and depends on a number of factors including the complexity of the model, the programming environment, and the quality of the original software code. Monolithic models written in procedural languages such as Fortran and C require more re-engineering than models written in object-oriented languages such as C++ or Java, which can be more easily divided into separate meaningful components.
We found that some simple subroutines written in Fortran were easy to set up as components because OMS3 already supported that language. This involved entering annotations into the code, which are lines of code to instruct the framework how to interact with the component, e.g., what are the input parameter variables.

More complex legacy Fortran models could require considerably more effort to transform them into OMS3 components. Typically, large legacy Fortran programmes involve a number of subroutines and data items in shared memory. Converting these into separate OMS3 components would require disentangling the subroutines and data. This requirement is not specific to OMS3, but would apply for any component-based framework. Actually, OMS3 does allow for code in Fortran, so translation to Java is not necessary, but restructuring and re-coding would still be required to develop a set of components from the legacy code. We did not attempt to set up a complex model in this way. The OMS3 developers have recently set up AgroEcoSystem-Watershed (AgES-W) (David et al. 2013). AgES-W a watershed model programmed in Java and is derived from complex Fortran programmes such as the catchment model Soil Water Assessment Tool (SWAT).

We found it difficult to set up dll’s (dynamically linked libraries) derived from Delphi or .NET languages as OMS3 components. We were eventually successful for OVERSEER dll’s, but not with components from the APSIM modelling framework. The same approached used for OVERSEER Version 6 could have been used but we did not have a seasoned .NET developer readily available in the project team to handle co-dependent dll’s in APSIM. For the WATYIELD model, where the original code was in the VB6 language and the user interface was tightly coupled with the computational engine, we found it easier to re-write the model in Java, and this was fairly straightforward.

It was straightforward to set up a simple Java model as a component to loop through a number of OVERSEER calculations.

**Setting up components to access data and models through web services**

We were successful in setting up and linking components to obtain climate and soils data that is compliant with OGC (Open Geospatial Consortium) standards through a web interface, and with running a version of OVERSEER as a web service. OMS3 does not specifically include this capability, but it was fairly straightforward for systems development specialists to write these components. The process involved writing Java wrapping code to handle the inputs and outputs of the services (typically XML documents) and deal with the service interfaces (e.g., SOAP, JASON).

**Linking model components**

In OMS3, a simulation file is written in the Java-based Groovy scripting language to link the different model components and set model parameters. This requires basic programming skills, but little specialist knowledge of Groovy itself, as only a small set of scripting commands and used and they are well documented in the OMS3 examples. The OMS3 console is a user interface that displays the content of simulations files and runs them. The user can edit the simulation files and rerun them as needed. We tested this functionality, by chaining components (for example, obtaining spatial data from a web service and feeding this to WATYIELD) and also setting up looping configurations. We did not test complex...
spatio-temporal models, although some complex models such as AgES-W have been set up by the developer team.

OMS3 does not have a graphical ‘canvas’ for selecting model components as graphical icons, establishing their interactions through drawing linkages, and selecting graphical icons or links to edit their properties. Other frameworks do use a canvas, as did early incarnations of OMS3. For OMS3, the developers chose to use a scripting approach instead, to give greater flexibility, but this does not preclude developing an icon-based interface in the future. The programmers involved in FIFM did not find it difficult to use the scripting interface for our simple examples, but consider that a canvas would be a useful addition for depicting and modifying the system.

Reliability

We found OMS3 to be very stable and reliable (the user interface did not crash).

The system has capabilities for setting up tests for model components, and for monitoring and stepping through a simulation, which help with debugging.

Setting up graphical user interfaces

We set up a simple user form to enter a model parameter for OVERSEER. This was essentially written from scratch. For more complex models, it could be a large undertaking to retain the original functionality of the user-interface, because the effort put into developing user interfaces is often larger than the effort required for the computational engine. Additionally, for legacy models, the user interface is often intertwined with the computational engine, so that large parts of the code would need to be re-written (or discarded) to provide full functionality. One work-around is to set up the model with the parameters from the ‘native’ model, and then to use the framework primarily to link the model and maybe modify a restricted set of parameters. This is a general problem for modelling frameworks, not just OMS3.

Visualisation of output

OMS3 has some capabilities for displaying time series, flow-duration plots, and scatter plots via its post-analysis components. The outputs appear as windows in the console. Other than that, there is little in the way of capabilities for visualising model outputs, such as maps or animations. We made a simple component to display multiple time-series, using third-party open-source libraries. Some spatial analysis libraries (see JGrass tools later) have been made compliant with OMS3. Similarly, mapping could be provided by setting up OMS3 components to access routines in the Java uDig spatial analysis and mapping library.

Documentation and training

OMS3 has several user tutorial-style online manuals, and examples are available. Also, a set of training presentations is available from a recent training symposium held in Italy. A detailed reference handbook has also been prepared recently, although this is incomplete, so that it is difficult to obtain information on some of the recently-developed or more advanced aspects of the system. These resources are available via the OMS3 wiki. They were not available in an organised form when we started testing; others learning the system should find these tutorials very useful. The source code is also available for those wishing to inspect
it directly. The developers were willing to provide advice at the systems developer level and to provide more general information. For example, they made a presentation at the second workshop for this FIFM project.

OMS does not seem to have a current infrastructure for registering, discovering and accessing existing components. A component repository (http://www.javaforge.com/project/omslib) was established at one time, but the entries seem to relate to an earlier version of OMS and are from about 2009. Within the institutions responsible for developing the framework, there is probably better access and exchange of components, and this probably reflects that OMS3 was set up primarily to serve US Department of Agriculture needs. We expect that this situation will change as the platform matures. Also, OMS3 does not have a formal way of specifying the meaning of variables (via a data dictionary, conventions, or an ontology) beyond a few simple optional annotations such as unit types, so documentation of these aspects is particularly important to assist with connecting components in a meaningful way.

3.2.3 Does the framework support static, temporal, spatial and spatio-temporal models?

Chaining of models

We were able to chain together simple model components such as the code to run OVERSEER repeatedly then run SPARROW catchment routing, as explained in Appendix B. The chaining order and which data is passed from one model to the other are specified in the OMS simulation file.

Time-stepping

OMS3 is not designed particularly to manage time or support time-stepping models. Time-stepping has to be done by writing code to loop through model components, or in a specific component set up for managing loops. Asynchronous models (different time-steps in a system) are not specifically supported by OMS3, unlike some other frameworks such as OpenMI. OMS3 does have the facility to graph time series, and dates are supported as a standard data type in ASCII input files.

Spatial models and data

OMS3 is not designed specifically to support spatial calculations. In contrast, some other frameworks are oriented to supporting grid-based numeric calculations, or for supporting hydrological calculations with grid and network structures. OMS3 does not support spatial analysis or transformation, standard spatial data types such as grids or shapefiles, nor is it aware of co-ordinate systems and spatial constructs such as spatial extent or resolution.

Instead, spatial models can be constructed by looping through spatial elements, or at a higher level incorporating spatial computations within a component. The work of importing and manipulating spatial data can be offloaded to components built for that purpose. The JGrassTools project (https://code.google.com/p/jgrasstools/) for example provides a set of OMS3-based components, which are mostly targeted at geomorphological and hydrological spatial processing requirements (e.g., slope, curvature, flow direction), but also include standard GIS functionality (e.g., resampling, interpolation, map calculation). JGrassTools in turn builds on GeoTools (http://www.geotools.org/), an open source Java GIS toolkit, and
leverages classes for spatial data handling (e.g., GridCoverage2D) to implement OMS3-based processing components.

