A Practical Approach to Developing a Web-based Geospatial Workflow Composition and Execution System

Jianting Zhang
Department of Computer Science
The City College of the City University of New York
New York, NY, 10031
jzhang@cs.ccny.cuny.edu

ABSTRACT

Motivated by lacking the capability of supporting geospatial workflow composition and execution in a Web environment from leading GIS (such as ESRI ArcGIS), we have developed a prototype by integrating mature open source and commercial software packages in an innovative way. Our prototype system includes a client module for visual and interactive workflow editing based on Ptolemy II (a modeling and design system), a geospatial actor library representing 500+ ArcGIS geoprocessing tools for drag-and-drop-based workflow composition, a middleware as a workflow engine to schedule and execute ArcGIS Geoprocessing tools based on composed geospatial workflows, and, a Web-GIS to visualize original and derived data along a workflow processing pipeline. By reusing the mature software packages, we are able to complete the prototype development within weeks instead of months or years. A site selection problem that involves multiple geospatial operations are used to demonstrate the functionality and features of the prototype system.

Keywords: Geospatial, Workflow, Web, Middleware, Integration, Reuse

1. INTRODUCTION

Visual modeling in Geographical Information Systems (GIS) has become increasingly popular. ModelBuilder (and the related Geoprocessing package) from ESRI ArcGIS [1] and Spatial Modeler of ERDAS Imagine [2] have attracted considerable application interests for a long time. These tools allow users to drag and drop icons of both data sources and processing units onto a canvas and then connect these icons together to compose a model for execution. Such visual geospatial modeling can be considered as a special type of scientific workflow system. More recently, as geospatial data are increasingly being published as Web services, including both Open Geospatial Consortium (OGC) Web services [3] and W3C WSDL (Web Service Description Language) based services [4], there are more and more applications that build geospatial data processing pipelines based on workflow technologies and use Web services as the basic processing units.

While Web-based visual workflow composition and execution are attractive due to its high usability, it is non-trivial to develop such a system due to technical complexities. First, enabling Web-based workflow composition and editing is challenging. The current generation of Web browser technologies does not provide high-performance 2-D graphics primitives to support sophisticated workflow drawing and easy-to-use event handling mechanisms to support highly interactive user operations. The newly emerging Rich Internet Application (RIA) frameworks [5], such as Adobe Flash and Microsoft Silverlight as well as HTML 5, have eased the problems to a certain extent. However, they are still not designed for sophisticated Web-based applications that can achieve the same level of user experiences as in a desktop computing environment when composing, editing and executing workflow visually and interactively. Practically, many of the existing Web-based workflow editors, such as the open source Eclipse BPEL Designer [6], provide only limited functionality when running in a Web environment. It is not a surprise that even some recent research prototypes still provide very preliminary visual and graphical editing functionality.

Second, there are a large number of geospatial operations in numerous applications and it is non-trivial to formalize these operations as workflow components due to various syntactic and semantic mismatches. Even within a single GIS where operations are usually categorized in a relatively coherent way, some operations may have complex input and output parameters with ad-hoc or non-structured default values. While they may be relative easy to specify using command lines, it can be challenging to represent them visually in a Web-based workflow editing environment. Although converting complex parameter structures into strings and use only primitive data types can certainly reduce the workflow software implementation complexity in this regard, it is undesirable as the semantics of the geospatial operations are often obscured due to the simplification of syntax.

Third, unlike business data processing that is control flow centric and requires relative insignificant computational resources, geospatial data processing can be data intensive, computation intensive, and, visualization intensive. Workflow scheduling can play an important role in reducing end-to-end completion time and improve workflow efficiency. However, very few existing geospatial workflow systems have the capability to schedule workflow executions based on different computing environments and available computing resources.

While these technical complexities are more related to software implementation engineering, they are critical in practical workflow applications. Without solving these problems properly to pave the roads for wide adoptions, much of the scientific workflow research becomes irrelevant to practical applications. Due to recourse constraints, it is unlikely that small research groups like ours can develop a highly usable research system of geospatial processing workflow in a short time by developing various components from scratch. In this research, we report our design and development of a Web-based geospatial processing workflow system in a quick manner by reusing and integrating mature open source and commercial software. More specifically, we use the open source Ptolemy II modeling and design system [7] developed by the Center for Hybrid and Embedded Software Systems (CHESS) at UC Berkeley for workflow editing and scheduling. We then extract the syntax of ESRI ArcGIS Geoprocessing tools [1] from their documentations and format them so that these Geoprocessing tools can be used as workflow components for visual composition. By hacking the Ptolemy II source code, we are able to pass a composed workflow to our middleware for remote execution. The middleware is implemented as an Apache Tomcat [8] Web application module to communicate with the workflow composer that runs at the client side and the ArcGIS

Geoprocessing engines that run at the server side. Once the workflow is passed to the middleware, the execution plan of the workflow can be calculated by Ptolemy II library and the tasks in the execution plan can be passed to ArcGIS Geoprocessing engines for real executions. We have also setup a Web-GIS system using open source MapServer [9] and OpenLayers [10] software so that Web clients can visualize original, intermediate and resulting geospatial data. All the software involved in our system, including Ptolemy II, Apache Tomcat, ArcGIS, MapServer and OpenLayers, have long development histories and play leading roles in their respective application domains. By reusing and integrating these mature software systems, we eliminate the significant development burdens to achieve our goals in developing a Web-based workflow system to support environmental and social-economic research. With the previous knowledge and experiences the author have gained in the respective software packages, a robust and demonstrable prototype system has been developed in weeks instead of months or years.

The rest of the paper is organized as the following. Section 2 introduces background and related works. Section 3 overviews Ptolemy II features that are relevant to geospatial workflow composition and execution. Section 4 presents the details of the system design and implementations. Section 5 provides a running example for demonstration purposes and discusses a few technical issues during evaluations. Finally Section 6 is the conclusion and future work directions.

