Computers & Geosciences 47 (2012) 34–45

Contents lists available at SciVerse ScienceDirect

Computers & Geosciences

journal homepage: www.elsevier.com/locate/cageo

GeoPWTManager: a task-oriented web geoprocessing system

Ziheng Sun[a], Peng Yue[a,*], Liping Di[b]

[a] State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, 129 Luoyu Road, Wuhan 430079, China
[b] Center for Spatial Information Science and Systems (CSISS), George Mason University, 4400 University Drive, MS 6E1, Fairfax, VA 22030, USA

ARTICLE INFO

Article history:
Received 20 March 2011
Received in revised form
29 November 2011
Accepted 30 November 2011
Available online 8 December 2011

Keywords:
Task
Geoprocessing
Web service
Workflow
SOA
GIS

ABSTRACT

Recent advancement of geospatial services has shown great promise to solve complex geoprocessing tasks in a distributed environment. Geoprocessing services are often chained as scientific workflows and executed in a workflow engine. This paper proposes a task-oriented architecture for Web geoprocessing systems, which leverages Web service and workflow technologies to design and execute tasks, and monitor and visualize the execution of tasks. The approach facilitates the expression of users' requirements, allows the monitoring of the task execution, and hides the complexity of technical details. A prototype system, named GeoPWTManager, is implemented to demonstrate the applicability of the approach.

1. Introduction

Geoprocessing Web services aim to publish spatial analysis functions through Web service technologies, and workflow technology has been widely used to chain the scattered services (Kiehle, 2006). The Open Geospatial Consortium (OGC) has published a series of standards on geospatial Web services, including Web Map Service (WMS), Web Coverage Service (WCS), Web Feature Service (WFS), Web Processing Service (WPS), and Sensor Observation Service (OGC, 2007). There are already a number of investigations in developing and chaining geoprocessing services (Di, 2005a, 2005b; Brauner and Schaeffer, 2008; Yue et al., 2010).

The access to and chaining of geoprocessing services have shown great promise to solve complex geoprocessing tasks in a distributed environment (Jaeger et al., 2005; Yue et al., 2007; Allen et al., 2008; Chen et al., 2010). However, existing approaches are not task-oriented and expose too much technical details, for example the Kepler workflow system (Jaeger et al., 2005). To bridge the gap between users' requirement and distributed services, the concept of task is proposed (Vuong Xuan and Tsuji, 2009), which can facilitate the expression of users' requirements and hide the complexity of technical details.

This paper proposes a task-oriented architecture for Web geoprocessing systems, which leverages Web service and workflow technologies to design and execute tasks, and monitor and visualize the execution of geoprocessing tasks. It includes three functional components: task designer, task executor, and task monitor. A prototype system, named GeoPWTManager, is implemented to demonstrate the applicability of the approach.

The remainder of the paper is organized as follows. Section 2 describes the concept of task. One running example is introduced to help in understanding the concept. Related work is provided in Section 3. Section 4 describes a task-oriented architecture for Web geoprocessing systems, in particular how the three functional components work are described. Section 5 presents the implementation of the prototype system. Evaluation and discussion are provided in Section 6. Finally, conclusions and pointers to future work are given in Section 7.

2. The concept of task

The following example is used throughout the paper to help understand the role of task and how the task can be performed using distributed heterogeneous data and various geoprocessing services.

Supposing Mr. Li is a staff member in a disaster monitoring department in China, and he wants to know the situations of flood inundation in a region around the Poyang Lake in China in 2009 (Jiang et al., 2007; Sun and Yue, 2010). One satisfactory result would be a thematic image for the flood area, which renders regions that have different flood situations using different colors.

Mr. Li would like to formulate a geoprocessing task on the flood analysis, which can generate such an image. He finds that a geoprocessing model defined by Maathuis and Van Westen (2005) can perform the flood analysis and output the image (Fig. 1). The model uses six input images in three periods-before,

* Corresponding author. Tel.: +86 27 68778755.
E-mail address: geopyue@gmail.com (P. Yue).