We successfully set up a simple spatial version of WATYIELD, which looped through all cells in a grid and called a water balance model for each cell, as described in Appendix A. We did not attempt to run a spatio-temporal model, but there are examples of such models being developed successfully in OMS3 see AgES-W and the J2K-S distributed hydrology model (Ascough II et al. 2010).

In the case of WATYIELD, we used the Geospatial Data Abstraction Library (GDAL) library to access spatial data. GDAL provides support for a variety of common input formats and our testing successfully accessed ESRI ASCII grids, ESRI shapefiles and Geography Mark-up Language (GML) documents. The data were stored and accessed on a locally accessible file-system.

We also considered how spatial data could be accessed through OGC web services, although we did not actually implement this. The request/response standards for OGC web services are generic (in that they are intended to be universal, not domain or application specific) and formally documented. With this necessary behaviour defined already, a set of OGC service modules could be implemented in OMS3 – either by formally adopting an existing API (for example GeoTools) or writing a new one.

OMS3 does not include core capabilities for displaying spatial data such as grids or maps. Hence, specific modules to achieve this would have to be written, probably by accessing mapping libraries as described above for GDAL and uDig.

3.2.4 Can OMS3 handle complex, high-performance simulation tasks?

We have not done any testing with complex models. Some of the available examples are quite complex, especially when temporal stepping is done within a simulation script rather than internally to the component.

As pointed out previously, OMS3 is multithreaded, allowing for independent components to be run at the same time, holding the results in memory until they are needed. This is managed by the software itself, without the modeller’s intervention. In contrast, OpenMI does not have such capabilities.

OMS3 is not set up for accessing high-performance computing facilities such as supercomputers, although work is underway on running OMS3 models on cloud computing platforms such as amazon EC2 (David et al. 2013). Other frameworks, such as the CSDMS Modelling Tool (http://csdms.colorado.edu/wiki/CMT_information), are more targeted to large-scale earth systems modelling.

3.2.5 Can sensitivity analysis and calibration be done?

OMS3 has functionality for calibration, but we did not test this. As part of its core, OMS3 can produce model performance measures (e.g., Nash-Sutcliff index, Root Mean Square Error), statistical summaries, and graphs. Furthermore, some calibration (parameter optimisation) routines have been linked to OMS3, in a utility called Luca which is accessible through the OMS3 console.
Multiple models can be run to obtain Ensemble Streamflow Predictions using an ESP simulation type. This uses a range of meteorological inputs (derived from historical data) to derive a range of outputs. These features could probably be extended to derive predictions from a range of weather forecasts or rainfall-runoff models.

3.2.6 Likely longevity

OMS3 is a collaborative project involving the U.S. Department of Agriculture. Development of OMS3 commenced in 2001 and has continued since then. Support for the framework core is provided by the ‘OMS Laboratory’ at Colorado State University, with a team of expert systems developers. A discussion forum and bug tracker are available, and the comprehensive wiki serves as a central information source. Also, within the large USDA Conservation Delivery Streamlining Initiative (CDSI)11, OMS3 is seen as a key “glue” for helping bring together a broad suite of USDA models. This augers well for longevity of OMS3.

There is little evidence of OMS3 uptake outside US government agencies, except that the University of Trento has close involvement with OMS3. OMS3 has not been proposed for acceptance by a standards body. This may hinder its uptake outside US government circles, and consequently its longevity.

3.2.7 Is OMS3 cross-platform capable?

The OMS3 framework is written in Java and Groovy (a Java based scripting language) hence it can be run on any platform (LINUX, Windows, MacOS). C, C++ and Fortran executables have to be adapted and recompiled for the specific platform. There are likely to be difficulties using .NET-based components on non-Windows platforms, because .NET is essentially designed for Windows.

The model components essentially need to be run on the same machine, although they could be on a different machine or platform if a way to communicate between models is created, such as web services or other networking protocols. We demonstrated the use of web services for running OVERSEER as a web service, and the USGS has run an OMS3 soil conservation model, RUSLE2, as a web service from a user interface that is run on a tablet (David et al. 2013). Communicating via the web will have implications for the speed of the overall system, but we have not quantified this for OMS3.

3.2.8 Openness and use of Standards

The OMS3 code is open source, and managed with a code repository. There is currently no open repository of model components.

OMS3 supports basic data types such as integers and real numbers and dates. It also has an annotated CSV file format for storing and retrieving tabular and time series data.

OMS3 does not inherently have the capability to read standard types of time series or spatial data (for example, GIS shapefiles) nor does it support spatial data or time series standards (e.g., Open Geospatial Consortium OGC standards such as SOS, WFS, and WaterML). A separate utility is needed to work with files provided through such standards. We set up

some web services to achieve this, and also set up some standard GIS libraries to convert formats.

### 3.3 OMS3 testing summary

In terms of auditability, OMS3 has good features for capturing and reporting metadata and repeatable assemblies of components, although we did not test this aspect directly. These features are not as advanced as might be found in a formal workflow framework, though. While OMS3 does not document modelling assumptions or concepts, neither do other frameworks.

We found it fairly straightforward to write new model components for OMS3 in the Java language, which is the base language used by OMS3. This does require programming skills in Java. Entering appropriate instructions into the code (annotations) to make the model into OMS3 components was straightforward.

For some models, especially simple models where the user interface was closely coupled with the calculation engine, we decided to re-write the model in Java rather than wrap the original model as an OMS3 component. This provides an opportunity for separation of the calculation engine from the user interface and visualisation aspects, clearer partitioning of the model engine into components, and general improvements to the code. Transferring the model to Java could be a considerable undertaking for complex models.

OMS3 does support the Fortran language, enabling transfer of legacy models written in Fortran. A feature of OMS3 that is helpful in this regard is that the annotations do not interfere with the normal operation of the code, and the framework does not force complex structures or methods upon the original code. However, considerable work could be required to set up complex legacy Fortran models as OMS3 components, because many of those models were not developed in a component-based programming style.

Other models can be retained in their original form (for example a Microsoft Windows executable file) by writing Java code to link to and call the model, that is, wrap the model as an OMS3 component. We found it more difficult to work with models provided as dll’s. In particular, complex dll’s written using Microsoft .NET are difficult to set up as OMS3 components. Essentially, we consider that it would be impracticable to set up complex .NET-based dll’s as OMS3 components.

It was straightforward to chain model components together by writing scripts in the OMS3 console, although this does require programming expertise. OMS3 does not have a drag-and-drop canvas-style visual interface, and we think this would be a useful addition to make the system more accessible to non-programmers and to communicate the arrangement of model components and their linkages. OMS3 does not have specific control structures for looping through spatial or temporal components – that is left up to the programmer to implement through scripts or within the components. This could become unwieldy for complex assemblages of models.

Documentation and tutorials and a range of examples are available, although not up-to-date, and training is held from time to time. The community of users is fairly small, with little evidence of activity on newsgroups. The developers were inclusive and very willing to interact, though.
OMS3 has largely been set up for running on a single computer rather than high-performance computers, although the developers are working on this. We were able to set up a component to run a model as a web service using basic web transfer protocols (i.e., hypertext transfer protocol or http), which may serve as a means for accessing remote computing power. However such capabilities are not currently part of the core functionality of OMS3.

In terms of user interfaces, we were able to develop a simple user interface for entering parameters. This user interface needed to be developed essentially from scratch. For more complex models, setting up a user interface would be a large undertaking, and OMS3 does not really provide specifically for developing such interfaces. However, the user interface is often a large and critical component of the development of production-level scientific software. This is a difficulty with other modelling frameworks too, where the emphasis seems to be on developing the data access and computational engine components, rather than the user interface.