2. BACKGROUND AND RELATED WORKS

Workflow technologies become increasingly popular in the past decade in both the business and scientific applications. Scientific applications of workflow technologies, or Scientific Workflow (SWF) for short, is an indispensible component of Cyberinfrastructure or eScience and we refer readers to review articles [11, 12, 13] for detailed information. The wide adoption of Web services technologies and the availability of Web services further have made workflow technologies practically useful and desirable. While considerable research has been focusing on automatic workflow composition based on semantics and some research proposed to use workflow for automatic data archiving and provenance, the basic functionality of drag-and-drop based visual modeling (or interactive workflow composition) is still an important feature (and sometimes the primary feature) to attract end users. Indeed, visual workflow composition relieves the burdens of end users from learning programming/scripting languages that often have steep learning curves and are error-prone. Quite a few scientific workflow systems, such as Kepler [14], VisTrail [15], Triana [16] and Taverna [17] are currently available and many of them are released as open source software to serve the research communities. On the business data processing side, major software vendors such as Oracle, Microsoft and IBM have provided software products based on the WSDL, Business Process Execution Language (BPEL) [18] and other industrial standards.

There have been significant research interests in applying workflow technologies to geospatial data processing. OPERA [19] and WASA [20] are two early scientific workflow systems that have been extended for processing geospatial data. In the context of Kepler scientific workflow applications, we have proposed a scientific workflow approach to distributed geospatial data processing using Web services [21]. The approach has been applied to species distribution modeling [22], biodiversity explorations [23] and storm tracking [24]. Additional geospatial applications using the Kepler SWF system have been reported in [25, 26]. The GeoBrain group at the George Mason University has published extensively on applying semantic web and workflow technologies for satellite and sensor data processing [27, 28, 29, 30, 31]. Many of these applications use BPEL based business workflow technologies. Geospatial workflow applications based on BPEL have also been reported in [32, 33].

The development of Ptolemy II system dated back to early 2000 based on the previous developments at UC Berkeley. Ptolemy II is written in Java and uses a Java software infrastructure called Diva to render workflow components (or directors, actors, ports and parameters in Ptolemy II terminologies) and interact with users. While Ptolemy II is primarily developed for modeling, simulation, and design of concurrent, real-time, embedded systems, many of its features meet the requirements of scientific workflow systems and was chosen as the base for the development of Kepler scientific workflow system [14]. However, as Kepler is designed to be a desktop system and is too voluminous for Web applications (by using Java Web Start technology [34]), in this study, we use Ptolemy II directly for workflow composition and scheduling. While the details of Ptolemy II and its suitability for Web-based geospatial workflow design are provided in the next section, we would like to stress that the robustness of graphics editing and the extensibility of the whole system are the primary reasons that we have chosen to use Ptolemy II.

There are a few attempts to extend desktop based scientific workflow systems to the Web environment, mostly for workflow viewing and editing purposes. The Kflex system presented in [35] is an attempt to re-implement Kepler's workflow composition functionality using Adobe Flex so that Kepler workflows can be composed over the Web using a Flash plug-in. Our preliminary evaluation reveals that, while the plug-in runs smoothly in many Web browsers, it is often awkward when connecting ports among workflow processing units. The implementation does not allow customized context menus and many essential functions that are available in desktop Kepler are not available in Kflex. More importantly, Kflex is designed to reuse Kepler's existing actors and no actors representing the Geoprocessing tools are available for composing geospatial workflows. Another noticeable work is the GeoPW framework that includes a web-based workflow designer called GeoPWDesigner and a set of GRASS and GeoStar GIS modules wrapped as Web services [31]. A composed geoprocessing workflow is then executed using a BPEL engine.

ESRI ArcGIS is an industrial leading GIS and has a large user community. Recent ArcGIS releases (especially version 10) have extensive supports for Web services and geoprocessing. For example, in addition to C/C++ and .NET languages, ArcGIS 10 also provides Java and Python languages for Geoprocessing tools. The Geoprocessing Java APIs are more robust than user-developed wrappers and are easier to be integrated with third-party Java-based packages, including Ptolemy II and Apache Tomcat. Unfortunately, the popular ArcGIS ModelBuilder tool [2] is still only available to desktop computing environments. Enabling geospatial workflow composition and execution over the Web is practically useful to many applications across domains.

3 PTOLEMY II FOR WORKFLOW APPLICATIONS: AN OVERVIEW

Ptolemy controls the execution of a workflow via so-called directors that represent models of computation. Individual workflow steps are implemented as reusable actors that can represent data sources, sinks, data transformers, analytical steps, or arbitrary computational steps. An actor can have multiple input and output ports. We refer readers to [36, 37] for more formal descriptions of the actor-oriented workflow design. Different from event-driven and control-flow driven business workflow systems, Ptolemy (and hence Kepler) is based on dataflow process networks [38] that have built-in support for stream-based and concurrent execution. In Ptolemy/Kepler, due to the dataflow and actor-oriented design, actors interact only via their communication channels (i.e., by data passed between their connections) and do not directly communicate with other actors [7, 37]. This approach leads to greater reusability of actors, and decouples actors from the overall workflow execution semantics. While we only use the simplest SDF (Synchronous Data Flow) director, where processing tasks can be statically scheduled on a serial machine or a parallel system, in our current study, it is also possible to use more complex directors (such as the Process Network [38] – PN director) to accommodate more heterogeneous computing environments.

In Ptolemy II, in addition to allowing an output port of an actor to be linked to the input port of another actor, a relation instance can be explicitly created to link among multiple ports as shown in Fig. 1. Internally, the direct link between two ports is represented as two links between the two ports and an implicit relation. A link serves as data channel where tokens can pass through and metered. The existence of Relation in Ptolemy II allows formally modeling sophisticated data flow scenarios such as feeding one token to multiple ports, a feature that is missing in many other workflow systems. Another feature in Ptolemy II is the notion of multiport where multiple data channels (links) can be connected to a single port. This is very useful when a port expects an array of values from output ports of multiple actors, such as Intersect and Union tools in ArcGIS Geoprocessing where one parameter requires a list of shapefiles. The notion of ParameterPort is also useful for geospatial workflows. A ParameterPort has a PortParameter whose value can be either derived from a predefined expression or be derived from the data token that is fed to the ParameterPort. Naturally, we can model default values in ArcGIS Geoprocessing tools as PortParameters and associate them with ParameterPorts. Another useful feature in Ptolemy II is its type checking system that allows to determine structural compatibility among ports [39]. For example, if the output of port has a string type while its connecting regular input port requires an array of strings then an error can be reported to identify potential problems in workflow design. We have explored the idea for geospatial data processing [40].

