*International Environmental Modelling and Software Society (iEMSs)*
*2010 International Congress on Environmental Modelling and Software*
*Modelling for Environment's Sake, Fifth Biennial Meeting, Ottawa, Canada*
*David A. Swayne, Wanhong Yang, A. A. Voinov, A. Rizzoli, T. Filatova (Eds.)*
*http://www.iemss.org/iemss2010/index.php?n=Main.Proceedings*

# Exposing the Kepler Scientific Workflow System as an OGC Web Processing Service

**Andrew Pratt** [a] **Chris Peters**[b], **Siddeswara Guru**[c], **Brad Lee**[b] **and Andrew Terhorst**[b]

[a]*Hydro Tasmania, Hobart Tasmania, Australia (andrew.pratt@hydro.com.au)*

[b]*CSIRO Tasmanian ICT Centre, GPO Box 1538, Hobart Tasmania, Australia (firstname.lastname@csiro.au)*

[c]*CSIRO Marine and Atmospheric Research, GPO Box 1538, Hobart Tasmania, Australia (siddeswara.guru@csiro.au)*

**Abstract:** The Open Geospatial Consortium (OGC) - Web Processing Service (WPS) provides an interface for distributed geoprocessing: from discovery and description of available processes, description of input and outputs, to execution and retrieval of results. While the integration of various geoprocessing libraries into WPS implementations negates the need to redevelop common algorithms, authoring of processes for the WPS is still very much in the hands of the software developer, rather than the scientist. We propose that by integrating a workflow execution engine (in our case, Kepler) as a WPS processing engine, process content authoring becomes more accessible to the scientist. Existing geoprocessing framework integration solutions have a large reliance on additional user-generated metadata to properly define capabilities as required by the WPS specification. There may be many different possibilities for integrating Kepler but we are particularly interested in solutions that reuse and extend the model markup description already available in the Kepler workflow descriptions to make the task of adding new processes less onerous for the user. However, runtime typing in Kepler and mismatch in complex type definition between Kepler and the WPS have made this difficult. As a result, we have had to define our own input and output parameter types in a separate file. The proposed Kepler-WPS integration work was inspired by the need to reuse a process for creating gridded rainfall time-series from point-based measurements for ingestion in a rainfall-runoff model.

***Keywords:*** Kepler workflow, Web processing Service, Scientific workflow

## 1 INTRODUCTION

In recent times, experiments conducted by researchers have become more collaborative in nature. This is partly due to the willingness of scientists to share data, resources and knowledge. This has also enabled scientists to conduct complex experiments with distributed data and resources. Therefore, the need has emerged for systems to couple data with processing programs and visualisation tools. Scientific workflow systems have become a popular choice to couple disparate environments, executable programs and data from different sources [Altintas et al., 2006]. For example, it is possible to provide data from file systems or databases to programs written in Java and visualise the result in an independent visualisation tool. In scientific workflow, the scenario explained is composed into tasks and they are decomposed into components of the workflow.

Even though scientific workflow tools can handle web services to access data and processes, the workflow tool itself is generally a stand alone desktop application. The common practice is to use the desktop application to design the workflow and later share it with others via a website

like `myExperiment.org`. From the WPS perspective, there has been a general lack of geoprocessing algorithms implemented [Olaya, 2010], inspiring the coupling of several geoprocessing libraries like SEXTANTE and Geographic Resources Analysis Support System (GRASS) with the WPS. Currently, the WPS authoring process is more often served by a one-off broad-brush integration activity of existing low level geo-algorithms rather than piece-wise high level user-generated content. The non-existence of high level processes produces a need for additional orchestration technologies on top of the OGC services stack, adding further technologies and complications.

In this paper, we describe the exposure of Kepler [Kepler, 2004] as a Open Geospatial Consortium - Web Processing Service (OGC-WPS) and some of the challenges and issues that arise. The aim is to allow users with little software development experience the means to author algorithms visible to the wider Service-Oriented community. Kepler thus becomes both an authoring tool for web service accessible process and the composition and execution engine behind the web service.

The remainder of the paper is organised as follows: Section 2 gives an overview of Kepler scientific workflow system. In Section 3, the WPS is explained. Section 4 describes the Kepler-WPS Integration design and Section 5 describes progress of the prototype. Section 6 gives some of the challenges in mapping concepts of Kepler to WPS. The conclusion and potential future work is outlined in Section 7.