OMS3 provides rudimentary functionality for visualising time series and scatter plots, and for summarising output variables. Other more sophisticated visualisation such as tabulating results, visualising grid outputs, or mapping is not provided specifically by OMS3. Hence, the user must develop these components, which would usually take place by wrapping third-party libraries. There does seem to be a need for providing some more advanced visualisation facilities, perhaps through a shared component library. The wrapping of geospatial manipulation libraries such as already done with JGrassTools, serves as a good example. Over time, provided there is sufficient development of a community of OMS3 users and the provision of a structured component repository, we expect more components of this nature to become available.

OMS3 only supports basic data types. It does not support arrays, compound data types, networks, polygons, or rasters/ grids, and there is only limited provision for time-series. Similarly, OMS3 does not support common standards such as shapefiles, NetCDF files, or OGC standards. The use of basic data types does provide some generality and frees users from conforming to specific structures imposed by a framework, but relies on the framework user to do much of the legwork to set up and use more complex data types or structures. This difficulty is shared by other frameworks such as OpenMI, but by contrast the FEWS framework and TIME provide data types and data adapters that are relevant and useful for freshwater modelling. It could be argued that Java itself provides sophisticated data structures, and that external libraries for working with geospatial data are already available, but we think that access to more complex data types and structures would provide a richer and more efficient framework better suited to freshwater modelling.
4 Discussion and next steps

Section 3 described our assessment of OMS3 in relation to a number of requirements that had been identified by end-users and the project team. This assessment consisted of hands-on testing with a range of models, reading of documentation, discussions with the developers, and attending a specialised training course. Here we discuss the overall outcome of this testing, and consider the next steps.

4.1 Summary of strengths and weaknesses of OMS3

The OMS3 framework was found to meet many of the requirements that were identified by end users. There were facilities to document model assemblages and metadata, although frameworks designed for scientific workflows are more complete than OMS3 in this regard.

It is fairly straightforward to set up and link models in OMS3 if the models are written or re-written in Java or are well-structure Fortran code. We were able to access a model as a web service, and we were able to access third party libraries written in Java by writing wrappers to access them. However, we had difficulty using models provided as dll’s, especially ones written in Delphi or Microsoft .NET languages.

OMS3 is supported by a large government programme with very good software engineering support, it is open, and a community of users is developing. The documentation is generally good, although it needs refreshing. There is no current community repository of components, and this is a shortcoming because component re-use and sharing should be a strength of interoperable modelling systems. OpenMI is in a similar position, but other frameworks such as TIME and FEWS have a larger library. It is fairly early days in terms of the propagation of OMS3 beyond the original development group, though, and we expect that documentation and the component library will increase over time.

OMS3 does not provide specifications or guidelines for development of user interfaces. Hence, to bring a model with a complex interface into the OMS3 and set it up as a polished end-product, the user interface would have to be re-written. Moreover, many legacy models intertwine the user interface with the computational engine, such that adapting the original model could be required to bring it into the OMS3 system would require significant effort. These difficulties are shared by most frameworks, and are a barrier to their adoption. An advantage of OMS3 is that it does “lightweight” and flexible and being based on the Java software language would allow for the development of complex user interfaces if desired.

OMS3 does not provide data structures suited specifically for hydrological modelling. For example, it does not provide for drainage networks. This avoids enforcing data structures on model components and provides flexibility, but in our opinion it would be desirable to have more complex structures available to provide for a more efficient and powerful system for freshwater modelling. Similarly, OMS3 does not provide support for standard data formats such as shapefiles or NetCDF files out of the box; such support could be provided by suitable components developed in the future or through third-party libraries. Some progress has been made in this regard with provision of GIS processing tools, but the number of such utility components currently available is small. This situation may well improve in the future. The core OpenMI standard has similar limitations, although it does allow for lists of spatial elements in ‘element sets’, and user-defined extensions to the data types can be introduced.
In contrast FEWS and TIME support more data structures and standards in the core implementation.

A further difficulty is that OMS3 has limited functionality for visualising results, limited to plotting of time series. While components can be built for visualisation, and there are many visualisation libraries for Java, the limited out-of-the-box capabilities are a barrier to rapid prototyping or the user of the system by modellers not familiar with the range of Java libraries.

OMS3 has no in-built capabilities for managing complex timing or spatial elements. OMS3 is aware of inter-dependencies, such as the need to wait for another component to run first if the output of that component is required as inputs. In general the user must manage time and spatial looping themselves, either within the component or within the scripting for the assemblage of components.

While we were able to set up simple spatial and temporal models, we did not attempt complex model configurations or computationally-demanding configurations, and we did not find information on how well OMS3 performs for computationally-demanding models. We know that for other frameworks, running complex interacting models in a framework imposes a large computational burden.

We expect that developing and deploying complex OMS3 will require considerable programming effort and expertise, for example, to re-write calculation components, set up user interfaces, link to visualisation and data translation utilities, or to set up complex spatial and temporal configurations.

4.2 Alternatives and trends

In the evaluation of OMS3, we found some weakness in relation to our requirements. Hence it is appropriate to re-consider some of the alternative shortlisted frameworks identified in our initial review (Elliott et al. 2012), taking into account what we have learned about freshwater interoperable modelling frameworks in general and OMS3 in particular. We also re-examine some of the recent trends in the area of interoperability frameworks.

4.2.1 Reconsideration of other existing frameworks

One of the key alternatives that we identified in our earlier work was OpenMI. This framework was developed from European initiatives to integrate water models – especially hydrologic and hydrodynamic models. We initially decided not to test OpenMI because there was uncertainty about its future uptake and software tools to implement the standard, and due to resource limitations. Knapen et al. (2013) discuss strengths and weaknesses of OpenMI.

An important development since we undertook framework screening was the ratification of OpenMI as a standard by the Open Geospatial Consortium (OGC). Potentially this could lead to wider uptake of OpenMI, especially considering the rapidly-increasing uptake of OGC standards for time series and geospatial data (including within New Zealand). Also, there are several recent papers on integrating OpenMI with other frameworks or data sources (e.g., Castronova et al. 2013b). Despite these developments, we have some reservations about adopting OpenMI standards immediately as a pathway for interoperable modelling in New Zealand, for the following reasons:
The latest version, OpenMI 2.0, is incompatible with 1.4. Anecdotally, there is a large overhead in shifting between versions, and consequently we do not see much evidence yet of components developed for OpenMI 2.0, despite this version being available since 2010 (for example, the list of models on the OpenMI website are all for version 1.4).

While establishment of a standard (for example, the set of rules that a model must obey) is a positive step, implementing the standard will require the development of suitable software infrastructure. The OpenMI technical committee does not plan to introduce an implementation of the OpenMI 2.0 standard (whereas they did provide a development kit for 1.4). HR Wallingford have developed a .NET-based implementation\(^\text{12}\) (precluding Java models), but the degree of uptake is unclear. Recently a wrapper for implementing OpenMI 1.4 within Matlab and Python environments has been developed (Bulatewicz et al. 2013), but we are unsure whether this will be translated to 2.0. CSDMS, which adopted OpenMI 1.4 for model interoperability on high-performance computing, does not yet seem to have moved to 2.0. Associated with this, there is no software infrastructure for developing user interfaces or visualisation within version 2.0 (and even for version 1.4, this was not a strong feature).

OpenMI can impose a large computational overhead. For example, a 7x increase in computational time compared with the native models was reported in one case (Shrestha et al. 2013), although another case, when the model run time itself was limiting, OpenMI imposed little additional overhead (Knapen et al. 2013). It is not clear whether the overhead is inherently associated with the standard, or whether the implementation is the cause of the overhead.

