International Environmental Modelling and Software Society (iEMSs)
2012 International Congress on Environmental Modelling and Software
Managing Resources of a Limited Planet, Sixth Biennial Meeting, Leipzig, Germany
R. Seppelt, A.A. Voinov, S. Lange, D. Bankamp (Eds.)
http://www.iemss.org/society/index.php/iemss-2012-proceedings

The Demeter Framework for Model and Data Interoperability

Eric Fritzinger[a], Sergiu M. Dascalu[a], Daniel P. Ames[b], Karl Benedict[c], Ivan Gibbs[a], Michael J. McMahon, Jr.[a], Frederick C. Harris, Jr.[a]
[a] *Dept. of Computer Science & Engineering, University of Nevada, Reno, USA*
[b] *Department of Geosciences, Idaho State University, USA*
[c] *Earth Data Analysis Center, University of New Mexico, USA*
[a] *ericf@cse.unr.edu,* [a] *dascalus@cse.unr.edu*

Abstract: This paper presents an overview of the Demeter Framework being developed at the University of Nevada, Reno as part of the Nevada Climate Change Portal. The Demeter Framework proposes a new solution to the model coupling problem by taking a component-based approach that allows almost any standard or type of component to be integrated within the system. For example, the DotSpatial toolbox from Idaho State University is currently being integrated into the Demeter Framework to handle complex geospatial processing, and can then be linked with components from many different sources. The Demeter Framework utilizes flow control and automated data conversions to help manage the issues that arise with non-homogenous component standards. It also encourages user-generated content that can expand the functionality and availability of components, and as a result will also allow researchers to share their work freely and with very few logistical hindrances. The framework also relies on web service-based components to not only enable access to high-powered computing resources but also to help transcend the programming language and operating system barriers that many software frameworks encounter. The paper provides details on the key architectural concepts and components (activities, translators, channels, and data format converters) that make model and data interoperability possible with Demeter, presents the framework's Silverlight-based "Persephone" interface designed for user-friendly scenario configuration and execution, describes the workflow runtime web-service being developed, and, finally, provides examples of integration involving WaterOneFlow web services as data sources and DotSpatial toolset items as computational components.

***Keywords**:* Software framework; model interoperability; components; workflow, web services.

1 INTRODUCTION

As our exploration of the world around us expands, so too does our need for software that can assist researchers in their efforts to answer new questions. Particularly useful is the ability to utilize software tools from related fields to help paint a more complete picture of complex interdisciplinary questions and their possible solutions. In the case of environmental modelling, there are many inter-related fields that can contribute to answering a specific scientific question, and the tools from these fields can be combined to provide more accurate and meaningful results. We attempt to do this by providing software support for coupling models and use data from such various fields.

However, access to and use of these tools is sometimes limited by diverse obstacles – domain knowledge, standards, and the need for high-powered computing resources, to name just a few. To address some of these problems, the Demeter Framework is being developed at the University of Nevada, Reno (UNR)

to provide a fast, easy way to link models and data, and to allow climate scientists to contribute their own models, components, and data to the pool of resources readily available to the scientific community.

1.1 Issues in Model and Data Interoperability

There are several problems inherent to creating component-based software. Differences in operating systems, programming languages, and standards can cause significant difficulties when trying to link components. Software for model and data interoperability suffers from these problems, and more [Bulatewicz, 2006; van der Steen, 2006; Holzworth et al., 2010; Okamoto et al., 2010].

For example, linked models are typically written in the same language, and they are coupled by copying both sets of source code and writing some kind of controller to facilitate the execution of both. Then, they are all compiled together into a single working program. This is a perfectly acceptable method, but it can lend itself to issues with maintenance and flexibility. Specifically, if a model becomes unstable for a certain range of input values, and begins outputting incorrect values, it may become necessary to swap out that model with a similar one that can provide more accurate results in that range. However, by using the aforementioned method, this would be a daunting challenge and undertaking.