## 2 KEPLER WORKFLOW SYSTEM

Kepler is a popular scientific workflow visual authoring and execution tool widely used in ecology, bioinformatics and hydrology domains [Kepler, 2004]. Kepler introduces domain polymorphism and modal models [Brooks et al., 2008] and supports actor-oriented design. Domain polymorphism enables kepler to use same component in different domains and modal models allow the combination of different models of computation. Actor-oriented modelling separates two modelling concerns: component communication (dataflow) and overall workflow coordination (orchestration) [Bowers and Ludäscher, 2005].

Directors, Actors, Ports and Parameters are the fundamental components of Kepler. Directors implement models of computation and handle workflow orchestration. Actors are configured using parameters and pass data to each other using ports. Actors implement the algorithms or processing steps that are chained together. An Actor can have a single port or multiple ports. Ports in an actor are used to define input and output data and are categorised as input, output or input/output. Relations are used to branch the data flow, and to send the same data to multiple actors. Workflows in Kepler are stored in an XML-based format called Model Markup Language (MoML). Figure 1 shows the kepler workbench environment.

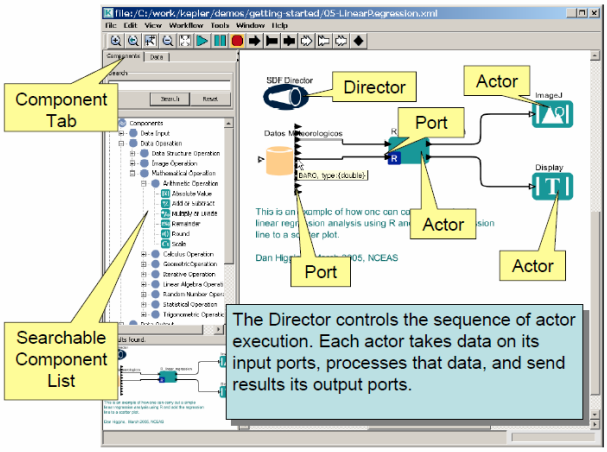


Figure 1: Kepler workbench: taken from [Kepler, 2004]

There are several projects in different domains that use Kepler for data acquisition, archiving, and analysis. The Science Environment for Ecological Knowledge (SEEK) is a system designed to facilitate data acquisition and archiving, integration, transformation, analysis and synthesis of ecological data [SEEK, 2005]. Real-time Environment for Analytical Processing (REAP) focuses on developing scientific workflow tools that can be used to access, monitor, analyse and present information from field-deployed sensor networks [REAP, 2007]. Due to its flexible design, pPOD has adopted Kepler for the bioinformatics domain [pPOD, 2010]. Jäger et al. propose using Kepler to orchestrate distributed geospatial processing in its traditional role as a desktop application, but without the focus on interoperable geospatial web services [Jäger et al., 2005]. We choose Kepler for its flexibility of reusing existing components, and the separation of its model of computation (in the form of directors). These two features give us the ability to reuse existing environmental data capture, processing actors and execute workflows with different models of computation without any modification to the workflow.

## 3 WEB PROCESSING SERVICE

The OGC-WPS is a web services based standard for providing description and execution of processes with a particular focus on geospatial processing of vector and raster data, designed around the publish, find, bind pattern [WPS, 2007]. Previous work in WPS-based research has exposed various Application Programming Interfaces (APIs) and geospatial toolsets to the WPS, with the aim of increasing the number of available low level geoprocesses and improving the runtime performance of resource-intensive models. Woolf and Shaon propose an approach to encapsulate grid infrastructure within the WPS, providing the means for running resource-intensive processing through a standard processing interface [Woolf and Shaon, 2009]. The SEXTANTE geospatial analysis tools are exposed through the WPS, providing access to more than 200 geoprocesses [Schäffer, 2009]. Brauner and Schäffer exposes the GRASS to the WPS, and proposes a method of chaining low-level processes together using Business Process Execution Language (BPEL) [Brauner and Schäffer, 2008].