Current implementations are not amenable to multi-threaded or parallel calculations (Knapen et al. 2013).

OpenMI supports only a basic data model in the core standard, so does not offer advantages to OMS3 in this regard. While low-level data models give flexibility, we note that in one application in the Seamless-IF project, it was necessary to extend the basic data model (van Ittersum et al. 2008).

As with OMS3, OpenMI focuses on the mechanics of interoperability and does not provide for semantic or conceptual integration.

There is no large funding for version 2.0; current work is largely on a voluntary or piecemeal basis.

Many of these concerns have been discussed in an informative blog\(^\text{13}\).

Another framework that was initially attractive was the CSIRO framework TIME (Rahman et al. 2004). We did not pursue this, as we were uncertain of the future support. Funding for this framework through eWater has essentially ended; current funding is directed more to developing the Source Catchments model\(^\text{14}\) and developing workflow approaches for routine

---


flood predictions. Source Catchments requires buy-in to access the code, which is contrary to the requirements of end-users for this project.

The earth systems modelling community continues to use and develop systems for large-scale climate and hydrologic computations, but these are considered to be more oriented towards the research community. A leading contender in this space is the Earth Systems Modelling Framework ESMF\textsuperscript{15}.

Workflow frameworks were another attractive option, especially since interoperable modelling is largely a workflow process, and supporting software from the earth sciences is fairly well developed. CSIRO have continued to investigate workflows for meteorology-flood prediction (pers. comm. David Lemon, CSIRO Land and Water). It seems that such workflows in the freshwater domain are largely targeted at routine workflows such as operational flood forecasting, so we do not see these developments immediately addressing the broader range of user needs identified in the workshops (such as being able to develop and prototype new assemblages of models). Another common use for workflows is in large-scale collaborative scientific research exercises (especially in biological sciences and physics), which is not particularly relevant to user needs. Nevertheless, workflow systems continue to be developed and extended, and hybrid systems may well emerge to meet our freshwater modelling needs (Granell et al. 2013b).

\subsection*{Models as a web service}

In New Zealand, there has been a surge in availability of environmental data through standardised web services, in line with spatial data initiatives around the world\textsuperscript{16,17,18}. Examples in the land and water domain include the NIWA Information Services Stack (NISS) and Landcare LRIS portal and associated interface\textsuperscript{19}, and recently the Smart Aquifers initiative (Kmoch et al. 2012), as well as the New Zealand Spatial data Infrastructure (NZSDI) by the New Zealand Spatial Data Office. In this project we successfully linked to such data sources through standard web services wrapped as OMS3 components. For example, we developed components to retrieve climate records exposed as an SOS service, and we retrieved vector and raster input data sets of land cover, rainfall, and potential evaporation from Landcare Research OGC-compliant web services.

This success with data provisioning raises the prospect of whether models can follow a similar standardised web services delivery approach. Indeed, Maué et al. (2011) speculated that Sensor Observation Services (SOS) may be used to expose ‘data’ generated by models, and Goodall et al. (2011) proposed a design for a web service standard for achieving this. In our testing of OMS3, we accessed OVERSEER through an http-based web service request. In our initial framework evaluation report (Elliott et al. 2012), we explored the prospect of linking models provided as OGC services (and also alternative World Wide Web Consortium semantic web approaches), as illustrated in Figure 3.

There has been a recent flurry of papers describing various architectures and visions for web services in the water modelling domain (Goodall et al. 2011; Fritzinger et al. 2012; Castronova et al. 2013a; Granell et al. 2013a; Laniak et al. 2013a; Nativi et al. 2013). Such
systems are largely in the proposal or prototype stages of development, but this is a promising and highly active area of development. Laniak et al. (2013a), based on a series of workshops to develop a roadmap for Integrated Environmental Modelling (IEM), argued that “a key goal of the IEM community must be to more effectively leverage the Web for publication, discovery, access, and integration of IEM information and software in order to achieve the ambitious goals set by the IEM community”. In a review of frameworks, Granell et al. (2013b) noted that web services approaches were more mature for scientific workflow frameworks (such as Kepler) in comparison with component-based integration systems (such as OpenMI). These workflow systems commonly make use of Web Services Description Language (WSDL), which is a common system for describing web services on the internet and is endorsed by the Worldwide Web Consortium (W3C). Granell et al. also noted that it is less common for Open Geospatial Consortium (OGC) approaches to web processing services to be used by integration frameworks.

Various demonstrations of web-based model linking systems have been made. Even in our work, we provided a link to an OVERSEER model service through OMS3, which was done on a fairly ad-hoc basis using standard Java and http calls. Saint and Murphy (2010) discussed a web-based link between a climate model run on a high-performance computer with ESMF and a catchment model run on a PC. Castronova et al. (2013a) described an approach for setting up models as Web Processing Services accessed as OpenMI components, and demonstrated this with an application of HydroModeler, a model coupling tool that uses OpenMI. They found that it was necessary to introduce modifications to their chosen standards-based approach (Web Processing Services, Restful middleware) to enable time-stepping and storage of model state. They found that a simple rainfall-model run in this way took 20 times longer to run as the original model, and that further development is required to develop more elegant and efficient web-based interoperable models.

![Diagram](image)

**Figure 3:** Illustration of how OGC Web Processing Services, (WPS), catalogue services (SC/W) and Web Feature/Coverage services (WFS/WCS) could be used to mediate between model computation components and data sources.
The advantages of implementing models as web services are:

- Model providers can retain control of their models and maintain the models as necessary in whatever software and computing platform that is suitable, as long as they expose the standard web service. This would overcome the current .NET versus Java divide that causes problems for systems such as OMS3. This type of flexibility encourages uptake of the standard, because organisations can maintain and control the model (or data) and only need to provide the appropriate interface rather than transform the entire model or database into a standard format.
- There are existing strong governance infrastructures already in place (such as the OGC and W3C).
- Web services provides a mechanism for accessing remote computing resources, potentially providing access to increased and scalable computing power.
- The use of web services for models is compatible with current and evolving web-based approaches for discovering, providing and sharing data.

However, potential drawbacks include:

- There is a divide between OGC and W3C approaches – hopefully there will be some convergence or integration of approaches, and some recent approaches to resolving the dichotomy, which typically involve setting up OGC services as more general WSDL/OGC services, have been reviewed (Granell et al. 2013b).
- Difficulty running complex suites of closely-coupled models, due to more complex arrangements of service requests and related requirements for frequent communications and storage of state over time. Running models as web services may be better suited to loosely-coupled models and workflows.
- Performance penalties from transferring data over the web. This will be more severe for large data transfers and frequent model requests. Ideally, in the future there would be provision for local data storage and shared memory to enable faster running of models when needed.
- There is no software to provide modelling-orientated orchestration of distributed model components. While there is a range of software that can be used for implementing web services, software that packages these facilities in an environmental modelling context are in their infancy. One potential pathway is to add web services to existing framework such as OMS3 rather than starting with existing web-centric software.
- Current frameworks such as OpenMI define model components in a standardised way through input and output data items and standard operations such as 'initialise-execute-standardise', which are at a suitably level for general web services, but it is not clear how more elaborate API's that might arise in a more complete system targeted at the freshwater modelling domain would be set up as web services.
Overall, we see models-as-a-service as an attractive approach to pursue. However, it is fairly early days in this area, so that it would be premature to commit fully to this pathway. We discuss a staged approach to adopting such approaches in New Zealand in the next section.