Another important issue is the interpretation of data, which pertains to data formatting. Thus, meaningful data formatting can be seen as a combination of two factors: file format and data context. For instance, XML is a general purpose file format, and a piece of software would not be able to properly read the XML file and interpret it without the human user's knowledge of its contents (data context). Regardless of whether they are in XML or are just strings of non-human-readable bytes, data formats require some amount of expert knowledge in order to facilitate the communication between models.

1.2 Proposed Solution

The Demeter Framework, developed in connection with the Nevada Climate Change Portal [McMahon et al., 2011; Dascalu et al., 2011], employs a component-based method of model coupling, allowing users to integrate almost any tools they need. Models and other data processing activities are interconnected in various modelling scenarios, certain intermediary (facilitating) components being inserted between models to make their communication possible. All of this is combined and executed in a web service-based runtime engine that allows users to start a long-running scenario, receive updates, and check on the status of the runtime execution from any place where Internet access is available. It is important to note that the Demeter Framework is a new project that does not incorporate any other existing scientific workflow toolsets or libraries.

In Section 2 of this paper we present the architecture of the Demeter Framework by introducing its key components and by describing its use of web services and runtime engine. Section 3 provides details of the "Persephone" Silverlight interface that allows users to create and manage scenarios. Section 4 focuses on applications that integrate out-of-the-box components into the framework. Section 5 provides a brief overview of related model coupling solutions. Finally, Section 6 wraps up the paper with several directions of future work and conclusions.

2 ARCHITECTURE

The architecture of the Demeter Framework was inspired by looking at many of the existing model and data interoperability standards [Benedict, 2011], in particular at component-based standards. OpenMI (http://www.openmi.org/] and the Common

Component Architecture (http://www.cca-forum.org/), to name only a couple, provided insight into how many of the existing standards are structured. With additional research, a common thread was found between them that allowed for the creation of Demeter's basic architecture [Dascalu et al., 2011]. Since then, this architecture has been evolved and several key new components have been introduced in Demeter, as described in the next sub-sections. Figure 1 depicts the interaction and the roles of these key Demeter architectural components.

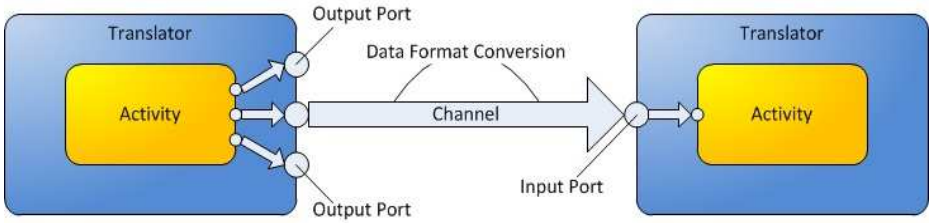

Figure 1. Key Components of the Demeter Framework

2.1 Activities

The "Activity" concept is the cornerstone of the Demeter Framework. It is the component which provides functionality and consumes the data being passed around to provide further data for a running scenario. An Activity can be anything from a simple scientific unit converter to a global climate model. Thus, an activity's granularity in a scenario will vary widely. In essence, activities are at the heart of the Demeter Framework's functionality.

2.2 Translators

The "Translator" is a component that allows activities to execute within the Demeter Framework. Where the Activity performs the computations, the Translator is provided as an intermediary between Demeter and the Activity. It literally translates the aspects of the Activity (input/output parameters, name, description, etc.) to the Framework. Likewise, it translates the needs of the Framework to the Activity. For example, Demeter will command the Translator to execute the Activity. A Translator is essentially a "wrapper" for a set of Activities, which allows the Demeter Framework to utilize them.

When dealing with different component-based standards, there needs to be a construct that is "aware" of the standard – its classes, data types, and functions. Many software frameworks are statically aware of these, whereas in the Demeter Framework the Translator is the only component "aware" of the standard. For example, an OpenMI Translator would be aware of all of the OpenMI constructs, but the Demeter Framework itself would (and needs) not know them.