0098-3004/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cageo.2011.11.031

in, and after the flood. In each period, there are two images in the first and second wavelength bands of MODIS data respectively. Each group of images goes through Normalized Difference Vegetation Index (NDVI) calculation, binarization, and rendering processes to gain the range of the flood in the same place in each period. Then, the three resultant images are mixed by a blend process to generate the required flood thematic image.

The task in the context of this paper reflects a user's demand, which can be achieved by a service's operation or a composition of service operations. A task that satisfies the requirement of Mr. Li can be described using the following information (Fig. 2):

(1) Task type: the task type describes the classification of tasks based on their functional properties. The flood analysis in the example is a kind of geoprocessing tasks.
(2) Task priority: it denotes the execution priority of the task. The value of this property determines the order on the allocation of geoprocessing resources.

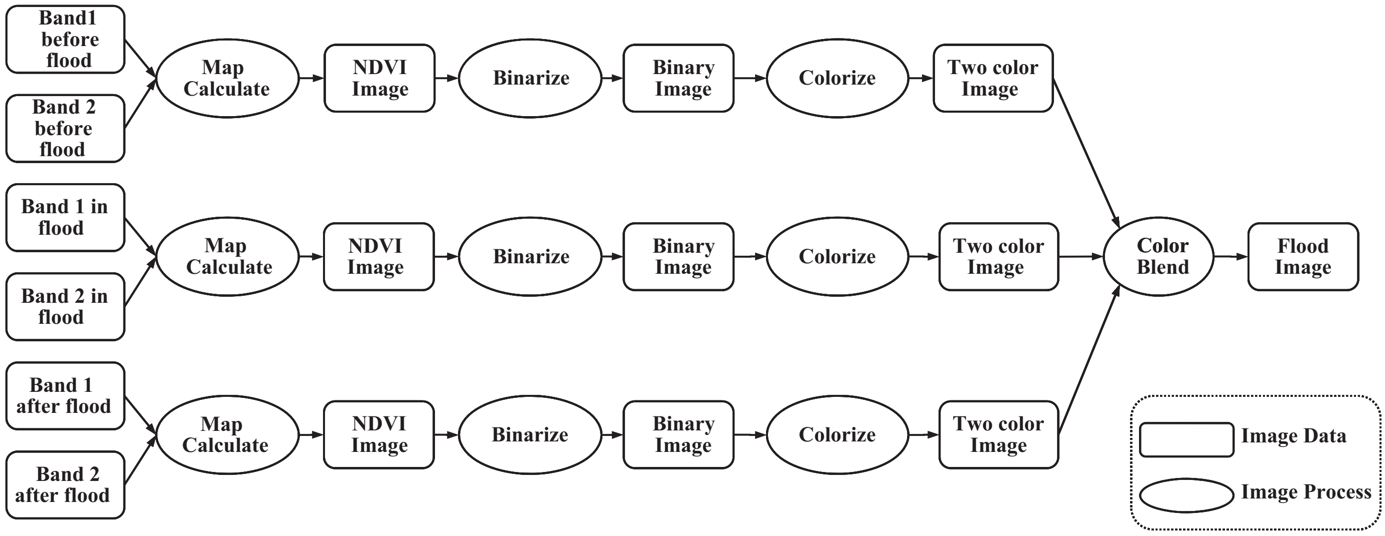


Fig. 1. The scientific model of the flood analysis.

