Soap or Rest - Web Services delivery patterns

This is a short note I wrote in response to a request from MFISH about our approach to web services. AVW.

Web services delivery patterns tend to fall into two camps - ReST and SOAP. (there are others)

These two systems are often portrayed as conflicting alternatives but they are not - they are different ways of doing web services due to different requirements and modes of use.
The key thing to remember is that SOAP is about Accessing Objects - i.e computational units consisting of data and functions while ReST is about accessing Resources - mostly documents and data but also models and processes.

The ReST model uses the concept of resources and sits on top of existing web protocols such as HTTP. It is stateless but still works as you can create a resource - a document that represents the request to run the model and receive back a URL, at a later date visiting this URL will give access to the results. In the mean time notification about the completion of the process can be done using APP - Atom Publishing protocol or RSS

I note the TIME system 'is platform specific and tied to Propriety technologies' This should be strongly avoided. although the interoperable framework should be agnostic as to the platform and programming language used for the model itself.

Also we should keep elements separate - in that a model development framework is different to a model integration framework.

I like the idea of a web accessible data storage service that limits the amount of information that needs to be passed around.

This can be tied to another OGC standard - Catalogue Service (CWS) This provides a way to access metadata about datasets, documents and presumably models.

We are already implementing a Metadata Catalogue here based on GeoNetwork software that implements this service.

So the elements you have are

1. a place to put data sets for easy access, each one has a URL

2. a catalogue to reference data sets, and models containing metadata that tells you what you need to know to use the data or run the model, e.g. file types

3. a protocol WPS that allows models to be made available as services

4. a protocol APP that can publish results

What would be developed would be a set of wrapper layers that connect hosted models to the network.

Some directory and governance elements - discovery, user management, access control, billing perhaps.

I am familiar with the various software tools mentioned in the paper, and although the SDT use different tools they fill similar niches providing MVC and IOC design patterns.

typically SDT uses JBOSS as the application host, and SEAM as the framework.

Currently we interface back end models written in Fortran, IDL & R to web services (Tides, SolarView, HIRDS)