This structure of using a Translator allows the Demeter Framework to integrate many different kinds of standards into a scenario without having to re-release the entire framework every time a new standard (or update to a standard) is implemented – only the Translator needs to be updated. This also provides a means of allowing a whole library of activities, written in the same standard, to be integrated into the Framework by only developing a single Translator for all of them.

2.3 Channels

A "Channel" is an architectural component that provides a line of communication between Translators. When an output from one Activity needs to be communicated to an input of another Activity, the outputting Translator sends the data across the

channel via an output port. The data is then picked up by an input port on the other Translator, and this Translator can then pass the data to the Activity it is translating. In addition to facilitating communication of data, they can also provide flow control between Activities. For instance, there might be some Activities that execute faster than others, or an Activity might constantly send data out during its execution. Other Activities in the scenario may be incapable of handling these situations, and may lose data in the process. In such cases the Channel can provide proper flow of control so as to not overburden Activities that aren't ready. Furthermore, the Channel can also provide crash recovery of the workflow execution by storing the data being sent temporarily until the receiving Activity is capable of processing it.

2.4 Data Format Converters

As data is being sent over a Channel, it needs to be transformed into a format that the receiving Activity can interpret. A "Data Format Converter," another fundamental Demeter component, does just that. During scenario configuration, the user can select an appropriate data format converter based on the expected inputs and outputs of each Activity. Each Converter can also supply conversion notes, including mentions of data loss and other descriptions. When an Activity is registered, each of the input and output parameters (represented by "ports" in the framework) are supplied, and each parameter specifies its expected data format in the form of a string-based key. To find a suitable Converter, the key is then compared against all Data Format Converters that have been registered.

One of the more useful aspects of the Demeter Framework is that it is capable of discovering a series of Data Format Converters, which eliminates the need for always requiring a direct conversion between two formats. For example, let us assume that Activity A provides a series of NetCDF files that contains climate data as output, and Activity B provides a visualization map overlay of climate data. However, Activity B expects a KML input data format. The Demeter Framework will search its library of Data Format Converters and will find that there is no direct conversion between the NetCDF files and KML. However, another user did create a Converter which will turn those NetCDF files into a WaterML format, and yet another user created a Converter which turns WaterML into KML. Now, the Demeter Framework can chain together these two Data Format Converters and provide a "conversion path" that will transform the data from Activity A into the appropriate format expected by Activity B.

2.5 Web Services

Access to high-powered computing resources has typically been needed for models and other computationally-intensive operations. Web services are applications running on a remote server, capable of receiving messages and, based on those messages, executing a series of specific commands. When creating the Demeter Framework, a significant emphasis was placed on the ability for users to use web services, and thus we employed the service-oriented architecture paradigm [Granell et al., 2010; Goodall et al., 2011].

These web services can be hosted on different operating systems and can involve different programming languages. The use of a standard communication protocol allows most web services to be accessed by any type of Activity, as long as the messages and protocol are known by the client application.

2.6 Workflow Runtime Service

The Demeter Framework utilizes a runtime web service, allowing users to access it from anywhere and utilize the University's computing resources. This enables the users to remotely manage their executing scenarios and will also provide them with the ability to start executing scenarios without burdening their own systems. Each

executing scenario is given its own space on the server hard drive (plus read/write rights on its allocated space) so if any Translators or Activities need to store (a reasonable amount of) information on the hard drives, they have the ability to do so.

2.7 Component Catalogue

A significant aspect of the Demeter Framework is the ability for users to submit their own components for use by the community. The component catalogue will allow users to browse online through Translators, Activities, and Data Format Converters that the other users of the system have submitted. In addition, descriptions, ratings, and comments will be available for the users to view, so they can decide if a particular component is right for their needs.