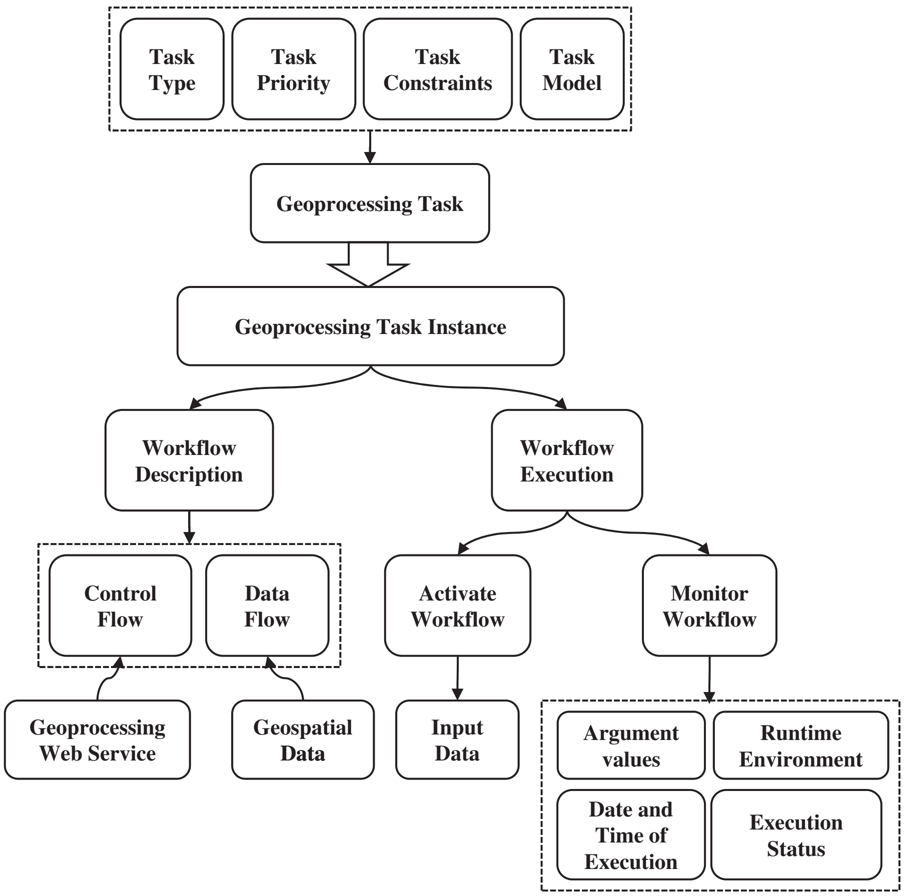


Fig. 2. The concept of the geoprocessing task.

(3) Task constraints: users may define metadata specifications that constrain the input or output data for the geoprocessing tasks. For example, Mr. Li specifies the spatial and temporal constraints of input data, and he wants an image as the output.
(4) Task model: the model constraints record the model that can accomplish the task. In the flood analysis example, the geoprocessing model provided by Maathuis and Van Westen (2005) is recorded as the model constraint.

The task can be executed using available geoprocessing services. The execution of tasks can be defined as task instances. A task instance includes the following information:

(1) Workflow description: it substantiates the task model in the task, and describes a collection of activities related by data and control flow relationship. The activities, in the context of this paper, are services providing different types of geoprocessing functions. However, they are not limited to services, and could be Application Programming Interfaces (APIs) providing local computing functions. Table 1 shows the Web services that are used in the flood analysis scenario. The description of workflows can use the existing workflow languages such as Web Services Business Process Execution Language (WSBPEL, shortly known as BPEL). A workflow itself can also be provided as a new activity or service (e.g., a BPEL service), which can be integrated with other workflows.
(2) Workflow execution: the runtime information of workflows is captured in this part. Such information includes the argument values (input or output data), the runtime environment (operating system, software library, hardware configuration, etc.), the date and time of the execution, and the execution status (start, in progress, end, etc.). Table 2 lists the input data used in executing the workflow in the example.

The separation of the task and the task instance advocates the expression of users' requirements at the task level and technical implementation at the instance level. The end users can focus only on the domain business logic instead of delving into the details on the background service technologies. The complexity of technologies is hidden.

