18th World IMACS / MODSIM Congress, Cairns, Australia 13-17 July 2009
http://mssanz.org.au/modsim09

A standards based web service interface for hydrological models

Peter Fitch[1] and Qifeng Bai[1]

Affiliations: [1]CSIRO Land and Water. (Email Peter.Fitch@csiro.au)

Abstract: There are many environmental modelling frameworks most of which are tied to a particular technology platform, thus limiting the impact and adoption. In this work we present a web service application, which allows TIME (the invisible modelling framework) models which are currently tied to Microsoft .NET technology, to be used by other technology platforms and a broader range of clients thus overcoming this limitation.

Currently there are a number of hydrological models that are coded using TIME. These models are typically used as application components and integrated into larger more complex modelling applications such as E2. TIME has had reasonable uptake in Australia and many useful models have been implemented using TIME.

In this work, an Open Geospatial Consortium (OGC) Web Processing Service (WPS) 0.4 interface specification web service has been developed to make those TIME models potentially available to a far broader set of clients.

The web service was developed in such a way to allow for easy integration of new TIME models by manually editing a configuration file. The application uses the Spring Framework to enable this dynamic functionality and uses nVelocity and MonoRail project (a Model View Controller) from Castle Project to facilitate the overall functionality.

Major issues found with hydrological modelling web services are unknown model run time is unknown and large model input and output data volumes. This causes problems to clients wanting to use using modelling web services over HTTP, which is stateless, and request response oriented. In such a situation the latency of the modelling runs and data transfer becomes problematic. These problems were overcome by using a WPS invocation strategy in which the web service would use web accessible resources for input and output data URL and return a URL for the results data. The client would poll the location returned periodically, until the output was available.

It was also found that the support for complex data structures with a current generic WPS client was not available and that the complex data structures also created difficultly with a generic workflow application.

In spite of these difficulties WPS was found to be a useful interface to hydrological models. The software for this application is available by emailing the author at peter.fitch@csiro.au.

Keywords: *TIME, OGC, WPS, Hydrological modelling, web service, SOAP, web, service*

1. A STANDARDS BASED WEB SERVICE INTERFACE FOR HYDROLOGICAL MODELS.

1.1. Introduction

Most model integration frameworks only partially solve the model integration problem. For example, the invisible modelling framework (TIME) first introduced in 2003 promised to provide a framework to allow models to be developed and integrated quickly, easily and consistently Rahman et al. (2003). This framework, which is widely by hydrological model applications, is platform specific and tied to Microsoft .NET technologies. It does not cater for the use of non-TIME models within TIME applications nor for TIME models to interoperate with non-TIME applications.

The benefits of interoperation include greater reuse which alleviates the need to port models to other platform which is an inherently error prone process Freebairn et al (2005). Using the same model code gives the modeler confidence in being able to compare model results, as the version and implementation of the model are identical. Web service technology has the promise of benefitting integration of environmental models Argent (2004) but has yet to realize that promise.

In this paper we describe a software application, which partially overcomes these limitations by using standards based web services to enable TIME models interface to and interoperate with a broader range of non-TIME clients. The web service standard used is the Open Geospatial Consortium (OGC) Web Processing Service (WPS) specification version 0.4. In addition this work assesses the suitability of WPS for hydrological modelling web services.

2. BACKGROUND

The need to integrate environmental models and the approaches used are well documented Argent (2004), Gijsbers et al (2005) and Rahman et al (2003) and the state of the art is described by Argent (2004) and the European HarmonIT State of the Art review undertaken by HR Wallingford Struve et al (2002). This review has directly led to the development of one of the more active model integration approaches being OpenMI Gijsbers et al (2005). OpenMI defines a standard software interface, which allows models that support OpenMI interface to be integrated into applications easily Dudley J. et al (2005).

TIME on the other hand specifies a framework, which abstracts model development complexities from environmental model developer. This abstraction and the provision of supporting components such as data handing and basic Geographical Information System (GIS) functionality has led to a number of models and applications such as E2 Argent et al (2005) being developed using TIME. TIME however is primarily a model development framework and the problem remains of how to integrate TIME models with non-TIME applications. Because of this uptake in TIME it is desirable to make TIME models available to a broader range of clients, with benefits as described previously.