This user-generated content is important because it allows domain experts to contribute their knowledge and experience to the community. For instance, a Data Format Converter would need to be written by someone who knows both data formats being converted; otherwise, the conversion could be incomplete. Another example would be if a user were to develop a Translator for a standard he or she does not fully understand; the translation would be lacking in features, or be non-functional. It is important that the domain experts be allowed to contribute their knowledge to the community, and the online component catalogue will enable that.

3 THE PERSEPHONE INTERFACE

To utilize the Demeter Framework, a user interface was developed for the purpose of contributing and retrieving components from the component catalogue, configuring and running scenarios, and providing visual aids on the status of the runtime execution. This interface, shown in Figure 2, is named Persephone and has been aimed at providing an effective and pleasant experience for the user.

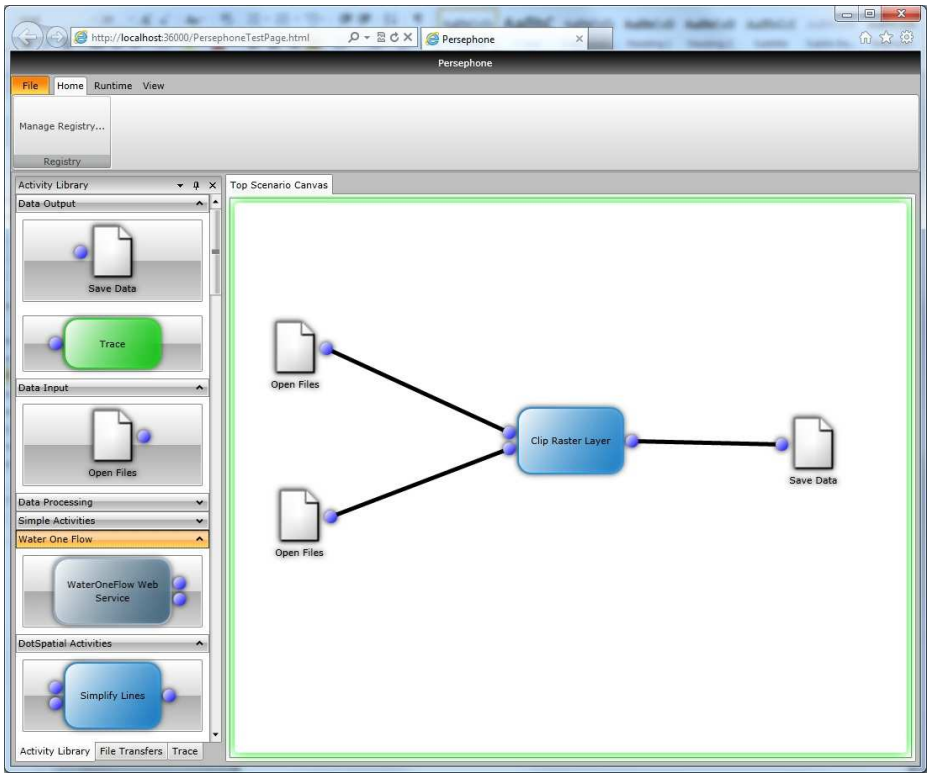

Figure 2. Demeter Framework's Persephone User Interface

3.1 Interface Structure

The organization of the interface is relatively simple at first glance, with the more complex elements residing deeper in the interface's structure. This design is intended to not overwhelm the user with all of the options presented right away.

There is a ribbon bar at the top, which provides the menu options and allows the user to switch between the framework's different modes of operation. A panel on the left side contains three smaller tabbed windows which show a list of available Activities, any pending file transfers, and a trace output window (mostly used for monitoring and debugging purposes). There is also the main canvas upon which the Activities are placed and linked together through Channels. Once the user begins registering or downloading Activities, they will show up as a list of icons in the Activity Library panel on the left-hand side of the screen.