Above all of the useful features we have discussed, we found the Modeling Markup Language (MoML) that is supported by Ptolemy II particularly useful for creating and editing workflows. Since both directors, actors and workflows are internally represented as MoML segments, adding actors to a new workflow through drag-and-drop or removing actors from an existing workflow are reduced to modifying MoML documents. Once editing of a workflow is completed, the resulting MoML document can be immediately streamed to other modules for further process. In our case, as detailed in the next section, the MoML document can be sent back to a Web server for remote execution. Users can also modify the MoML representations of actors that are typically Java classes. By adding ports or modifying parameters, actors that are used in workflows can be different from their original forms without reprogramming, a feature is desirable in many cases. In this study, as detailed in the Section 4.1, we utilize this feature to configure a generic geoprocessing actor into 500+ actors by providing module specific properties using MoML representation. The similar approach has been applied to designing "conceptual" actors and "abstract" workflows [41] and semantic workflow validation [42].

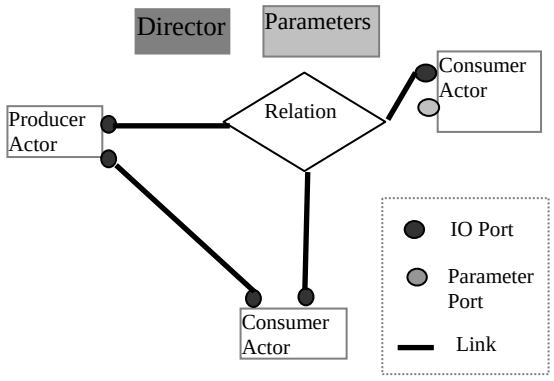

Fig. 1 Illustration of Basic Components in Ptolemy II

4 SYSTEM DESIGN AND IMPLEMENTATIONS

Our system has four major components, including a Java applet based geospatial workflow composition environment, a geospatial actor library representing 500+ ArcGIS geoprocessing tools for drag-and-drop-based workflow composition, a middleware to execute composed geospatial workflows and a Web-GIS application to visualize original, intermediate and result data. Before providing details of the implementations of individual components, we next briefly explain the high-level design and the communications among the components (also see Fig. 5 in Section 4.4).

The Java applet for geospatial workflow composition is developed on top of Ptolemy II code base. In order to expose the interfaces of ArcGIS geospatial processing tools to workflow users, we have decided to represent these interfaces as Ptolemy II actors so that they can be used as workflow processing units through drag-and-drop based composition, in a way similar to using other Ptolemy II actors. Towards this end, we have developed a Java program to extract the syntax of hundreds of ArcGIS processing tools semi-automatically from ArcGIS Geoprocessing manual and online resources. We have also modified Ptolemy II graph editor by adding a new top-level button (as shown in Fig. 6 in Section 5). When the button is triggered, instead of executing the workflow at the client side as Ptolemy II Web Start programs do, the workflow is sent back to a middleware implemented as a Java servlet residing in Apache Tomcat Web server for remote execution. The middleware, which also serves as a workflow execution engine, then parses the composed workflow and invokes Ptolemy II APIs to schedule the execution of the workflow. For each scheduled task

represented as an actor, the middleware derives input values from the output ports of connecting actors and formulate proper APIs to executable the modules, including ArcGIS processing tools, using the input values. Since all of the original, intermediate and resulting geospatial data are stored as well-accepted geospatial data formats, they can be accessed over the Web by using a Web-GIS. The URLs of Web-GIS applications to visualize these data are then output to a dynamic Web page which will be sent back to Web clients when executions of the geospatial workflows are completed. Web users then can follow the links in the result summary page to visualize the data that are involved in any step of workflow executions.

4.1 Generating Geoprocessing Actors

We have found that the ArcGIS Geoprocessing Quick Guide has the most succinct syntax to express different types of parameters of the tools and is suitable for expert users. However, the explanations of the parameters are missing in the Quick Guide. On the other hand, the online resources have detailed information on the parameters but the syntax of the tools seems to be not sufficiently informative. For example, the options of parameters are provided in the parameter section instead of the tool syntax section. We thus combine the two resources by creating two tables, one for tool level syntax and one for parameter level annotations and join them based on the tool names and parameter names. An example of the Intersect tool in the Overlay toolset of the Analysis toolbox is shown in Fig. 2.

We use the following rules to formulate various types of information associated with ArcGIS Geoprocessing tools into Ptolemy actors. (1) The mandatory inputs and outputs are mapped to ptolemy.actor.TypedIOPort (2) The non-mandatory categorical parameters are mapped to actor properties. (3) The non-mandatory numeric parameters are mapped to ptolemy.actor.ParameterPort. (4) For categorical inputs (mostly non-mandatory), ptolemy.actor.gui.style.ChoiceStyle is used to enumerate the lists of options. (5) Description of a tool is represented as a property of an actor using ptolemy.kernel.util.StringAttribute. (6) Description of a parameter is represented as a property of the parameter also using ptolemy.kernel.util.StringAttribute. The development of the parsing tool is straightforward based on the following well-accepted notation adopted in ArcGIS Geoprocessing documentation: parameters within <> are mandatory while parameters within {} are non-mandatory. For each parameter, if the expression is separated by semicolons then it is a collection/array type parameter otherwise it is a singular type parameter. An expression separated by | denotes a list of mutually exclusive options and the parameter is considered categorical. As an example, the MoML representation of the Intersect tool after the formulation process is listed in Fig. 3 with key elements highlighted or underscored. Fig. 4 shows the corresponding graphic representation of the ArcGIS Intersect actor (upper-right part) and the configuration interface of its parameters. Note that a black triangle represents a regular port, a gray triangle represents a parameter port and a white triangle represents a multi-port. As a PortParameter, cluster_tolerance can receive values from either the configuration dialog or from its parameter port.