2.1. Open Geospatial Consortium (OGC)

The Open Geospatial Consortium is a non-profit consensus driven standards organisation that is leading the development of standards for geospatial applications making them available and accessible. OGC has been successful in developing and promoting standards for data exchange (Web Mapping Service and Web Feature Service) achieving good adoption with software vendors. OGC is also interested in standards for processing services. In addition CSIRO is a member of OGC and is interested in contributing to the OGC with relevant work in our problem domain. It is for these reasons WPS was chosen for this work.

2.2. Web Processing Service (WPS)

WPS is a general-purpose web processing standard mainly used for geo-processing applications to date. WPS (like all OGC web service standards) is a service which responds to both GET and POST requests via HTTP. With the GET request, the request data is passed to the service as part of the request URL, where as with the POST request the request data is passed separately as key value pairs.

The current version of WPS is 1.0 though this work used the pre-release version 0.4 (OGC 05-007r4) of WPS as 1.0 was unavailable at the time. The specification is extensive and complex and is only briefly described here. For further information the reader is directed to WPS version 1.0.0 document 05-007r7 on the OGC web site.

2.3. WPS methods

The three methods supported by WPS are described in a bit more detail below.

WPS Method	Method Description
GetCapabilities	This method describes the capabilities of the service, in particular the URL and port of the other methods, as well as providing an enumeration of processes available and some meta information on those.
DescribeProcess	Returns more detailed information to the client on a process offering, sufficient to allow invocation without prior knowledge. This includes data required and structure for invocation as well as more detailed description of what the process actually does.
RunProcess	Invokes the process, returning either the results or a URL for the results depending on the invocation call.

Table 1: WPS Methods

2.4. WPS Data Structures.

The WPS specification does not define any specific data structures for the processing with those being left to the implementation. The specifics of the implementation can be determined dynamically by using the DescribeProcess WPS method. The objective of the DescribeProcess method is to provide a client sufficient information to invoke the method without prior knowledge. Below are two snippets from the DescribeProcess for our SimHyd Chiew et al (2002) rainfall runoff model from our application. These snippets give the reader an insight into the data structures required to run a hydrological model. The first snippet defines an input parameter to the model (infiltration coefficient). Note that the definition is a minimum only requiring a value, as this is a proof of concept work, but should be tightened considerably for production use. It is possible to define data type, unit of measure (UoM), and constrain the multiplicity to 1.

```
<Input>
        <ows:Identifier>infiltrationCoefficient</ows:Identifier>
        <ows:Title></ows:Title>
        <ows:Abstract></ows:Abstract>
        <LiteralData>
            <SupportedUOMS defaultUOM=""/>
            <ows:AnyValue/>
        </LiteralData>
        <MinimumOccurs>1</MinimumOccurs>
</Input>
```

The second snippet shows the rainfall input time series definition noting that the data is defined as ComplexData conforming to a particular schema. The schema supported by this application is a customized based on the TIME TimeSeries data structure Rahman et al (2003). Unfortunately although the standard supports tremendous flexibility with data types, this flexibility makes the development of generic WPS clients very difficult as handling generic data structure without any structure information know *a-priori* is very difficult.

Again this definition is minimal with further restriction required for production use.

```
<Input>
        <ows:Identifier>rainfall</ows:Identifier>
        <ows:Title></ows:Title>
        <ows:Abstract></ows:Abstract>
        <ComplexData defaultFormat="text/XML" defaultEncoding="base64" defaultSchema="">
          <SupportedComplexData>
            <Format>TimeSeries</Format>
            <Encoding>UTF-8</Encoding>
            <Schema></Schema>
          </SupportedComplexData>
        </ComplexData>
        <MinimumOccurs>1</MinimumOccurs>
</Input>
```

3. DESIGN CONSIDERATIONS

The software application was designed to meet a number of design criteria which were identified prior to development. Perhaps the most important of these is the recognition that WPS over HTTP protocol is stateless, connection based, without guarantee of message delivery. The consequence of this is firstly the clients will time out if a response is not available within a reasonable timeframe (often 30-60 seconds) and secondly subsequent requests have no knowledge of previous requests. If the timeout is increased the connection to a web service is prone to disconnections for a variety of reasons. As a result there is a requirement to have an asynchronous invocation protocol, where the run can be started and when completed the client is notified. Fortunately WPS has a mechanism where the output from the processing service is transferred to a web accessible resource, and if the client knows the URL of the output, it can poll the location periodically to detect when the output is available and hence determine when the run has finished.