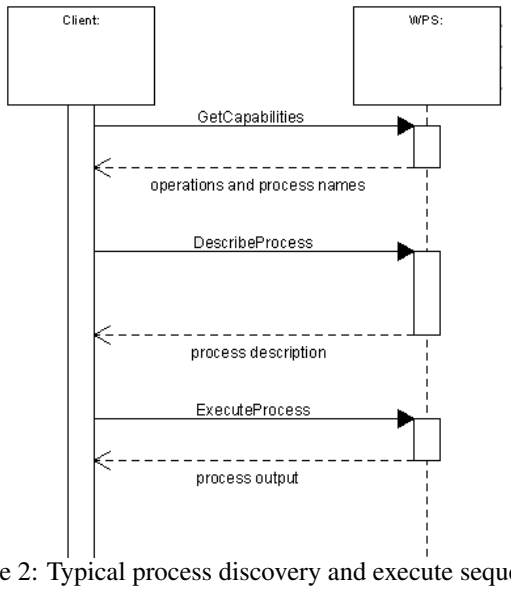


Figure 2: Typical process discovery and execute sequence

The WPS provides several operations for discovering, querying and executing processes via various interfaces including Simple Object Access Protocol (SOAP), Hypertext Transfer Protocol (HTTP) GET and POST. The WPS Specification summarises the operations as follows [WPS, 2007]:

- GetCapabilities: This operation allows a client to request and receive back service metadata (or Capabilities) documents that describe the abilities of the specific server implementation.
- DescribeProcess: This operation allows a client to request and receive back detailed information about the processes that can be run on the service instance, including the inputs required, their allowable formats, and the outputs that can be produced.
- Execute: This operation allows a client to run a specified process implemented by the WPS, using provided input parameter values and returning the outputs produced.

Figure 2 shows a typical client-server call sequence to discover, query and execute a process on WPS.

Schäffer proposes a transactional profile, defining a DeployProcess and UndeployProcess operation for dynamically registering new processes with the WPS [Schäffer, 2008]. These transactional operations introduce the concept of deployment profiles, which allow technology specific profiles to be implemented on WPS instances. The default profile for the 52°North WPS implementation is BPEL. The solution proposed in this paper includes a deployment profile to allow Kepler Model Markup Language (MoML) to be deployed to the WPS, using the 52°North deployment profile design.

## 4 KEPLER-WPS INTEGRATION DESIGN

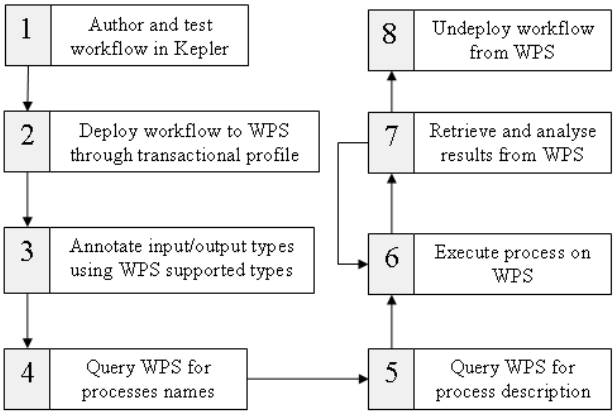

Figure 3: Lifecycle of Kepler workflow in the WPS

Figure 3 illustrates the full lifecycle of the proposed deployment of kepler workflow on WPS. In step 1, a user designs, experiments and tests a workflow using the traditional Kepler workflow environment. In Step 2, they use the DeployProcess call of the transactional WPS to submit the workflow in the form of a Kepler MoML file (xml-based). In Step 3, the user annotates the workflow through the admin webpage of the WPS. The WPS parses the Kepler MoML file, and prompts the user with the inputs and output ports of the workflow requiring annotation with supported WPS simple or complex types. This creates a WPS configuration file on the server that associates the relevant port descriptions with valid WPS type specifications. Steps 4, 5, 6 and 7 are the typical sequence for interacting with the WPS through process discovery, additional process metadata retrieval, and finally execution and analysis of results. Step 8 may occur when the user decides that the workflow is of no further use or they wish to edit and replace with a newer version of the workflow.