Intersect: creates an output feature class containing features that fall within the area common to both input datasets

Intersect <features{Ranks};features{Ranks}...> <out_feature_class> {ALL | NO_FID | ONLY_FID} {cluster_tolerance} {INPUT | LINE | POINT}

<in_features {Ranks};in_features {Ranks}...> A list of the input feature classes or layers. …
<out_feature_class> The feature class to which the results will be written. …
{NO_FID | ONLY_FID | ALL} Determines which attributes from the Input Features will be transferred to the Output Feature …
{cluster_tolerance} Cluster tolerance is the distance range in which all vertices and boundaries in a feature class are considered identical or coincident. …
{INPUT | LINE | POINT} Choose what type of intersection you want to find. …

Fig. 2 Command syntax and documentation of ArcGIS Geoprocessing Tools using the Intersect tool as an example

4.2 Customizing Workflow GUI

Despite a powerful system, the code base of Ptolemy II remains small. The Java jar file for Ptolemy II 4.01 is less than three megabytes which makes it suitable to run Ptolemy II graphical applications as Java Web Start applications, as demonstrated in the Ptolemy II website [7]. However, Ptolemy II Web Start applications run entirely at the client side which is not suitable for geospatial workflow applications. The primary reason is that, while ArcGIS Geoprocessing tools can be executed as Web services conveniently (especially the most recent release 10), many geospatial processes are data intensive and it is not efficient to stream huge amount of data as tokens within Java applets at the Web client side. Alternatively, we want to send the composed workflows for remote execution at the server side to reduce I/O costs. Towards this end, we have added a top-level button by modifying a class called RunnableGraphController (under ptolemy.vergil.basic package) in Ptolemy II. A class called RemoteExecAction is also created by extending FigureAction class to respond to the triggering of the button. After users have completed a workflow and submit it for remote execution, the workflow is exported to a string in MoML format and sent back to our middleware by opening an URL connection. The execution results are also written to the URL connection. If the execution is successful, the URL of an HTML page summarizing execution results will be sent back to

Web clients. Subsequently users can follow the links in the summary page for further visualization and exploration. We note it is possible to launch a new browser window within the workflow GUI environment. Fig. 5 illustrates the interactions among different components during the process. It is obvious that the development of workflow GUI is the key to the success of our fast system prototyping. Fortunately, Ptolemy II provides a solid basis for the extension.

4.3 Executing Geospatial workflows using Geoprocessing Tools

After the middleware receives the workflow in MoML representation, it can call Ptolemy II APIs to parse the MoML documentation back to internal workflow representation without any GUI involvement. For each director (e.g., SDF), Ptolemy II can compute an execution schedule by simply call the getScheduler() function of the director. While it is possible to simply execute the workflow within Ptolemy II if ArcGIS Geoprocessing tools are wrapped as executable actors, we have decided to develop our own execution module for the following practical considerations. First, there is a semantic mismatch between the definition of inputs and outputs in Ptolemy II and ArcGIS Geoprocessing tools. In ArcGIS Geoprocessing tools, names (including physical file paths) of data files, instead of geospatial data themselves, are used as inputs. As such, the names of output data files are considered as inputs rather than outputs. Literally there are no output parameters in ArcGIS Geoprocessing Tools (but there are output files). We could have designed actors in this way by following the interpretation strictly. However, the logics of processing pipelines are rather obscure in this case. Second, in order to execute actors within Ptolemy II, we would have to wrap ArcGIS Geoprocessing tools as Ptolemy II actors physically (note that we have only exposed the interfaces of ArcGIS Geoprocessing tools as conceptual actors as described in Section 4.1). It is highly inefficient to initialize ArcGIS Arcobjects engine every time a wrapped Geoprocessing tool is invoked. It would be much more efficient if the initialization is performed only once per workflow.

```
<entity name="Intersect" class="ArcGISActor">
        <property name="desc" class="ptolemy.kernel.util.StringAttribute" value="… ">
                <property name="style" class="ptolemy.actor.gui.style.LineStyle"/>
        </property>
        ……
        <property name="output_type " class="ptolemy.data.expr.StringParameter" value="INPUT">
                <property name="style" class="ptolemy.actor.gui.style.ChoiceStyle">
                        <property name="C0" class="ptolemy.kernel.util.StringAttribute" value="INPUT"/>
                        <property name="C1" class="ptolemy.kernel.util.StringAttribute" value="LINE"/>
                        <property name="C2" class="ptolemy.kernel.util.StringAttribute" value="POINT"/>
                </property>
        </property>
        <port name="features" class="ptolemy.actor.TypedIOPort">
                <property name="input"/>
                <property name="multiport"/>
                <property name="type" class="ptolemy.actor.TypeAttribute" value="string"/>
                <property name="desc" class="ptolemy.kernel.util.StringAttribute" value="A list of the input feature classes or
layers">
                        <property name="style" class="ptolemy.actor.gui.style.NotEditableLineStyle"/>
                </property>
                <property name="_showName" class="ptolemy.kernel.util.SingletonAttribute"/>
        </port>
        <port name="out_feature_class" class="ptolemy.actor.TypedIOPort">
                <property name="output"/>
                <property name="type" class="ptolemy.actor.TypeAttribute" value="string"/>
                <property name="desc" class="ptolemy.kernel.util.StringAttribute" value="The feature class to which the
results will be written">
                        <property name="style" class="ptolemy.actor.gui.style.NotEditableLineStyle"/>
                </property>
                <property name="_showName" class="ptolemy.kernel.util.SingletonAttribute"/>
        </port>
        …
</entity>
```

Fig. 3 MoML representation of the Intersect tool interfaces (with excerpt for clarity)

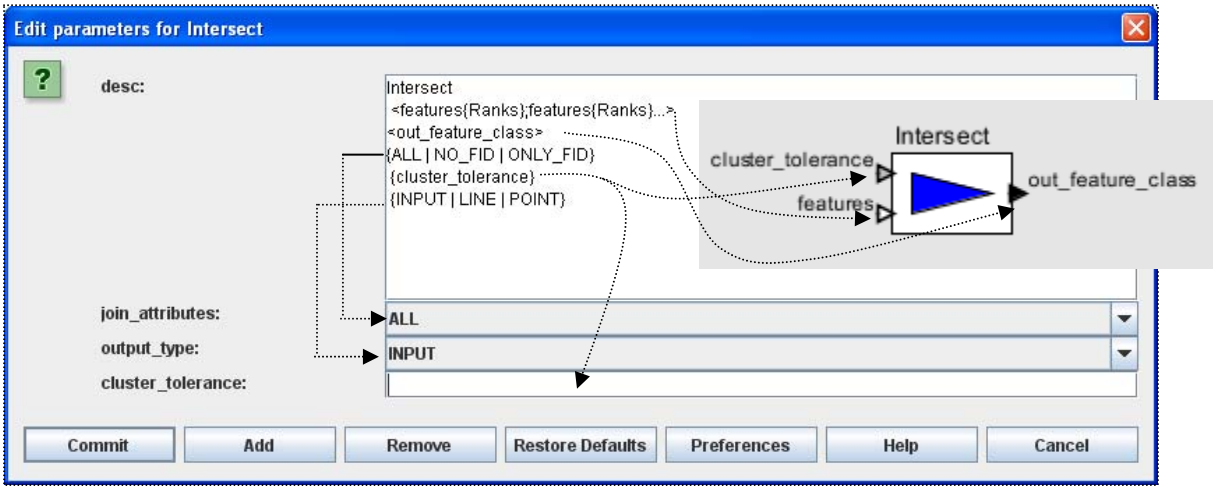

Fig. 4 Illustration of the GUI of the configuration dialog for the Intersect Actor