Another consideration is the recognition of the overhead transferring large volumes of data to and from models over the web. To alleviate this problem, a web accessible storage area was created which could be used to store infrequently changing datasets. These data sets could then be used by the modelling web service as WPS accepts a URL as a data source.

Another requirement was that the implementation should make it easy for new TIME models to be added and configured. This was achieved by using the Spring framework (http://www.springframework.net), which allows software modules to by dynamically loaded and used, based on settings within a configuration file. This configuration file was manually edited as new models were added.

Also it was desired to have a convenience SOAP interface using an ad-hoc specification, which would be used by TIME applications and would be "TIME friendly". This requirement was achieved by having the WPS interface as a proxy on top of the SOAP interface.

4. IMPLEMENTATION ARCHITECTURE

The implementation architecture makes use of a layered approach with three layers: WPS Service, WPS SOAP Service and TIME Model Library,

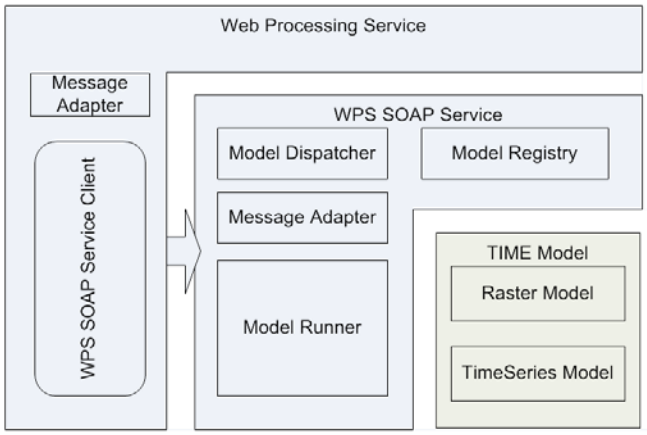

Figure 1: High level architecture and interfaces

Figure 1 above presents the high level architecture as a block diagram illustrating that the Web Processing Service is effectively a proxy "on top" of a Simple Object Access Protocol (SOAP) web service. This was done to have a convenient non-standards based interface to the modeling service for TIME applications. The SOAP interface has 4 methods, which mimic the WPS methods, which are: GetCapabilities, DescribeFeature, RunModel (Timeseries Model), RunRasterModel. The SOAP interface separates time series and raster based models because the required datatypes are different and have to be explicitly specified.

Lastly, the SOAP web service makes use of the TIME model library which is a directory containing TIME models which are registered for use in the model registry.

4.1. Frameworks used.

This work makes use of Spring and two additional software frameworks. The rationale and benefits of the frameworks are discussed here. Spring is a framework, which uses the principle of inversion of control to provide a consistent means of configuring and loading objects dynamically at run time (http://www.springsource.org).. MVC is a well-known software design pattern found in Fowler (2003) and this was implemented using the Castle Project Model View Controller (MVC) Monorail (http://www.castleproject.org/MonoRail). The pattern splits the user interface into three distinct independent roles: the model; the underlying business object classes, the view; different presentations of those classes, and a controller; the logic that determines which view to use depending on what part of the model has been changed. An advantage of MVC is low coupling between model, view and controller.

The other framework used is the nVelocity templating framework (http://nvelocity.sourceforge.net). This framework uses an XML template in conjunction with some scripting on the document to decouple the software domain objects from the view. The benefit of this is that it is easy to update schema changes by updating the template using a text editor with no need to recompile the application. This allows changes to the response document to be made in a matter of seconds.

4.2. Configuration and adding new models.

The application is designed to make it extremely easy to add new models. The web application has a library directory in which all the TIME model Dynamic Linked Libraries (DLL's) are placed. Each model placed in that directory with an accompanying xml definition file is automatically added to model registry.