4.2.3 Semantics and conceptual interoperability

OMS3 and many of the other modelling frameworks primarily address the mechanics of how to make models and data interoperate (technical integration) without addressing whether those links are meaningful (conceptual and semantic integration). There are some existing standards, ontologies and dictionaries for water data (e.g., the WaterML standard for hydrological time series\textsuperscript{20} and the CUASHI-HIS controlled vocabulary for hydrology data\textsuperscript{21}) but there is little in the way of dictionaries or ontologies for models or the modelling process in the water domain, (Elliott et al. 2012) (although some are emerging, see WRC below). Hence conceptual linkage between components is largely ad-hoc, leading not only to difficulties in discovering and linking data and model components, but also potentially to ‘integrons’ – strange combinations of models and data that fit together at a mechanical level but with a meaningless or even “beastly result” (Voinov and Shugart 2013).

Our work categorised models and data sources according to some standard metadata fields (https://teamwork.niwa.co.nz/display/IFM/), enabling potential links between models to be identified based on common input and output items. This work did not, however, ensure that models could or should actually be linked in any meaningful way. The FRAMES framework for environmental risk assessment (Whelan et al. 2014) also uses dictionaries (naming conventions) to ensure that data is exchanged in a meaningful and consistent way.

Recently, Elag and Goodall (2013) developed the Water Resources Component (WRC) ontology for models of water resource systems, which is a sophisticated ontology aimed at hydrology and can be used to find and classify models components and their suitability for linking. Such ontologies should allow for a more structured and standard way for identifying the characteristics of model components, to make model coupling more meaningful. It still remains to be seen whether this ontology is adopted by the modelling community.

4.3 End-user feedback from testing

In the webinar held in April 2013, which included 27 representatives from 15 organisations, we provided the audience with information on OMS3 and our testing of the framework. There was consensus that the technology looked promising, and that further testing was warranted. Some of the Indirect Users (see Section 2.1.1) pointed out that they were primarily interested in the end-product of the framework, namely integrated models. If the framework enabled the modellers to produce this output, then that was good, but they would need evidence that a polished end-product could be developed before committing their full support to that framework. This provided an important pointer for the next steps.

4.4 Further steps

End-user consultation assessment in this project has confirmed the strong perceived benefits of an interoperable modelling system, such as improving integrated models across freshwater domains, re-using model components, and making better use of the increased

\textsuperscript{20} http://www.opengeospatial.org/projects/groups/waterml2_0SWG
\textsuperscript{21} http://his.cuashi.org/mastercvdata.html
availability of environmental datasets and associated standards. Our trialling of OMS3 to date has demonstrated the feasibility of linking freshwater models in a variety of ways, and also of linking to environmental datasets.

Despite these positive signals and successful testing, there is still some reluctance to commit to OMS3, for the following reasons:

- It is difficult for end-users to assess the added value of the OMS3 framework without evidence of a polished end-product in the New Zealand context.
- Scientists and research institutions are wary of committing to frameworks in general and to a particular framework and associated modelling standards for several reasons including:
  - there is a risk of locking in on one framework at a time of uncertainty about the pre-eminent frameworks. Considerable effort is required to set up components, only to have to turn around and re-do some of the work if a different framework choice is made
  - current frameworks have some shortcomings so that development work is required
  - significant resourcing is required to adapt models and data to become framework components, build new user interfaces, maintain the components as the technologies evolve, adapt and extend frameworks to overcome their limitations, and contribute to international collaboration and governance.

In short, people generally like the idea of interoperability frameworks and have high aspirations for them, but they are understandably cautious regarding the level of commitment and resources required to develop the standards, build appropriate infrastructure, and develop appropriate institutional capacity needed to make the shared aspirations a reality. Both OMS3 and OpenMI demonstrate that despite large investments (e.g., several million or perhaps 10’s of millions of US dollars or Euros) these frameworks have limited uptake outside the originating institutions.

To address these questions, we propose following steps:

1. Develop a showcase integrated model built with OMS3, scaling up from our experience to date. The developers of frameworks OMS3 and TIME have used this approach to build support for those frameworks. This will demonstrate the capabilities of the framework to end-users, as well as provide a higher level of testing (for more complex and computationally-demanding models and more elaborate user interfaces).

2. Further exploration of using models as a web service in a staged process, initially by using web-based data sources and geo-processing utilities, then to more complete computational services as the technologies develop. This will provide early benefits (enhanced data provision to models) with longer-term benefits such as institutions being able to maintain control of their models and be least bound by language and platform dependencies.
3. Establish institutional and funding arrangements to co-ordinate efforts across and within institutions, build consensus on standards adoption, keep abreast of developments and remain part of international communities, assist with setting up models in the framework and provide education and training.

4.4.1 Proposed showcase integrated models

Based on our work to date, we propose the development of a national daily hydrological model built using OMS3. We propose that this could be an extension of the WATYIELD model accessing spatial layers and time series obtained through web services, but providing a user interface and mapping functionality. This would test the framework in terms of building larger spatio-temporal models with more complex user interactions, and provision of data via web services. It could also offer the potential to test aspects of multi-threaded or high-performance computing.

A second showcase would be related to water quality. We propose a system that would essentially reproduce several of the key aspects of CLUES, including using OVERSEER as a source term, linking with stream transport models, and linkage to receiving-water models for large catchments. It would also test linking to spatial land-use layers and performing geospatial manipulations, interacting with the user in a mapping context, and providing map-based outputs.

A further avenue, which is being discussed within NIWA and GNS, is the possibility of using OMS3 as a linking technology for models in the current Riskscape project, starting with a component to run the current landslide model as a web service.

We also note that the Ages-W application being developed by the USGS will further demonstrate the ability of polished products to be produced from OMS3.

We propose that the showcase applications will be built with OMS3. While this framework has some limitations and the technology is changing rapidly, this framework should serve at least as a stepping-stone to any other framework. As noted earlier, much of the work in integrating models in a framework is related to separating the calculation engine from the user interface, deciding conceptually how to link models, componentising the computational aspects, and modernising the code overall. In this sense, even if OMS3 does not eventually become the preferred framework of choice in New Zealand, development of the showcase model within OMS3 will yield generic benefits in terms of building cross-institutional capacity in interoperable modelling, similar to the benefits that the current project has yielded in terms of enhancing our shared understanding about the benefits and challenges of successfully developing and implementing an interoperable modelling framework.

4.4.2 Exploration of models as a web service

Our tests have demonstrated the feasibility of obtaining data for models from web services using OMS3, and the ability to run a simple model as a web service. Internationally, there is a flurry of activity and interest in development of systems to run environmental models as standardised web services (see Section 4.2.2). However, a software infrastructure to provide freshwater models as a service is not yet available and the cost of developing one would be high. With this in mind it is proposed that a staged approach be used.
The following steps are recommended:

1. Establish a common set of digital data formats and standards for model input and output. Much progress has been made in this area in New Zealand recently, in relation to adoption of OGC standards for delivery of climate and spatial data. We expect to adopt standards such as WaterML and GroundwaterML where an extensive community of practice has invested time and money in testing the standards through live deployments. We expect to extend this to compact binary formats such as netCDF, geodatabases, and HP5 to enable more efficient transfer of larger datasets, especially vector and gridded spatial data types.

2. Construct a set of OMS3 components to provide these data to models via web services such as SOS and WPS, and to provide model output in these forms via standardised web services.

3. Wrap selected computational components (such as those involved in the proposed showcase model) in standard web-service interfaces (such as W3C or WPS standards), then wrap these further as OMS3 components, and finally use OMS3 as the framework for linking and co-ordinating models.

4. Shift to more web-centric linking and co-ordination as technologies become available. This may ultimately consist of a hybrid system built on workflows and web services (Granell et al. 2013b).