While our design is more efficient for geospatial workflows that require multiple geoprocessing tools and is more intuitive in understanding the workflow logics (see more discussion in Section 5), we need to propagate output values of a prior actor to the input ports of the connecting actors according to the actor execution plan provided by Ptolemy II. While this sounds trivial, we have found that the implementation needs some careful thoughts. First, as discussed in Section 3, Ptolemy II workflow graph model does not connect ports directly. Instead, the edges of graphs are actually the connections between ports and relations. Although this makes it possible to send a same output data token to multiple input ports based on some well-founded formalism, it is nontrivial to program the token propagation mechanism when programming from scratch. To do so essentially requires simulating Ptolemy II workflow execution process outside the system. Second, the output data file names, which are now represented as the values of output parameters, should be derived from inputs according to Ptolemy II. However, in geospatial modeling practices, these output data file names are either provided by users or assigned to unique identifiers automatically generated by a system and have nothing to do with inputs. The semantic mismatch needs to be addressed properly.

Our solution to the first issue relies on a hash table data structure. For each actor in the schedule to be executed, we find its input ports. We look up the value of all the input ports in the hash table by using the combination of the actor name and its port name as the key. Once all the values of the mandatory parameters are retrieved and validated and the values of non-mandatory parameters are checked and updated, the Geoprocessing tool corresponding to the name of the actor can be executed. Upon the successful completion of execution, we need to send the values of the output ports to the input ports of connecting actors for next steps of executions. The procedure is as the following. First, for each of the output port of the actor, we retrieve its connecting input ports of the next actor in the workflow graph. We then add entries to the hash table by using the combinations of the name of the connecting actor and the names of the input ports of the connecting actors as the key and the value that are supposed to send to the output port of the current actor as the value. We note that there are source actors (with no input ports) and sink actors (with no output ports). As the execution of a workflow begins with source actors and ends with sink actors, the above hash table looking processes are guaranteed to be successful.

Our solution to the second issue relies on a novel use of PortParameter in the context of geospatial workflow composition. By adding input ParameterPort(s) for actors that need to specify output file names, users are allowed to provide constant string values or to provide expressions that can be evaluated to string values in runtime through the actor configuration interface (c.f. Fig. 4). The string values representing output file names can also be provided by actors that output string values through their output ports when these ports are connected to the ParameterPorts. A unique file name will be generated if the PortParameter is not configured and its ParameterPort is not connected.

4.4 Developing a Web-GIS to visualize workflow execution results

As workflows are executed remotely in our prototype system, it is important to have the capabilities to visualize the original, intermediate and final results in a Web environment conveniently. Using a Web-GIS for this purpose is a natural solution. As shown in Fig. 5, our design enables visualization and explorations through a dynamically generated HTML page that summarize a workflow execution. The relevant file paths are mapped to URLs in the HTML page. When the URLs are accessed at the client side, the corresponding data will be served by a WebGIS and visualized in browser windows. While it is natural to use ArcGIS Server as the Web-GIS for this purpose, we have found that automatically publishing Web services for the dynamically generated geospatial datasets in an ArcGIS Server is more difficult than we have expected. To speed up the development process, we have decided to use the open source MapServer and OpenLayers instead.

It is fairly straightforward to publish geospatial data as OGC Web services in MapServer and use OpenLayers to

visualize these services, provided that necessary metadata of the geospatial data are already defined in MapServer map configuration files. We have made such publishing and visualization process dynamic by using MapServer PHP APIs and the key-value mechanism supported by the HTTP GET protocol. Basically we append layer names to URLs to be visualized while they are being generated by geoprocessing tools (as described in Section 4.3) so that the layer names can be passed to OpenLayers Javascript code when OGC Web services requests are being formulated. When the OGC Web services requests are processed by a MapServer PHP script at the server side, the layer names are then parsed from the requesting URL to the PHP script. Subsequently a dynamic MapServer layer is created by the PHP script. After the dynamic layer is created, all OGC Web services requests can be served in the same way as if the layers are statically set.