With reference to Figure 1 the addition of new TIME models is achieved by editing the XML model configuration document, which is loaded into the model registry. An example definition for a SimHyd rainfall runoff model is show below. Note the four main sections of the configuration: name and location of SimHyd TIME DLL, description and type of inputs, outputs and parameters.

```
<?xml version="1.0"?>
<ModelConfiguration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
Name="D:\src\mdbSRC\TIME\Models\SimHyd\SimHydCS\bin\Debug\SimHyd"
LoadFrom="SimHydCS.dll">
<Description>SimHyd Rainfall Runoff Model</Description>
  <Inputs>
          <Input Name="Precipitation" Type="TimeSeries" />
          <Input Name="PET" Type="TimeSeries" />
  </Inputs>
  <Outputs>
         <Output Name="runoff" Type="TimeSeries" />
  </Outputs>
          <Parameters>
                  <Parameter Name="imperviousThreshold" Type="double" />
                  <Parameter Name="rainfallInterceptionStoreCapacity" Type="double" />
                  <Parameter Name="perviousFraction" Type="double" />
                  <Parameter Name="soilMoistureStoreCapacity" Type="double" />
                  <Parameter Name="infiltrationShape" Type="double" />
                  <Parameter Name="infiltrationCoefficient" Type="double" />
                  <Parameter Name="interflowCoefficient" Type="double" />
                  <Parameter Name="rechargeCoefficient" Type="double" />
                  <Parameter Name="baseflowCoefficient" Type="double" />
         </Parameters>
</ModelConfiguration>
```

5. SYSTEM OPERATION

Figure 2 below is a UML sequence diagram describing the operation of the system's major components for a RunModel call.

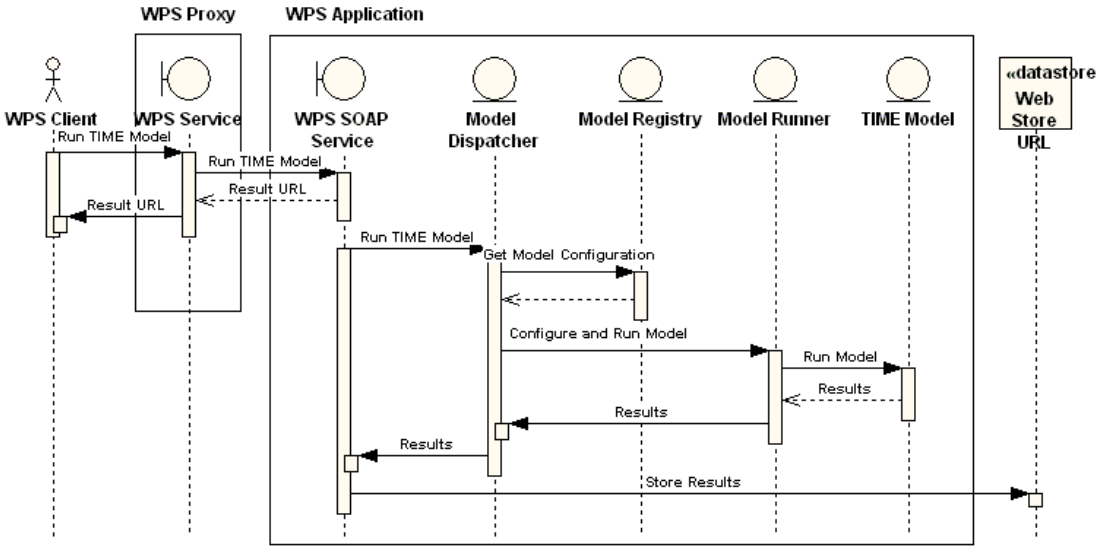

Figure 2: System Run Model Sequence Diagram

The sequence is initiated by a call from the client to run a particular TIME model, supplying data either in the form of inline data or web accessible URL's, and parameters for the model. The client receives a response from the WPS service of the web accessible URL at which, on completion of the model runs, the result will be found. The data store is represented on the right hand side of the figure. The other notable aspect of this sequence is the way in which the model dispatcher configures the model runner based on configuration information contained within the model registry. This approach implemented with the Spring framework allows new TIME models to be added easily to the web service, ready for immediate use without any recompilation of code.