### 4.4.3 Institutional and funding arrangements

As well as the specific staged activities identified above, there is a need for some baseline funding for an interoperability framework to co-ordinate efforts across and within institutions, build consensus on standards adoption, keep abreast of developments and remain part of international communities, assist with setting up models in the framework and provide education and training. For example, it would be desirable to become part of a proposed international moves to establishing cross-framework standards and protocols for integrated modelling (Laniak et al. 2013b). However, we propose to rely largely on international initiatives to fund development of core aspects of the framework, because considerable effort (for example, 5-10 FTE) are required to develop and maintain core framework software.

Ideally, the local activities would be funded from a dedicated MBIE or other central government funding stream, as other funding streams are largely committed, and individual programmes are usually more interested in developing data or models in their immediate domain of interest. An alternative is to build interoperability as a component of the National Science Challenges, or new Centre of Research Excellence funding.
5 Conclusion and recommendations

The project successfully:

- Identified requirements of a computer framework for interoperable freshwater modelling.
- Catalogued the range of freshwater-related data sources and models used in New Zealand, cataloguing their attributes and potential linkages.
- Reviewed a wide range of framework technologies, screening them against criteria, and selecting a particular framework for more detailed trialling.
- Tested and demonstrated the chosen framework with a carefully-constructed suite of tests.

We found that the selected framework, OMS3, met many of the criteria that had been established. For example, we were able to set up components of a variety of sources, link and run them, develop simple user interfaces, visualise the results, and access data and simple models provided as web services. Hence, OMS3 has been demonstrated to have many of the desired building blocks for a modelling framework for New Zealand.

However, users are not yet in a position to adopt OMS3 for several reasons:

- Ongoing funding is not available currently to support transferring models to the framework, maintaining the models and components, and contributing to international development communities.
- There are some weaknesses in the framework, including difficulty in setting up models written using .NET, there is no current publicly-available repository of components, documentation is patchy, user interfaces need to be constructed from scratch, there is no core support for many data structures commonly used in hydrology (such as networks), there is very little functionality for visualising results, and no core geospatial awareness. Many of these shortcomings could be addressed by building new components, especially ones using third-party libraries such as GDAL, and over time more components and utilities will become available, but at present the extra work required presents a barrier to adoption by a wide user base.
- The ‘indirect users’ have called for demonstrating of a polished showcase demonstration project, to demonstrate complex arrangements of model components and a polished user interface and visualisation.
- Recent adoption of OpenMI as an OGC standard for interoperability has created some confusion about the best pathway. We do not propose shifting to OpenMI at this stage, though.

Therefore, a short to medium term roadmap has been developed, including the following steps:
1. Develop one or more showcase integrated models in OMS3, such as a national-scale hydrological model or a linked leaching-catchment-lake model, with a polished user interface and visualisation.

2. Commence a staged series of tasks to develop data sources and model components through web services.

3. Develop and maintain institutional arrangements and funding to support model interoperability and to contribute to international development communities.

6 Acknowledgements

The authors would like to thank members of the project Steering Group and contributors to the two workshops for their valuable input to the project.

This project was funded under Ministry of Business Innovation and Employment contract C01X1003.
7 References


Saint, K., Murphy, S. End-to-End Workflows for Coupled Climate and Hydrological Modeling. *International Environmental Modelling and Software Society (iEMSs) 2010 International Congress on Environmental Modelling and Software*, Ottawa, Canada.


Appendix A  Table for initial assessment of shortlisted frameworks
This table is from (Elliott et al. 2012). Further details on candidate frameworks are on the project wiki [https://teamwork.niwa.co.nz/display/IFM](https://teamwork.niwa.co.nz/display/IFM). Note that since the table was prepared, we have learned more about the capabilities of various frameworks, especially OMS3 and OpenMI, but the table below has not been updated to take this additional knowledge into account.

<table>
<thead>
<tr>
<th>Importance</th>
<th>Category</th>
<th>Criterion</th>
<th>CSDMS</th>
<th>OMS3</th>
<th>OpenMI</th>
<th>OpenPalm</th>
<th>TIME</th>
<th>Pegasus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>Auditable</td>
<td>Previous model/data/parameter assemblages can stored and re-run</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Key</td>
<td>Dynamic</td>
<td>Supports dynamic models</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Key</td>
<td>Dynamic</td>
<td>Supports static models</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Key</td>
<td>Extensible</td>
<td>Can add/remove/substitute components, not a fixed assemblage</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Key</td>
<td>Open</td>
<td>Framework software is open-source</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Key</td>
<td>Open</td>
<td>Framework is available for anyone to use at low or nil cost</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Key</td>
<td>Open</td>
<td>Uses open standards for model interfaces</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Key</td>
<td>Open</td>
<td>Uses open standards for data interfaces</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Key</td>
<td>Powerful</td>
<td>Can work at a range of spatial scales</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Key</td>
<td>Spatial</td>
<td>Supports spatial models</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Key</td>
<td>Spatial</td>
<td>Supports geospatial data</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Key</td>
<td>Stable</td>
<td>Continuity and good prospects of longevity</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Key</td>
<td>Usable</td>
<td>GUI for configuring and setting up models</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>High</td>
<td>Auditable</td>
<td>Description of models can be accessed/recorded</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>High</td>
<td>Auditable</td>
<td>Ability to record and trace the assumptions, data sources, model version, uncertainties</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>High</td>
<td>Dynamic</td>
<td>Can manage different time-steps between models</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>High</td>
<td>Powerful</td>
<td>Ability to run computationally intensive models</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>High</td>
<td>Spatial</td>
<td>Includes geospatial processing components</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>High</td>
<td>Stable</td>
<td>Robust</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>High</td>
<td>Stable</td>
<td>Is maintained, improved, and bugs are fixed</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>High</td>
<td>Supported</td>
<td>User forum, support, bug reporting</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>High</td>
<td>Usable</td>
<td>Little effort to set up linkages and add existing components in an assembly</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Importance</td>
<td>Category</td>
<td>Criterion</td>
<td>CSDMS</td>
<td>OMS3</td>
<td>OpenMI</td>
<td>OpenPalm</td>
<td>TIME</td>
<td>Pegasus</td>
</tr>
<tr>
<td>------------</td>
<td>----------</td>
<td>---------------------------------------------------------------------------</td>
<td>-------</td>
<td>------</td>
<td>--------</td>
<td>----------</td>
<td>------</td>
<td>---------</td>
</tr>
<tr>
<td>High</td>
<td>Usable</td>
<td>Easy to adapt existing models for the frameworks. Low intrusiveness</td>
<td>=</td>
<td>=</td>
<td>✓</td>
<td>✓</td>
<td>=</td>
<td>✓</td>
</tr>
<tr>
<td>High</td>
<td>Usable</td>
<td>Scenarios - can investigate and compare models with a range of inputs or parameters</td>
<td>=</td>
<td>=</td>
<td>=</td>
<td>=</td>
<td>=</td>
<td>=</td>
</tr>
<tr>
<td>High</td>
<td>Usable</td>
<td>Provides utilities or components for visualization of temporal and spatial data</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>x</td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>High</td>
<td>Usable</td>
<td>Provides utilities to obtain data from external databases</td>
<td>?</td>
<td>=</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>=</td>
</tr>
<tr>
<td>High</td>
<td>Usable</td>
<td>Ability to distribute pre-populated sets of models and data</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>High</td>
<td>Usable</td>
<td>Polished end-user applications can be set up</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>=</td>
<td>✓</td>
<td>=</td>
</tr>
<tr>
<td>Medium</td>
<td>Dynamic</td>
<td>Looped configurations allowed</td>
<td>✓</td>
<td>✓</td>
<td>=</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Medium</td>
<td>Powerful</td>
<td>Parameter sensitivity analysis</td>
<td>x</td>
<td>=</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Medium</td>
<td>Powerful</td>
<td>Calibration tool</td>
<td>x</td>
<td>=</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Medium</td>
<td>Powerful</td>
<td>Uncertainty analysis and ensemble modelling</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Medium</td>
<td>Supported</td>
<td>Training available</td>
<td>x</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Medium</td>
<td>Usable</td>
<td>Can generate basic user GUI's easily</td>
<td>✓</td>
<td>?</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>Medium</td>
<td>Usable</td>
<td>Supports simple models that can be run quickly in a workshop</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td>x</td>
</tr>
</tbody>
</table>
### Revised version