3. Related work

A number of investigations have been devoted for building geoprocessing workflows. Zhao et al. (2006) create a model builder to create geoprocessing models, which can be transformed into executable BPEL workflows. The geoprocessing model is represented using a logical workflow language (Chen et al., 2009). The tool is implemented using a Java applet, which can be delivered to clients from a Web server and run in a Web browser. Scherp and Hasselbring (2010) specifies that the workflow model, executable workflow, and workflow execution can be defined as a model-driven framework for scientific workflows. Sun and Yue (2010) implemented a Web browser-based geoprocessing workflow designer, by integrating the Web 2.0 technologies and interoperable geoprocessing services. The work in this paper is an extension of the work in Sun and Yue (2010), by proposing the concept of the task to hide the technological details of workflows, and adding functions on monitoring the task execution. Friis-Christensen et al. (2009) investigate the architecture for distributed geographic information processing. In particular, they focus on the control flow and data flow patterns in the execution of a workflow.

Monitoring workflow execution is an important issue in the workflow management. There are already some approaches on monitoring and controlling business processes in the workflow management (Heloisa Martins and Tseng 1996). As Web services have been widely used in the workflow execution, the problem of the workflow execution monitoring has attracted much attention, and some solutions have been proposed. Baresi et al. (2010) discuss a set of approaches on monitoring workflow executions in the Web environment. The Service Centric Monitoring Language (SECMOL) is proposed, which can monitor BPEL processes. Our work transforms the low-level information from workflow execution monitoring into a user-friendly form in the context of task, i.e. the so-called task monitoring, which also prevents users from the details in the service technologies.

The idea of task was proposed in the domain of Geographical Information Systems (GISs) in the 1990s (Albrecht, 1994). Task is differentiated from GIS functions. The task describes "all actions that require human input or the knowledge about context", and GIS functions are concrete actions to implement the task (Albrecht, 1995, 1996, 1997). The work in this paper facilitates the implementation of tasks in a distributed service-oriented environment. In the general information domain, Chaolin et al. (2005) presented a method for distributing and managing tasks in the environment of the grid computing, and the task is the practical computing mission there. Others' work have investigated the separation of tasks and Web Services (Gorton and Reiff-Marganiec, 2006; Cardoso et al., 2004; Yu et al., 2007), in which business goals are achieved using tasks, and tasks are executed

Table 1
Services involved in the flood example.

Servicename	Short title	Service description
RasterMapcalcProcess	NDVI	Calculate NDVI from two input images, which are different bands of the same image.
RasterBinaryProcess	Binary	Binarize an image according to a specified threshold.
RasterColorsProcess	Color	Colorize an image with various scenarios.
RasterBlendProcess	Blend	Blends color components of two raster maps by a given ratio.

Table 2
Images involved in the flood example.

Image	Band	Date	Format	Size	Sensor	Extent
Image1	1	05/31/2009	GeoTiff	2.96 M	EOS MODIS	115.025E∼117.110E
	2			2.96 M		28.066N∼29.977N
Image2	1	08/01/2009		2.96 M		
	2			2.96 M		
Image3	1	08/27/2009		2.96 M		
	2			2.96 M		

using Web services. Our work focuses on the geospatial domain and enriches tasks with the domain business logic so that they can be performed by geoprocessing services.

In the Semantic Web area, a task ontology based the Web Ontology Language, shortly known as OWL-T, is proposed to depict the business task, and a prototype system has been developed to support the use of OWL-T (Vuong Xuan and Tsuji, 2007, 2009). Our future work will investigate the role of semantic tasks in automating the task formulation and execution.

4. Architecture

In this section, an architecture is proposed to realize the life cycle of a geoprocessing task. This design contains four modules: geoprocessing task designer, geoprocessing task executor, geoprocessing task monitor, and geoprocessing task results. They form a life cycle for geoprocessing tasks (Fig. 3).