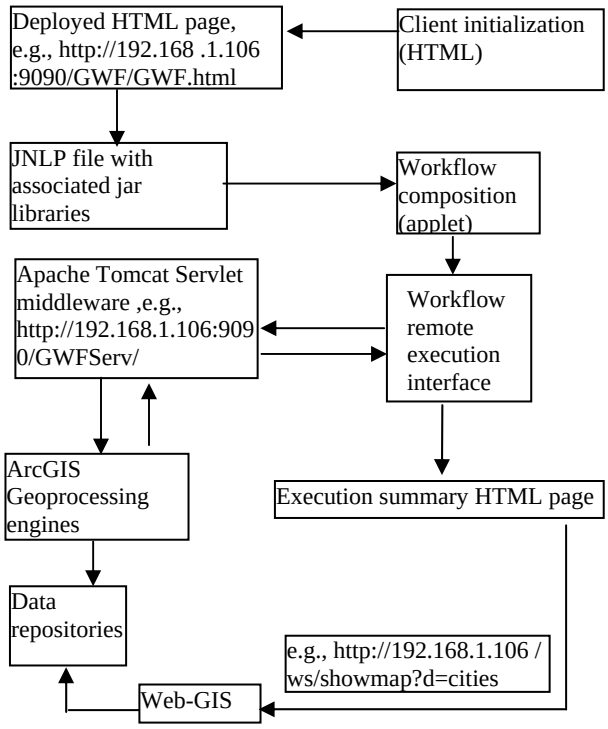

Fig. 5 Illustration of component interactions in our prototype system

5 DEMONSTRATIONS AND EVALUATIONS

A snapshot of the Java applet window for geospatial workflow composition is shown in Fig. 6. Under the "actor library" list at the top-left of the figure, actors representing four ArcGIS processing tools, i.e., Buffer, Union, Intersect and Clip, are used for demonstration purposes. A typical site selection geospatial processing workflow using the four actors/tools has been composed. The workflow works as the following. Given a point dataset representing the sites of thermal springs (sthermals), we wish to find suitable places to build facilities that are not only close to the thermal springs, but are also close to either cities (scities) or major roads (sroads). As such, we build a 10-mile (16000 meters) buffer around each thermal spring and a 5-mile (8000 meters) buffer around cities and roads. The buffers of the cities and roads are unioned before intersecting with the buffers of thermal springs. Finally we clip the intersection result by the county boundary dataset (scounty) to highlights the suitable places that are within a set of preselected counties. The names of the original input datasets and the buffer sizes are specified by using a StringConst actor. In addition, the top-level button to execute composed geospatial workflows is highlighted at the top of Fig. 6. The workflow execution summary page and the Web-GIS interface to visualize the final result (i.e., dataset var5) are shown in Fig. 7. To put the result into context, we also display the map of the original datasets at the right side of Fig. 7. While our Web-GIS application currently still lacks the capability of conveniently customizing layer combinations and symbolization (as a typical desktop GIS does), we have planned to improve the functionality in the future.

We next provide some preliminary evaluations on the prototype system we have developed. As we have discussed previously, we have adopted a fast prototyping deign principle by reusing and integrating existing software components. Due to the maturity and robustness of Ptolemy II workflow composition and scheduling infrastructure, compositing and editing geospatial workflows are intuitive, highly interactive and user friendly. Using the workflow system essentially eliminates the needs to program ArcGIS directly. While programming ArcGIS Geoprocessing tools is relatively straightforward due to the coarse-grained design of the tools, it is non-trivial to initialize Arcobjects and get through license validation process programmatically which involve considerable technical hurdles as we have experienced. More importantly, our workflow composition system can run over the Web from any web browsers that support Java applet. In contrast, as of the time of writing, ArcGIS ModelBuilder can only run within ArcGIS desktop environment. We believe our prototype system is the first to facilitate visual chaining ArcGIS Geoprocessing tools over the Web and provide a similar functionality of desktop based ModelBuilder.

Despite the close relationship between our prototype system and ArcGIS ModelBuilder, we next discuss a few differences between the two. We argue that our system has certain advantages over ArcGIS ModelBuilder. For comparison purposes, we have built the same site selection workflow in ArcGISBuilder which is shown in Fig. 8. From the figure, it is clear that the inputs and outputs of processing units and the data communications in ModelBuilder are not as formally modeled and represented as in Ptolemy II. All the inputs and outputs of a processing tool in ArcGIS need to be configured in a dialog associated with the tool. The connections between the outputs of a tool and their connecting tool(s) are just "symbolic" links for visualization purposes. For example, in the left part of Fig. 9, while the file names should be combined to form a single input parameter according to the syntax of the Intersect tool (c.f. Fig. 2), they are represented as two separate inputs in ArcGIS ModelBuilder. On the contrary, Ptolemy II faithfully represents the ArcGIS tool syntax as shown in the right part of Fig. 9. The process of combining multiple output ports to a single input port is through the multi-port mechanism that has been formally defined in Ptolemy II. During workflow composition, while ArcGIS ModelBuilder requires connecting a data source to a

Geoprocessing tool followed by selecting a parameter among all the parameters of the tool, Ptolemy II allows to connect ports directly, which, based on our evaluations, is more intuitive. While ModelBuilder in ArcGIS 10 has been improved significantly, we hope our evaluations can help further improve its functionality and usability.