<table>
<thead>
<tr>
<th>Importance</th>
<th>Category</th>
<th>Criterion</th>
<th>CSDMS</th>
<th>OMS3</th>
<th>OpenMI</th>
<th>OpenPalm</th>
<th>TIME</th>
<th>Pegasus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>Auditable</td>
<td>Previous model/data/parameter assemblages can stored and re-run</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Key</td>
<td>Dynamic</td>
<td>Supports dynamic models</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Key</td>
<td>Dynamic</td>
<td>Supports static models</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Key</td>
<td>Extensible</td>
<td>Can add/remove/substitute components, not a fixed assemblage</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Key</td>
<td>Open</td>
<td>Framework software is open-source</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Key</td>
<td>Open</td>
<td>Framework is available for anyone to use at low or nil cost</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>X</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Key</td>
<td>Open</td>
<td>Uses open standards for model interfaces</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>X</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Key</td>
<td>Open</td>
<td>Uses open standards for data types and interfaces</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Key</td>
<td>Powerful</td>
<td>Can work at a range of spatial scales</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Key</td>
<td>Spatial</td>
<td>Supports spatial models</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Key</td>
<td>Spatial</td>
<td>Supports geospatial data</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>High</td>
<td>Auditable</td>
<td>GUI for configuring and setting up models</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>High</td>
<td>Auditable</td>
<td>Description of models can be accessed/recorded</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>High</td>
<td>Dynamic</td>
<td>Can manage different time-steps between models</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>High</td>
<td>Powerful</td>
<td>Ability to run computationally intensive models</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>High</td>
<td>Spatial</td>
<td>Includes geospatial processing components</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>High</td>
<td>Stable</td>
<td>Robust</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>High</td>
<td>Stable</td>
<td>Is maintained, improved, and bugs are fixed</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>High</td>
<td>Supported</td>
<td>User forum, support, bug reporting</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>High</td>
<td>Usable</td>
<td>Little effort to set up linkages and add existing components in an assembly</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>High</td>
<td>Usable</td>
<td>Easy to adapt existing models for the frameworks. Low intrusiveness</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Importance</td>
<td>Category</td>
<td>Criterion</td>
<td>CSDMS</td>
<td>OMS3</td>
<td>OpenMI</td>
<td>OpenPalm</td>
<td>TIME</td>
<td>Pegasus</td>
</tr>
<tr>
<td>------------</td>
<td>----------</td>
<td>---------------------------------------------------------------------------</td>
<td>-------</td>
<td>------</td>
<td>--------</td>
<td>----------</td>
<td>------</td>
<td>--------</td>
</tr>
<tr>
<td>High</td>
<td>Usable</td>
<td>Scenarios - can investigate and compare models with a range of inputs or parameters</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>High</td>
<td>Usable</td>
<td>Provides utilities or components for visualization of temporal and spatial data</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>x</td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>High</td>
<td>Usable</td>
<td>Provides utilities to obtain data from external databases</td>
<td>?</td>
<td>⬤</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>=</td>
</tr>
<tr>
<td>High</td>
<td>Usable</td>
<td>Ability to distribute pre-populated sets of models and data</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>High</td>
<td>Usable</td>
<td>Polished end-user applications can be set up</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>=</td>
<td>✓</td>
<td>=</td>
</tr>
<tr>
<td>Medium</td>
<td>Dynamic</td>
<td>Looped configurations allowed</td>
<td>✓</td>
<td>✓</td>
<td>=</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Medium</td>
<td>Powerful</td>
<td>Parameter sensitivity analysis</td>
<td>X</td>
<td>=</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>X</td>
</tr>
<tr>
<td>Medium</td>
<td>Powerful</td>
<td>Calibration tool</td>
<td>X</td>
<td>=</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>X</td>
</tr>
<tr>
<td>Medium</td>
<td>Powerful</td>
<td>Uncertainty analysis and ensemble modelling</td>
<td>X</td>
<td>X</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>X</td>
</tr>
<tr>
<td>Medium</td>
<td>Supported</td>
<td>Training available</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>X</td>
</tr>
<tr>
<td>Medium</td>
<td>Usable</td>
<td>Can generate basic user GUI's easily</td>
<td>✓</td>
<td>?</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Medium</td>
<td>Usable</td>
<td>Supports simple models that can be run quickly in a workshop</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td>x</td>
</tr>
</tbody>
</table>
Appendix B  Details of tests with New Zealand data and models

A.1 WATYIELD
Setting up the legacy WATYIELD code as an OMS3 component

The WATYIELD model is available as a legacy Visual Basic (VB6) application, which is a programming language developed by Microsoft. The WATYIELD application guides the user through a series of input dialogs and prompts him to specify Excel-based input files containing model parameters. This test looked at the feasibility to extract and wrap this sample code into an OMS3 component. Technically, it would have been possible to transform the VB code into a VB-based dll, prepare a wrapper for the new dll such as was undertaken for OVERSEER (see later description), and then call the WATYIELD dll via JNI from an appropriately annotated OMS3 component. However, following an analysis of the original VB code, it was discovered that the legacy VB code mixes user-interface functionality (e.g., the graphical user interface components or GUI) with computational functionality, which made it more difficult to extract the required functionality for implementing a DLL. It was easier in the end to re-implement the core WAYTIELD functionality (i.e., point-based water budget calculations) directly in Java.

Therefore a new version of a point-based WAYTIELD model was implemented in Java as an OMS3 component. Appropriate annotations were implemented for required input parameters, such as time series for rainfall and evapotranspiration as well as land cover specific coefficients. It was fairly easy to set up the model engine in this way. The test showed that re-writing as Java code is in some cases a faster and ‘cleaner’ way of creating OMS3 modelling components than the wrapping of legacy models and/or code. This test also demonstrated the capabilities of OMS3 for dealing with simple time-stepping models.

An example WAYTIELD point-scale simulation was run and set up successfully (see later for example output).

Setting up the model with climate time series provided from a web data service and plotting results

The one-point WATYIELD simulation was run chained to a Sensor Observation Service (SOS) OMS3-compliant client component, which was set up by NIWA, eventually as an external service. The SOS client queried the service to obtain rainfall and evaporation data for a given location, which was then input to the WATYIELD model for calculation. This component was set up successfully and without problems.

Two additional components were written in addition to annotating the WATYIELD model to make it OMS3 compliant. One component was written to communicate with the SOS service: to send the appropriately formatted request, and then parse the response in order to extract the data required by WATYIELD. Another component was written using java JFreeChart library in order to plot the three output arrays: rainfall, profile, and discharge.
Displaying time series results

The results of the chained components (data retrieval, model run) passed to a graphical plotting component to generate a graphical display of the results as described above. OMS3 has inbuilt capabilities for plotting time-series, but at the time we were unaware of this fact. An example output is shown below: the image shows the console with the top display of the simulation file, the bottom display of the output, and the windows with the graphical output.
Set up model spatially

To run the point-based OMS3 WATYIELD component in a spatial context (e.g., for all hydrological subunits of a catchment) we tested how to run it in conjunction with spatial input data. To achieve this OMS3-based component was set up for calling the point-based WATYIELD model for each pixel of the rasterised input data sets and produced an ASCII grid file as output. This was linked with a data acquisition component as described below.