3.2 Modes of Operation

There are two main modes of operation in Demeter: edit and execute. During edit-time, the users will generally use the "Home" ribbon tab, importing Activities from the local computer or the online catalogue and creating a scenario. During run-time, the users will use the "Runtime" ribbon tab to manage a scenario's execution.

Editing a Demeter scenario is relatively easy. First, the users have to browse and import Activities from the component catalogue, or register their own Activities (stored on their local hard drives). Once that is done, Activities will populate the "Activity Library" panel on the left in the form of icons. Next, when the user clicks an Activity icon, it will place an instance of that Activity onto the scenario canvas. From there, the user can move the Activity around on the canvas via click-and-drag actions and begin connecting Channels by clicking on the "port" symbols attached to the Activities. The ports on the left side of an Activity are input ports, and the ports on the right side of an Activity are output ports, following a "left-to-right" convention of workflow execution.

When a Channel is connected, a window will open, asking the user to configure the Channel. This is where the Framework will begin its search for a Data Format Converter path. The user will be able to select any appropriate Data Format Converter path, or "No Conversion" to simply transmit the data format as sent by the outputting Translator.

Executing a scenario is straightforward. The user has to switch to the "Runtime" ribbon tab and click "Start." This will upload the Demeter scenario to the workflow runtime service, set up the runtime system, and execute the scenario according to its configuration. Until the execution is completed, no changes can be made to the scenario configuration, for example no additional Activities or Channels can be placed onto the canvas.

The user will be updated on various events, such as which Activity is executing and what trace output is being sent. During execution, the user can click the "Pause" or "Stop" buttons to perform those respective actions, and the runtime engine will attempt to respond to the instruction as quickly as possible. If an Activity sends a file or list of files to the interface, Persephone will receive the information and add them to the "File Transfers" sub-window. When the scenario is finished, the user will be able to edit the scenario once again and the data from the execution will be cleaned up from the server. Pending file transfers, however, will remain for a time.

4 EXAMPLES OF APPLICATION

Through collaboration with our partners at Idaho State University, we are in the process of integrating several proof-of-concept components into the Demeter Framework. Specifically, the WaterOneFlow web services allow researchers to

gather environmental data using a web service standard and a separate data format standard called WaterML (http://his.cuahsi.org/wofws.html). The DotSpatial toolkit (http://dotspatial.codeplex.com/) is also being integrated into Demeter to supply a variety of geospatial processing activities [Ames et al., 2008].

4.1 Integration of WaterOneFlow Web Services

WaterOneFlow web services offer users the ability to access data from various sources using a specified standard The data format used for delivery of the data via WaterOneFlow services is known as WaterML, an extensible XML-based data standard capable of representing environmental data. The hosts of these web services consist of prominent organizations such as the USGS, NOAA, and EPA. The integration of WaterOneFlow is in the form of a GUI-based Activity, shown Figure 3, which allows the user to setup the queries and download the data into the scenario configuration. During execution, the Activity simply sends its downloaded data out in WaterML format. The user may also elect to additionally download the data to the local hard drive during edit-time as the query is being made.