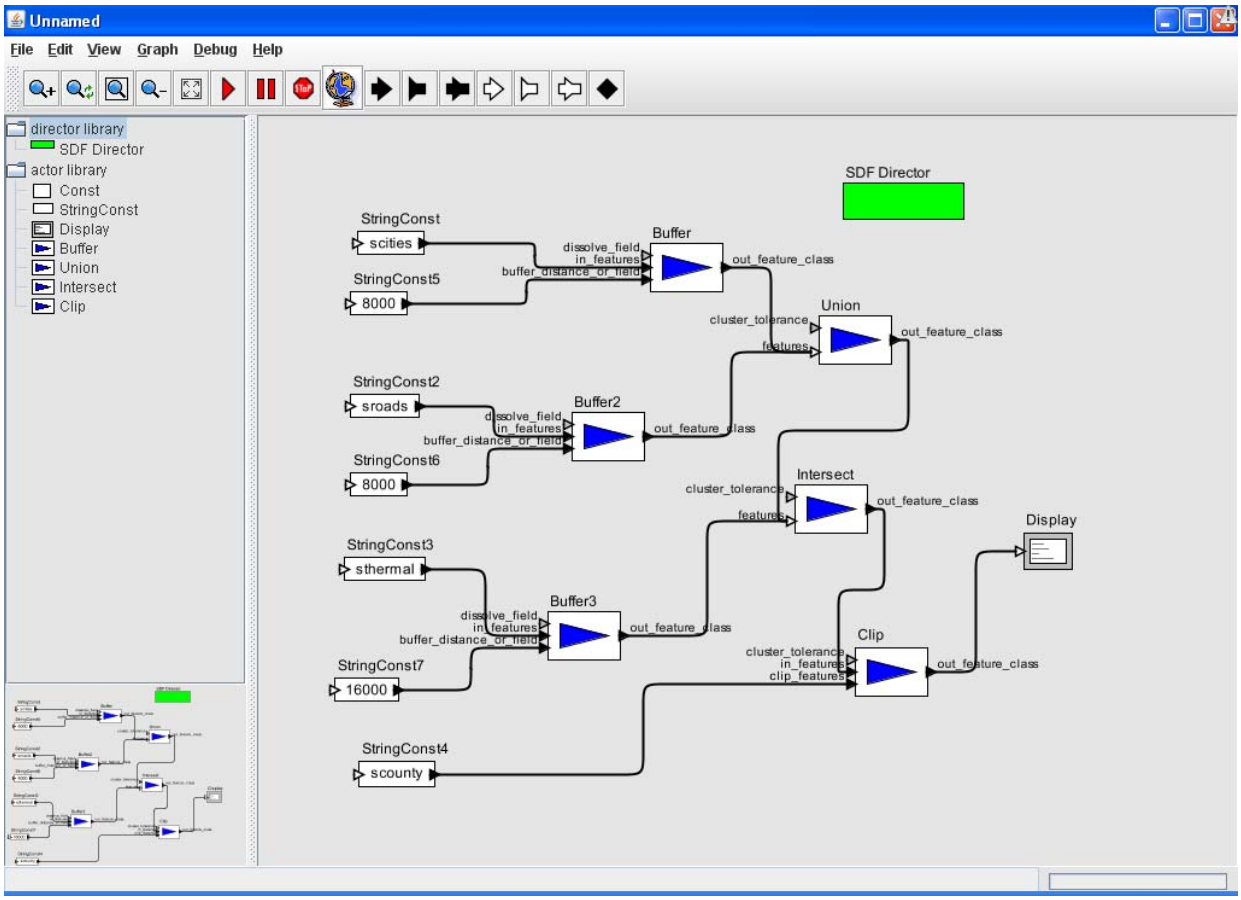

Fig. 6 Snapshot of the workflow composition environment (with remote execution trigger at top)

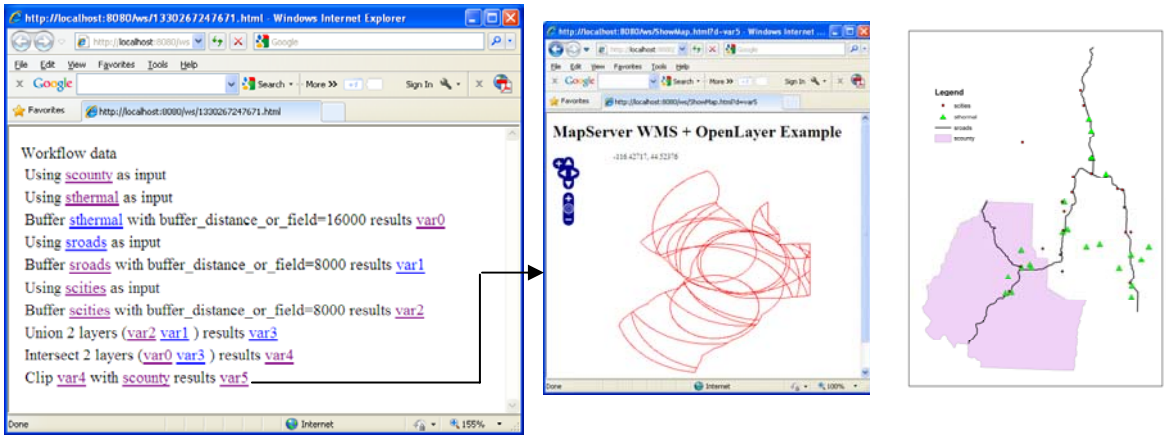

Fig. 7 Snapshots of workflow execution summary page and Web-GIS visualization interface (with overlaid original data displayed on the right)

There are plenty room left for improvements for our prototype system as well. First of all, we currently use a very preliminary mapping convention between the input data file names used in workflow compositions and the data files residing on remote servers. It would be much more useful if users are allowed to drag-and-drop a data source actor to select proper datasets by interacting with the data source actor in a way similar to the EMLDataSource actor that we have used in the Kepler workflow system [23]. As the resulting data source in this case can be any addressable URI instead of plain file names residing on the same sever hosting the middleware and ArcGIS Geoprocessing tools, it becomes possible to advance our prototype system to fully utilize distributed computing environments. Second, our Web-GIS application is very preliminary at present. While we plan to allow visualizing multiple layers simultaneously and customizing the symbolizations of the layers on the flay using open source GIS, we are also investigating the possibilities of dynamically publishing geospatial data as ArcGIS Web services and visualize the data using standard ArcGIS Server Web APIs.

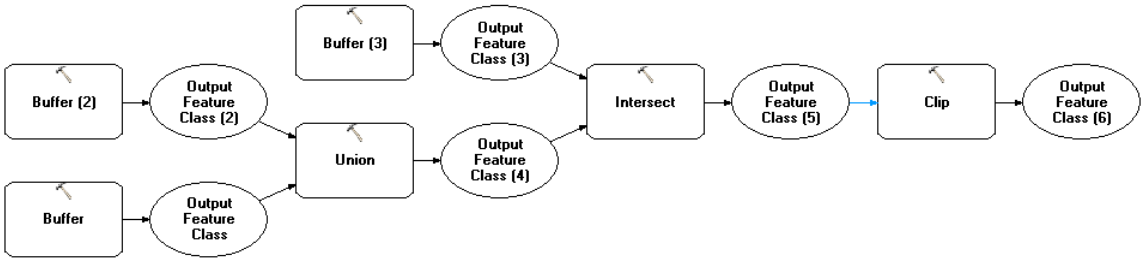

Fig. 8 Snapshot of the example geospatial processing workflow in ArcGIS ModelBuilder

Fig. 9 Comparisons of modeling inputs and outputs of processing units in ArcGIS ModelBuilder (left) and Ptolemy II (Right)

7 CONCLUSION AND FUTURE WORK

Motivated by the reality that a Web-based geospatial workflow composition system is missing to use ArcGIS Geoprocessing tools conveniently in a way similar to ArcGIS desktop-based ModelBuilder, we have developed such a prototype system by reusing and integrating various software packages. Instead of developing the Web-based geospatial workflow composition system from scratch which may take years, our prototype is developed in weeks with many desirable features including Web-enabled, highly usable and robust. A site selection example has been used to demonstrate the utilization of the prototype system. With the ability to fully utilize the geospatial processing power of the most popular commercial GIS system in a workflow environment, our initial vision on the scientific workflow approach to distributed geospatial data processing [21] is a step further to make the vision a reality.

For future work, in addition to what have been discussed in the evaluations, we plan to apply the prototype system for online collaborative modeling in multiple disciplines, including social-economic analysis of water resources and trip purpose identification from large-scale taxi trips. We also plan to add ontology-based semantic validation to the prototype by extending our previous works on geospatial workflow validations in the desktop Kepler scientific workflow system [42].