Figures 4, 5 and 6 show the interaction of the components for each of the WPS request types. In Figure 4, the WPS hosts an algorithm repository (a folder on the file system) where Kepler MoML files are stored along with port data type annotations that conform to WPS simple and complex type definitions. Process names are retrieved directly from the top-level Actor in the Kepler workflow file. During the DescribeProcess request in Figure 5, metadata from the Kepler

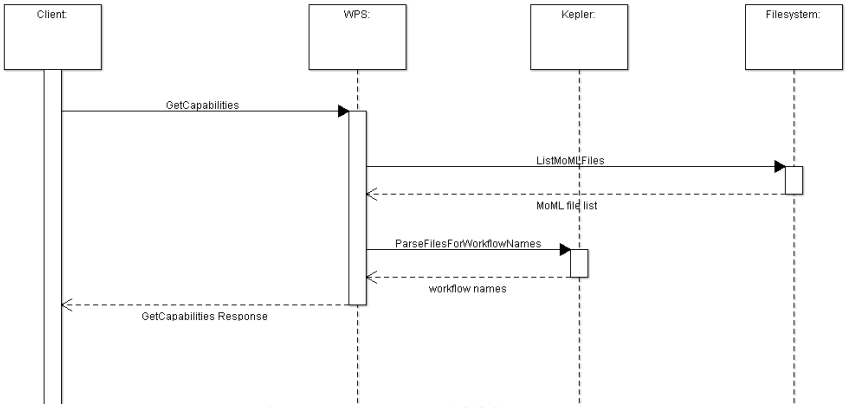


Figure 4: GetCapabilities request

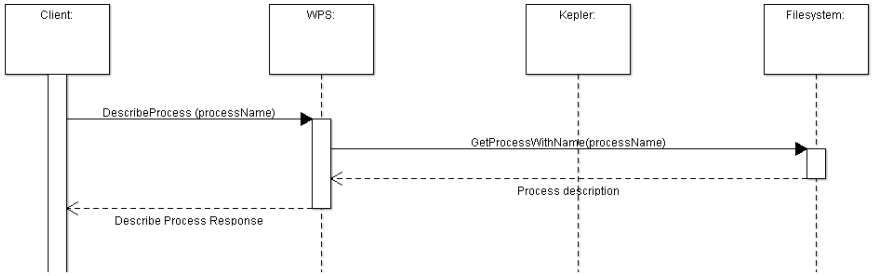


Figure 5: DescribeResponse request

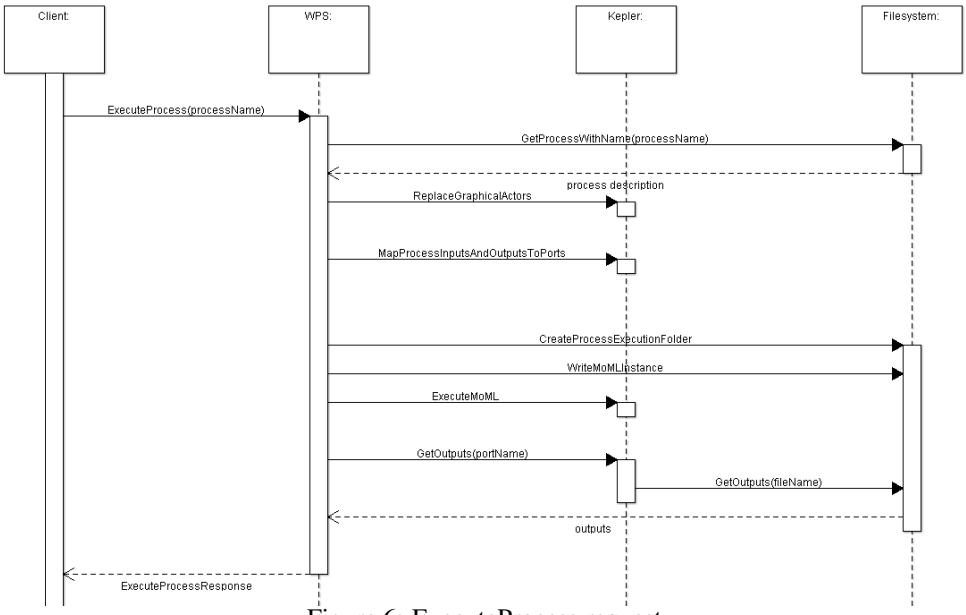


Figure 6: ExecuteProcess request

workflow file is combined with the corresponding WPS configuration file containing the annotated port types to provide a description of the process along with names and data types of its inputs and outputs. In Figure 6, graphical actors are replaced with non-graphical equivalents in order to allow the workflow to execute without a Graphical User Interface (GUI). This modified workflow is then further modified so that it writes inputs and outputs to a uniquely named job folder for the workflow run, and its ports are dynamically connected to Actors linking it to input parameters provided through the WPS interface. Upon completion of the process execution, the workflow outputs are retrieved from the uniquely named job folder and returned through the WPS interface.