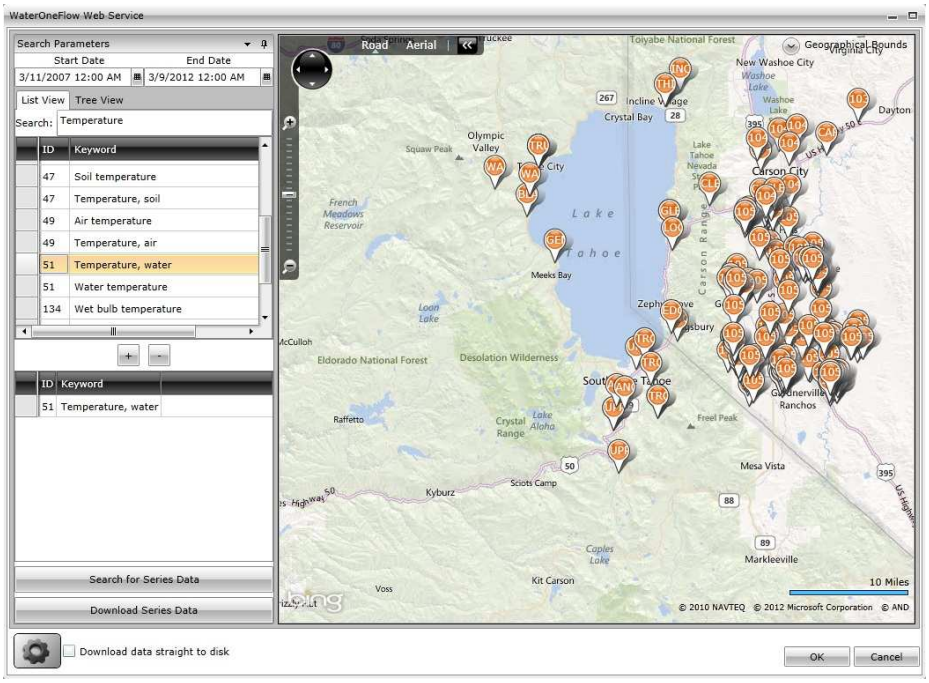

Figure 3. WaterOneFlow Integrated as a User Interface-based Demeter Activity

4.2 Integration of the DotSpatial Toolbox

The DotSpatial toolbox is an open-source software package that provides a library of spatial analysis and mapping functionality. The tools themselves are integrated as Activities, with their input and output parameters represented as ports. The DotSpatial toolbox libraries were compiled directly from the open-source DotSpatial project found on CodePlex. The Translator is being developed in such a way as to integrate the DotSpatial tools into Demeter without having to change any DotSpatial source code. In addition, due to the DotSpatial tools' need for various extensions, the Demeter's Translator infrastructure has been modified to allow the user to specify those extensions and dependencies. Currently, the Translator can read the tools, thus registering them as Activities. At the time of writing this paper, a single, example DotSpatial tool called "Clip Raster Layer" can be executed.

5 RELATED WORK

There are several model and data interoperability frameworks similar to Demeter that provide largely the same functionality. The two closest in structure and concepts to Demeter are described next.

The Earth System Modeling Framework (ESMF) provides a full runtime and coupling engine that is based on the integration of components which implement their standards and behaviours (http://www.earthsystemmodeling.org/). The user creates a hierarchical tree of modelling components and the runtime engine manages each component's execution. To adjust the modelling behaviour of the system, the branches and leaves of these trees can be replaced with similar component structures. ESMF is a solid framework that has been continually refined and maintained over the years [Hill et al., 2004; Zhou 2006].

The Community Surface Dynamics Modelling System (CSDMS) is an open-source modelling framework developed at the University of Colorado, Boulder (http://csdms.colorado.edu/wiki/Main_Page). It has a graphical user interface that allows researchers to couple models easily. The users only need to click on the component they want, and the component will then detect the other components in use and its inputs and outputs will link themselves to the appropriate inputs and outputs of these other components. CSDMS has a large library of components from which to choose and allows developers to contribute their own open source components that are written in the CSDMS standard.

Demeter differs from these frameworks in that it attempts to allow contribution of components without the need to alter source code (hence the use of Translators). Similar to CSDMS, it will maintain an in-house library of components to be made available to the users, and will allow contribution from the community. However, aside from ensuring that no malicious software is introduced, the components will not have to be open-source, and the users will be able to use their own proprietary components without contributing them. In addition, Demeter allows the users to have control over every aspect of Activity execution and communication, which enables them to refine their scenario configurations with minimal effort.

6 FUTURE WORK AND CONCLUSIONS

There is a long way ahead before Demeter will be as complete as some of the existing frameworks. However, we believe that it will be capable of performing well with further sustained research and development work. As we go forward, there are several areas of expansion that we plan to explore, as follows.