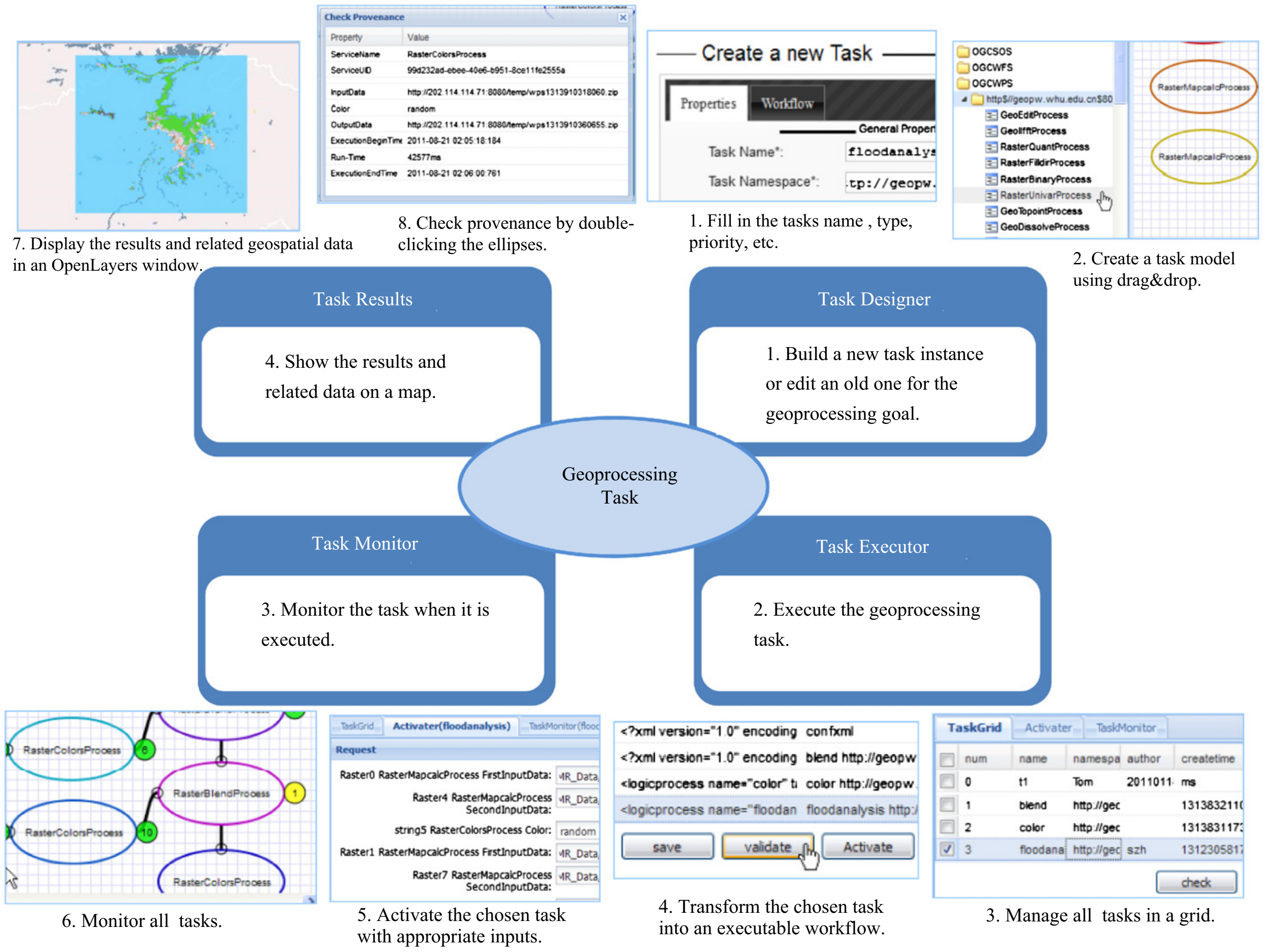
A detailed architecture is shown in Fig. 4. It shows the interactions among these modules. All these modules serve for the life cycles of geoprocessing tasks. A task, as the basic unit in this architecture, is produced by a geoprocessing task designer, performed by a geoprocessing task executor, supervised by a geoprocessing task monitor, and visualized by a geoprocessing task exhibitor. The detailed descriptions are provided as follows.