## 5 IMPLEMENTATION

We have implemented a bare skeleton of the solution, by adapting the 52°North WPS implementation. The handling of complex types, deployment profiles and type annotation has not yet been implemented.

The significant working parts of the prototype are:

- Implementation of the WPS GetCapabilities interface via linking process names to MoML files in the file system
- Implementation of the WPS DescribeProcess by parsing associated Kepler MoML files for simple types.
- Draft implementation of the WPS ExecuteProcess call through the Kepler runtime engine via non-graphical replacements actors using Hydrant libraries for parameterless processes [King, 2010].

## 6 ISSUES AND CHALLENGES

The aim of this work was to provide a type marshalling layer between the WPS and Kepler that would allow workflows designed in Kepler to be uploaded to the WPS and run seamlessly without additional process metadata. However, there are some challenges to make this a reality.

There is a mismatch between the Kepler runtime typing and the WPS static typing. In the WPS, the names and types of input and output parameters must be declared during description of the process. In Kepler, while the input and output parameter names can be inferred from port names, the types of those parameter may not necessarily be defined declaratively. Kepler actors can choose to implement runtime type checking allowing them to cope with a variety of different input parameter types. For instance, an actor that takes two numbers as input, sums them and returns a single number as output need not be rewritten for each numeric type available, but instead checks that the type of each of the input parameters conforms to the set of intended types that can be sensibly summed, and asserts the output parameter type as that type at runtime. While this feature is useful in Kepler for reuse of actors across a range of well-defined types, it lacks the sort of declarative definition that the WPS requires. As a result of this mismatch, the number of actors that can be automatically mapped to the WPS is drastically reduced, and artificial constraints are imposed during the workflow design phase in order to cope with the WPS context later in the workflow lifecycle.

Kepler has only fairly rudimentary capability for declaring complex types as the inputs to workflows or actors (workflow components). Kepler ports can be labelled with a type, but for all types that aren't primitives (int, float, double, string etc.), the base type Object is declared on the port, and it is up to the actor to type cast from Object to the appropriate class. Developers can extend Kepler to define new port token types, but this involves software changes to the Kepler distribution for each new type added. This is in sharp contrast to the WPS that defines complex types with a combination of mime-type, encoding, and schema descriptors. Since the WPS is defined as an interface specification, it expects that inputs and outputs need to be defined in an unambiguous, platform-neutral manner in order to deserialize data streams through the interface in a format that the process can understand. The schema property is used for XML based formats, and is a

URL pointing to the XML Schema definition. Table 1 lists a variety of different triples of mime-type, encoding and schema descriptions for some well known and not-so-well-known formats and compares them against their declarative type in a standard Kepler distribution.

| WPS datatype | Mime-type | Encoding | Schema | Kepler datatype |
|---|---|---|---|---|
| JPEG | image/jpeg | | | Object |
| NetCDF | application/x-netcdf | | | Object |
| GML | text/xml | UTF-8 | ./om/1.0.0/observation.xsd | Object |
| O&M | text/xml | UTF-8 | ./gml/3.2.1/gml.xsd | Object |

Table 1: Example Complex Data Type descriptions (note: schemas truncated for brevity)

## 7 Conclusions and Recommendations

We initially set out to provide a simple wrapper over Kepler to allow its workflows to be exposed to the OGC Web Processing Service without annotation to the workflow descriptions or links to external metadata. This was not an easy task due to a mismatch between input and output datatype descriptions in the two systems. Therefore, we resorted to using the existing WPS process description file in use by the 52°North implementation to describe those parts that Kepler can't be guaranteed to describe through its MoML. This proposed solution provides the ability to author and test workflows in a traditional GUI environment before deploying to the WPS for later discovery and execution by a wider community of Web service client applications. This provides the research community with a means to deploy larger workflows without the need for complex orchestration technologies to bridge numerous web services calls and pass complex data sets. Additionally, it abstracts server-side processing, allowing scientists to be content authors in the Software as a Service (SaaS) paradigm, alongside software developers.

While there are many existing workflow engines, we were particularly interested in reusing Kepler workflows already produced for the project. Future work would be most useful in examining typing systems in other workflow tools and solutions that close the conceptual gap in input/output data type definition in order to make switching from visual workflow authoring to WPS-based process execution more transparent.