6. TESTING AND DISCUSSION

The application was successfully tested using browsers and custom .Net testing code. The uDig WPS client from 52North (http://52North.org) was tested but found not to work with our hydrological modelling WPS due to limited support for non-spatial data types. So although WPS provides a standard interface for invocation of hydrological models, there is still a difficulty with support for complex and non-standard data types. This data problem is potentially alleviated by the development of WaterML developed by the Consortium of Universities for the Advancement of Hydrological Sciences Inc. (CUASHI http://www.cuahsi.org/WaterML/1.0). WaterML specifies an XML encoding for hydrological time series data and getting reasonable uptake in the U.S. with some testing by the Bureau of Meteorology in Australia.

Some testing was also done with Kepler (http://www.kepler-project.org) which is a scientific workflow package and has its origins in the eco-informatics community. Kepler has a generic SOAP web service component (actor in Kepler terminology), which was used to create a workflow to test the hydrological modeling service. Unfortunately the difficultly with handling the complex data structures would require implementation of custom data handling components using Java, which was not done due to time constraints.

7. CONCLUSION

This paper describes the development of a software application, which provides TIME model functionally to a broader range of clients. Our work was motivated by the desire to reuse exiting models and thereby negate the need to port these models to different platforms as well as to interoperate with a wide range of technology neutral clients now becoming available such as the uDig WPS client and Kepler. Unfortunately the flexibility of WPS with respect to required to data structures is both a blessing and a curse. It means that WPS can be configured support complex data types, but that support correspondingly does not exist in the generic WPS clients currently available.

In spite of that, WPS was found to be a useful standards-based interface to TIME hydrological models.

In developing the application all the frameworks used were found to be extremely helpful. In particular, Spring for enabling flexible model addition. nVelocity was also notable for its ease in enabling changes to syntax of the response to very easily done by manually editing the response template.

Currently the application does not support model calibration and requires the model parameters to be supplied and known a-priori. A useful extension would be to include model calibration. As well the application only currently supports spatial or temporal model and not spatio-temporal models.

8. ACKNOWLEDGEMENTS

The authors would like to acknowledge the helpful suggestions from reviewers Peter Taylor and Rick Meng, which have allowed the manuscript to be improved. Also, thanks to the anonymous reviewers who have provided good advice on how to improve the manuscript for publication.

9. REFERENCES

Argent R.M (2004). An overview of model integration for environmental application-components, frameworks and semantics, *Environmental Modelling &ASoftware*, 19 (2004) 219-234.

Argent R.M., Grayson R.B, Podger G.M, Rahman J.M, Seaton S., Perraud J-M (2005b), E2-A Flexible Framework for Catchment Modelling, Proceedings MODSIM 2005.

Chiew F, Scanlon P, Vertessy R, Watson F (2002), Catchment Scale Modelling of Runoff, Sediment and Nutrient Loads for the South East Queensland EMSS", CRC for Catchment Hydrology Technical Report 01/2002, Melbourne, 2002.

Dudley J., Daniels W., Gijsbers PJA., Fortune D, Westen S, Gregersen JB, Applying the Open Modelling Interface (OpenMI), Proceedings of MODSIM 2005.

Freebairn A., Rahman J., Seaton S, Perraud J-M, Hairsine P, Hotham H (2005), Development of an automated testing tool for identifying discrepancies between model implementations, Proceedings of MODSIM 2005.

Fowler M., Patterns of Enterprise Application Architecture, Addison Wesley, 2003

Gamma, E., R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: elements of reusable object oriented software,* Addison Wesley, 1994.

Gijsbers P.J.A, Gregersen J.B, OpenMI: A glue for model integration, *Proceedings of MODSIM 2005.*

Rahman J.M, Seaton S.P, Perraud J-M, Hotham H., Verrelli D.I and Coelman J.R (2003), Its TIME for a New Environmental Modelling Framework, *Proceedings of MODSIM 2003.*

State of the Art Review, Work Package 1, Struve J, Western S, Millard K, Fortune D, *HR Wallingford Report SR 598 September 2002.*