REFERENCES

1. ESRI ArcGIS ModelBuilder. http://help.arcgis.com/en/arcgisdesktop/10.0/help/
2. ERDAS Imagine Spatial Modeler. http://www.erdas.com/service/support/SpatialModels/Spatial_Models.aspx
3. OGC Web Services Standards. http://www.opengeospatial.org/standards/common
4. W3C Web Services Description Language (WSDL) standard. http://www.w3.org/TR/wsdl
5. Rich Internet Application. http://en.wikipedia.org/wiki/Rich_Internet_application
6. Eclipse BPEL Designer. http://www.eclipse.org/bpel/
7. Ptolemy Project. http://ptolemy.berkeley.edu/ptolemyII/
8. Apache Tomcat. http://tomcat.apache.org/
9. MapServer. http://mapserver.org/
10. OpenLayers. http://openlayers.org/

11. Yu, J. and R. Buyya (2005). A taxonomy of scientific workflow systems for grid computing. ACM SIGMOD Record 34(3): 44-49.
12. Davidson, S. B. and J. Freire (2008). Provenance and scientific workflows: challenges and opportunities. Proceedings of ACM SIGMOD Conference. 1345-1350
13. Deelman, E., D. Gannon, et al. (2009). Workflows and e-Science: An overview of workflow system features and capabilities. Future Generation Computer Systems 25(5): 528-540.
14. Kepler Scientific Workflow System. http://www.kepler-project.org/
15. VisTrail Scientific Workflow System. http://www.vistrails.org
16. Triana Problem Solving Environment. http://www.trianacode.org/
17. Taverna Workflow Management System. http://www.taverna.org.uk/
18. Business Process Execution Language (BPEL). http://en.wikipedia.org/wiki/Business_Process_Execution_Language
19. Medeiros, C. B. , G. Vossen, et al. (1995). WASA: A Workflow-Based Architecture to Support Scientific Database Applications. Proceedings of International Conference on Database and Expert Systems Applications (DEXA): 574-583.
20. Alonso,G. and C. Hagen (1997). Geo-Opera: Workflow Concepts for Spatial Processes. Proceedings of the International Symposium on Advances in Spatial Databases (SSD), LNCS, 1262: 238-258.
21. Jaeger, E., I. Altintas, et al. (2005). A Scientific Workflow Approach to Distributed Geospatial Data Processing using Web Services. Proceedings of the International Conference on SSDBM.
22. Zhang, J. T., D. D. Pennington, et al. (2005). Using web services and scientific workflow for species distribution prediction modeling. Proceedings of Advances in Web-Age Information Management (WAIM), LNCS 3739: 610-617.
23. Zhang, J., I. Altintas, et al. (2006). Integrating Data Grid and Web Services for E-Science Applications: A Case Study of Exploring Species Distributions. Proceedings of EEE International Conference on e-Science and Grid Computing.
24. Zhang, J. (2006). Tracking Dynamics of Geospatial Phenomena in Distributed and Heterogeneous Environments Using Scientific Workflow and Web Services Technologies. Proceedings of the Fifth International Conference on Grid and Cooperative Computing.
25. Pratt, A., Peters, C., Siddeswara, G., Lee, B., Terhorst, A. 2010. Exposing the Kepler Scientific Workflow System as an OGC Web Processing Service. iEMSs. Ottawa, Canada.
26. Migliorini, S., M. Gambini, et al. (2011). Workflow technology for geo-processing: the missing link. Proceedings of Com.Geo.
27. Yue, P., L. Di, et al. (2007). Semantics-based automatic composition of geospatial Web service chains. Computers & Geosciences 33(5): 649-665.
28. Han, W., L. Di, et al. (2008). Design and Implementation of GeoBrain Online Analysis System (GeOnAS). Proceedings of Web and Wireless Geographical Information Systems (W2GIS), Springer LNCS 5373: 27-36.
29. Chen, N., L. Di, et al. (2010). Geo-processing workflow driven wildfire hot pixel detection under sensor web environment." Computers and Geosciences 36(3): 362-372.
30. Yue, P., J. Gong, et al. (2010). GeoPW: Laying Blocks for the Geospatial Processing Web. Transactions in GIS 14(6): 755-772.
31. Sun, Z., P. Yue, et al. To Appear. GeoPWTManager: a task-oriented web geoprocessing system. Computers and Geosciences.
32. Hobona, G., D. Fairbairn, et al. (2007). Semantically-assisted geospatial workflow design. Proceedings of the ACM-GIS Conference.
33. Hobona, G., D. Fairbairn, et al. (2010). "Orchestration of Grid-Enabled Geospatial Web Services in Geoscientific Workflows." IEEE Transactions on Automation Science and Engineering 7(2): 407-411.
34. Java Web Start. http://en.wikipedia.org/wiki/Java_Web_Start
35. Tuot, C. J., M. Sintek, et al. (2008). IVIP --- A Scientific Workflow System to Support Experts in Spatial Planning of Crop Production. Proceedings SSDBM.
36. Bowers, S. and Ludäscher, B. (2005). Actor-Oriented Design of Scientific Workflows. International Conference on Conceptual Modeling (ER), LNCS, 3716: 369-384
37. Ludäscher, B., I. Altintas, et al. (2006). Scientific workflow management and the Kepler system. Concurrency and Computation: Practice and Experience 18(10): 1039-1065.
38. Lee, E. A. and Parks, T. M. (1995). Dataflow process networks. Proceedings of the IEEE, 83(5): 773-801
39. Zhao, Y., Y. Xiong, et al. (2010). The design and application of structured types in Ptolemy II. International Journal of Intelligent Systems 25(2): 118-136.
40. Zhang, J., D. D. Pennington, et al. (2005). Validating compositions of geospatial processing Web services in a scientific workflow environment. Proceedings of IEEE International Conference on Web Services (ICWS).
41. Zhang, J., Pennington, D. D., et al. (2006). Automatic Transformation from Geospatial Conceptual Workflow to Executable Workflow Using GRASS GIS Command Line Modules in Kepler. Proceedings of ICCS.
42. Zhang, J. (2006). Ontology-Driven Composition and Validation of Scientific Grid Workflows in Kepler: a Case Study of Hyperspectral Image Processing. Proceedings of GCCW.