The major limitations in coupling Kepler with the WPS are related to the way in which input and output types are declared. The Kepler optional runtime typing doesn't conceptually fit well with the WPS declarative type system. Additionally, the use of Object for complex types in Kepler, which within a Java-only environment allows actors to runtime type check and provides maximum flexibility and reusability, wasn't a good conceptual fit for the WPS declarative type system based around mime-type, encoding, and schema for complex types. Therefore, future studies around these aspects and how they are addressed in various workflow environments and the potential for addressing with enhancements to Kepler may be considered. Additionally, the integration of workflow tools with other OGC services may be explored, to examine the benefits of different interface descriptions on the ease of coupling the service interface with the execution engine.

## Acknowledgments

Part of this work was conducted when Andrew Pratt and Siddeswara Guru were working in CSIRO Tasmanian ICT Centre.
This project is jointly funded by the CSIRO Water for a Healthy Country Flagship and the Tasmanian Government. The Tasmanian ICT Centre is jointly funded by the Australian Government through the Intelligent Island Program and CSIRO. The Intelligent Island Program is administered by the Tasmanian Department of Economic Development and Tourism.

## REFERENCES

Altintas, I., O. Barney, Z. Cheng, T. Critchlow, B. Ludäscher, S. Parker, A. Shoshani, and M. Vouk. Accelerating the scientific exploration process with scientific workflow. *Journal of Physics*, 46: 468–478, 2006.

Bowers, S. and B. Ludäscher. *Conceptual Modeling ER 2005*, volume Volume 3716/2005 of *Lecture Notes in Computer Science*, chapter Actor-Oriented Design of Scientific Workflows, pages 369–384. Springer Berlin / Heidelberg, 2005.

Brauner, J. and B. Schäffer. Integration of grass functionality in web based sdi service chains, 2008. Obtained through the Internet: "http://www.osgeo.org/ocs/index.php/foss4g/2008/paper/view/133" [accessed 03/12/2009].

Brooks, C., E. A. Lee, X. Liu, S. Neuendorffer, Y. Zhao, and H. Zheng. Heterogeneous concurrent modeling and design in java (volume 2: Ptolemy ii software architecture). Technical Report UCB/EECS-2008-29, EECS Department, University of California, Berkeley, Apr 2008.

Jäger, E., I. Altintas, J. Zhang, B. Ludäscher, D. Pennington, and W. Michener. A scientific workflow approach to distributed geospatial data processing using web services. In *Proceedings of the 17th international conference on Scientific and statistical database management*, 2005.

Kepler. The kepler project, 2004. Obtained through the Internet: "http://kepler-project.org" [accessed 20/01/2010].

King, T. Web front end for the kepler scientific workflow application, 2010. Obtained through the Internet: "http://code.google.com/p/hydrant-kepler" [accessed 26/04/2010].

Olaya, V. Introduction to geoprocessing services using SEXTANTE, 2010. Obtained through the Internet: "http://geostat2010.info/system/files/sextante_en.pdf" [accessed 01/03/2010].

pPOD. Kepler/pPOD, 2010. Obtained through the Internet: "http://daks.ucdavis.edu/kepler-ppod/" [accessed 20/01/2010].

REAP. Realtime Environment for Analytical Processing, 2007. Obtained through the Internet: "http://reap.ecoinformatics.org" [accessed 20/01/2010].

Schäffer, B. Towards a transactional Web Processing Service (WPS-T), 2008. Obtained through the Internet: "http://www.gi-days.de/archive/2008/downloads/acceptedPapers/Papers/Schaeffer.pdf" [accessed 01/03/2010].

Schäffer, B. From geodata to geoinformation - 52°north web processing service (wps) and sextante, 2009. Obtained through the Internet: "http://download.osgeo.org/osgeo/foss4g/2009/SPREP/1Wed/ParksideGO4/1500/wedg041530Schaeffer.ppt" [accessed 01/03/2010].

SEEK. The Science Environment for Ecological Knowledge, 2005. Obtained through the Internet: "http://seek.ecoinformatics.org" [accessed 20/01/2010].

Woolf, A. and A. Shaon. An approach to encapsulation of grid processing within an OGC Web Processing Service. In *Grid Technologies for Geospatial Applications*, 2009.

WPS. OGC (2007a). OpenGIS Web Processing Service. OGC Implementation Specification. Technical report, Open Geospatial Consortium, 2007.