During execution, it may be necessary to alter the behaviour of the runtime engine in order to accommodate more complex scenario configurations. This includes being able to introduce conditional runtime controls that will only execute Activities under certain data circumstances (if-then-else conditions) or run an Activity multiple times based on data output (while loops). Other runtime controls may include sub-workflows, which would enable users to create smaller, reusable scenario configurations that can be utilized in a "plug-and-play" fashion.

Another potential enhancement would be to provide users with the ability to define their own runtime engines and sources. For instance, if a user wants a specialized behaviour to take full advantage of a sub-set of components, a new runtime engine could be developed and provided to the user to execute. Additionally, the source of the runtime engine could be altered, so that instead of the workflow runtime web service that has been developed and hosted at UNR, the users could specify a runtime web service they are hosting using their own servers, or perhaps even execute the engine on their local computers.

In summary, the Demeter Framework provides capabilities for integrating and running existing components, and enables the users to manage various aspects of scenario configuration and execution. In the near future it will also maintain a library of components available for free download and use. While similar in function and structure to other frameworks, Demeter has been developed to limit any restrictions on its use and to allow users from all disciplines to quickly get started with interconnecting models and data sources.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under grants EPS-0814372 and EPS-0919123.

REFERENCES

Ames, D.P., Michaelis, C.D., Anselmo, A., Chen, L., and H. Dunsford, MapWindow GIS, in Shekar, S. and H. Xiong (editors), *Enclyclopedia of GIS*, pp. 633-634, Springer, 2008.

Benedict, K., Key interoperability standards for Earth science research and applications, presentation given at the International Symposium on Remote Sensing of Environment, Sydney, Australia, April 10-15, 2011.

Bulatewicz, T.F, Jr., Support for model coupling: an interface-based approach, PhD dissertation, University of Oregon, 2006.

Dascalu, S.M., Fritzinger, E., Okamoto, S., and F.C. Harris, Jr., Towards a software framework for model interoperability, *Proceedings of the 9th IEEE International Conf. on Industrial Informatics*, pp. 705-710, Lisbon, Portugal, July 26-29, 2011.

Goodall, J.L, Robinson, B.F., and A.M. Castronova, Modeling water resource systems using a service-oriented computing paradigm, *Journal of Environmental Modelling & Software, 26*(5), 573-582, 2011.

Granell, C., Diaz, L, and M. Gould, Service-oriented applications for environmental models: reusable geospatial services, *Journal of Environmental Modelling & Software, 25*(2), 182-198, 2010.

Hill, C., DeLuca, C., Balaji, V., Suarez, M., and A. da Silva, The architecture of the Earth System Modeling Framework, *Journal of Computing in Science and Engineering, 6*(1), 18-28, 2004.

Holzworth, D.P., Huth, N.I., and P.G. de Voil, Simplifying environmental model reuse, *Journal of Environmental Modelling & Software, 25*(2), 269-275, 2010.

McMahon, M.J., Jr., Dascalu, S.M., Harris, F.C., Strachan, S. and F. Biondi, Architecting climate change data infrastructure for Nevada, in Salinesi, C. and O. Pastor (editors), *Lecture Notes in Business Information Processing, LNBIP-83*, 354-365, Springer, 2011.

Okamoto, S., Fritzinger, E., Dascalu, S.M., Harris, F.C., Jr., Latifi, S., and McMahon, Jr., Towards and intelligent software tool for enhanced model interoperability in climate change research, *Proceedings of the 2010 World Automation Congress*, pp. 1-6, Kobe, Japan, September 19-23, 2010.

Van der Steen, A., J., Issues in computational frameworks*, Journal of Concurrency and Computation: Practice and Experience, 18*(2), 141-150, 2006.

Zhou, S.J., Coupling climate models with the Earth System Modeling Framework and the Common Component Architecture, *Journal of Concurrency and Computation: Practice and Experience, 18*(2), 203-213, 2006.