Importing spatial data from a web service

OMS3 provides no bindings to the OGC Web Coverage Service (WCS) and Web Feature Service (WFS) interfaces used as data sources during the WATYIELD testing. Modules were created in WATYIELD that allowed parameterised calls to be created and sent WCS and WFS instances (these modules were deliberately designed for re-use outside of WATYIELD). The modules constructed URLs containing WFS/WCS complaint key value pair parameters that were submitted using wget. The responses we saved as files local to the WATYIELD application and then fed into the GDAL functions that processed the data.

We developed OMS3-based web-service clients to retrieve vector and raster input data sets of land cover, rainfall, and potential evaporation from Landcare Research OGC-compliant
web services using web feature services (WFS) for vector data and web coverage service (WCS) for raster data. The retrieved files were stored locally and converted by purpose-built OMS3-based components into ASCII grid files. These purpose-built components called GDAL-based routines for data conversion. These components were set up, linked, and run successfully.

A.2 OVERSEER and SPARROW

Set up OVERSEER as an OMS3 component

Two different OVERSEER versions were available. The ‘OVERSEER5’ version was a simplified version of OVERSEER provided as a dynamic link library (dll) built in the Borland Delphi compiler. The second version, OVERSEER6, was a .NET based dll of the full OVERSEER model. We describe how both of these variants were implemented, because they highlight different aspects of OMS3.

For OVERSEER5, we first attempted set up an OMS3 component by calling the OVERSEER5 dll from java, but this was unsuccessful due to incompatibilities between java used in OMS3 and the Delphi library. We also attempted to set up Fortran code to call the dll, and to set this up as a Fortran-based OMS3 component. While this ran, there were memory problems due to incompatibilities between the open-source version Fortran compiler used by OMS3 and the Delphi library interface. Hence, our only option was to compile the Fortran as an executable using a proprietary Fortran compiler that was more compatible with Delphi, and then to wrap this executable as an OMS3 component. This example highlights some of the difficulties when working with dll’s compiled in other programmes, but also the successful work-around of wrapping the dll in an appropriate way.

We also set up the .NET-based OVERSEER6 model as an OMS3 component. The OVERSEER6 engine is distributed as a dll rather than source code, so we were forced to work with the dll. OMS3 is written in Java so a mechanism was needed to enable it to communicate with the .NET dll to create an OMS3 compliant component.

In OMS3, integration for non-Java languages is achieved through JNI (Java Native Interface) “stubs” that act as the bridge between Java and the native code. For Fortran, C and C++, annotations can be entered into the code in the non-Java language, and OMS3 then generates the stubs automatically. Integration of .NET is also feasible using JNI, but OMS3 does not automate this, and the modeller has to construct the JNI stubs themselves.

There is an open source library called jni4net that strives to hide all the complexities of setting up a JNI bridge between Java and .NET. This library is a work in progress, and therefore incomplete in its support of all .NET features, and would not have been suitable to support the full OVERSEER6 dll interface. However, we were able to simplify the OVERSEER6 engine interface by writing a very simple .NET wrapper that only exposed the aspects of the dll that we were interested in (those used in the CLUES model). That done, jni4net produced the necessary JNI stubs and to set up an OMS3 component for a restricted OVERSEER6 model.

Set up a user interface to modify OVERSEER input file and run the model

The OVERSEER6 engine described above takes as input a complex XML file containing the full list of parameters needed to run the model. We prepared a simple form-like graphical
editor that prompts the user for changes to a number of specific parameters. The inputs are passed to an XML editor that programatically edits the input files and then runs OVERSEER using the new set of supplied parameters. The OMS3 user interface then enables these components to be linked, initiated and monitored.

Note that this test proved the capability of modifying the OVERSEER input parameters but does not include the major scientific difficulty of ensuring that the system described is biologically feasible.

Below as an image of the editor interface:
Set up and run OVERSEER provided as a web service

The OVERSEER Ver6 engine was set up by Rezare Systems (www.rezare.co.nz) as a web service hosted on their servers. Basically it allows the user to programatically submit an OVERSEER input file via a simple http POST command and retrieve the output file without having to utilize the complex web interface.

An OMS3 compliant service client was written to submit the input file and retrieve the output and substituted for the locally-run OVERSEER Ver6 engine component described above. The client successfully called and executed OVERSEER via this web service.

Call OVERSEER repeatedly to loop over a number of records

The CLUES model needs OVERSEER to be run repeatedly to calculate the leaching from each spatial unit, which are stored in a simple csv-based record structure. As explained above, we were not successful in setting up OVERSEER Ver 5 dll as an OMS3 component directly or to be called as a component from a Fortran-based OMS3 component. Hence, to
conduct the repeated OVERSEER5 calculations, we resorted to a higher level of wrapping, that is, wrapping a compiled Fortran executable which calls OVERSEER5 repeatedly as an OMS3 component. This was a simple solution to an otherwise complex problem. Although it is not ideal, it still allows a model to be implemented as OMS3 component with minimal effort and thus be utilized as part of a chain in a larger interoperable model. A potential advantage is that future updates to a DLL could be relatively easily incorporated into OMS3.

For calling OVERSEER6 repeatedly, we set up an OMS3 component written in Java repeatedly call the OVERSEER6 dll linked through jni4net. This was successful. A further step, which was not done due to lack of time, would have been to set up the repeated calls to an OVERSEER6 OMS3 component via an OMS3 simulation file (script).

**Set up SPARROW catchment routing OMS3 component.**

In the CLUES model, the files produced by after running OVERSEER repeatedly are fed to ‘SPARROW’ routines which add additional sources such as point sources, calculate decay in streams and lakes, and route the loads down the drainage network. These calculations are done with four simple Fortran subroutines. Since OMS3 can work with Fortran code, the sparrow subroutines were a good candidate for testing how easily a Fortran model can be adapted to OMS3.

Each of the subroutines was annotated with the appropriate OMS3 annotations. Inputs and output parameters were listed in an OMS3 simulation file. An OMS3 simulation project was created by using the standard OMS3 layout and build files. By running the build target, all binaries (Fortran, Java and JNI stubs) are created and the simulation is ready to be run by opening the simulation file in the OMS3 console.

This was a clear demonstration of how easy it can be to create OMS3 components from existing, relatively simple Fortran code. Each subroutine became an OMS component which could be reused in other simulations. However, in the case of a more complex model, much more work may be needed to re-architect the original code into a collection of viable sub-components that can be implemented within OMS3.

Additionally, because OMS3 is multithreaded, components that are not interdependent ran at the same time. Results were held in memory until they were needed.
Linked OVERSEER6 runs and SPARROW catchment routing

The final step was to link the component that run the set of OVERSEER runs to the SPARROW routing components using an OMS3 simulation file. This was successful.

A.3 APSIM

An attempt was undertaken to compile APSIM (a combination of .NET, C++ and Fortran libraries) on a Linux 64 platform with the intention of using the jni4net library to generate some stubs on a sample .NET library. Unfortunately, jni4net is too primitive at this stage to support Mono, and additionally the .Net APSIM libraries are often interdependent, which complicates the creation of a wrapper. This example highlights the difficulties of re-using components from another framework, especially between .NET and Java – based frameworks.