4.1. Geoprocessing task designer

This module's functionality is to provide users an interface to create a new geoprocessing task or edit an old one. Users specify the task type and task priority from predefined lists. The specific requirements on the input and output data are specified using the task constraints.

The task model can be either selected from existing model warehouses or be generated on demand. The proposal of the task model follows the model-driven approach. In the software engineering, the Model-Driven Architecture (MDA), proposed by the Object Management Group (OMG) in 2002, is an idea that makes the business information independent from the program and data, and the latter changes with the former (OMG, 2002). We take this method to separate the geoprocessing task model from the Web service, workflow technologies, and data. Such a method is flexible in that we can choose alternative services, workflow engines, or data to support the same model.

To be more specific, in the flood analysis example, Mr. Li expresses his requirements using the Extensible Markup Language (XML) schema, and the geoprocessing task designer can provide him a friendly interface to input the information of the geoprocessing task. When designing the task model, Mr. Li can drag and drop graphic components into a graphic panel to describe the model. Each component represents a specific type


7. Display the results and related geospatial data in an OpenLayers window.
8. Check provenance by double-clicking the ellipses.
Check Provenance
1. Fill in the tasks name , type, priority, etc.
Create a new Task
Properties
Workflow
General Proper
Task Name*: floodanalys
Task Namespace*: tp://geopw.
OGCSOS
OGCWFS
OGCWPS
http://geopw.whu.edu.cn:80
GeoEdfProcess
GeoIfftProcess
RasterQuantProcess
RasterFildirProcess
RasterBinaryProcess
RasterUnivarProcess
GeoTopointProcess
GeoDissolveProcess
RasterMapcalcProcess
2. Create a task model using drag&drop.
Task Results
4. Show the results and related data on a map.
Task Designer
1. Build a new task instance or edit an old one for the geoprocessing goal.
Geoprocessing Task
Task Monitor
3. Monitor the task when it is executed.
Task Executor
2. Execute the geoprocessing task.
RasterColorsProcess
RasterBlendProcess
6. Monitor all tasks.
TaskGrid
Activater(floodanalysis)
TaskMonitor(flood
Request
save
validate
Activate
5. Activate the chosen task with appropriate inputs.
4. Transform the chosen task into an executable workflow.
Activater
TaskMonitor
check
3. Manage all tasks in a grid.

Property	Value
ServiceName	RasterColorsProcess
ServiceUID	99d232ad-ebee-40e6-b951-8ce11fe2555a
InputData	http://202.114.114.71:8080/temp/wps1313910318060.zip
Color	random
OutputData	http://202.114.114.71:8080/temp/wps1313910360655.zip
ExecutionBeginTim	2011-08-21 02:05:18:184
Run-Time	42577ms
ExecutionEndTime	2011-08-21 02:06:00:761

Raster0 RasterMapcalcProcess FirstInputData: MR_Data
Raster4 RasterMapcalcProcess SecondInputData: MR_Data
string5 RasterColorsProcess Color: random
Raster1 RasterMapcalcProcess FirstInputData: MR_Data
Raster7 RasterMapcalcProcess SecondInputData: MR_Data

```
<?xml version="1.0" encoding   confxml
<?xml version="1.0" encoding   blend http://geopw
<logicprocess name="color" ti  color http://geopw
<logicprocess name="floodan    floodanalysis http:/
```

	num	name	namespa	author	createtime
☐	0	t1	Tom	2011011	ms
☐	1	blend	http://gec		1313832110
☐	2	color	http://gec		1313831173
✓	3	floodana	http://gec	szh	1312305817

Fig. 3. Life cycle of a geoprocessing task.

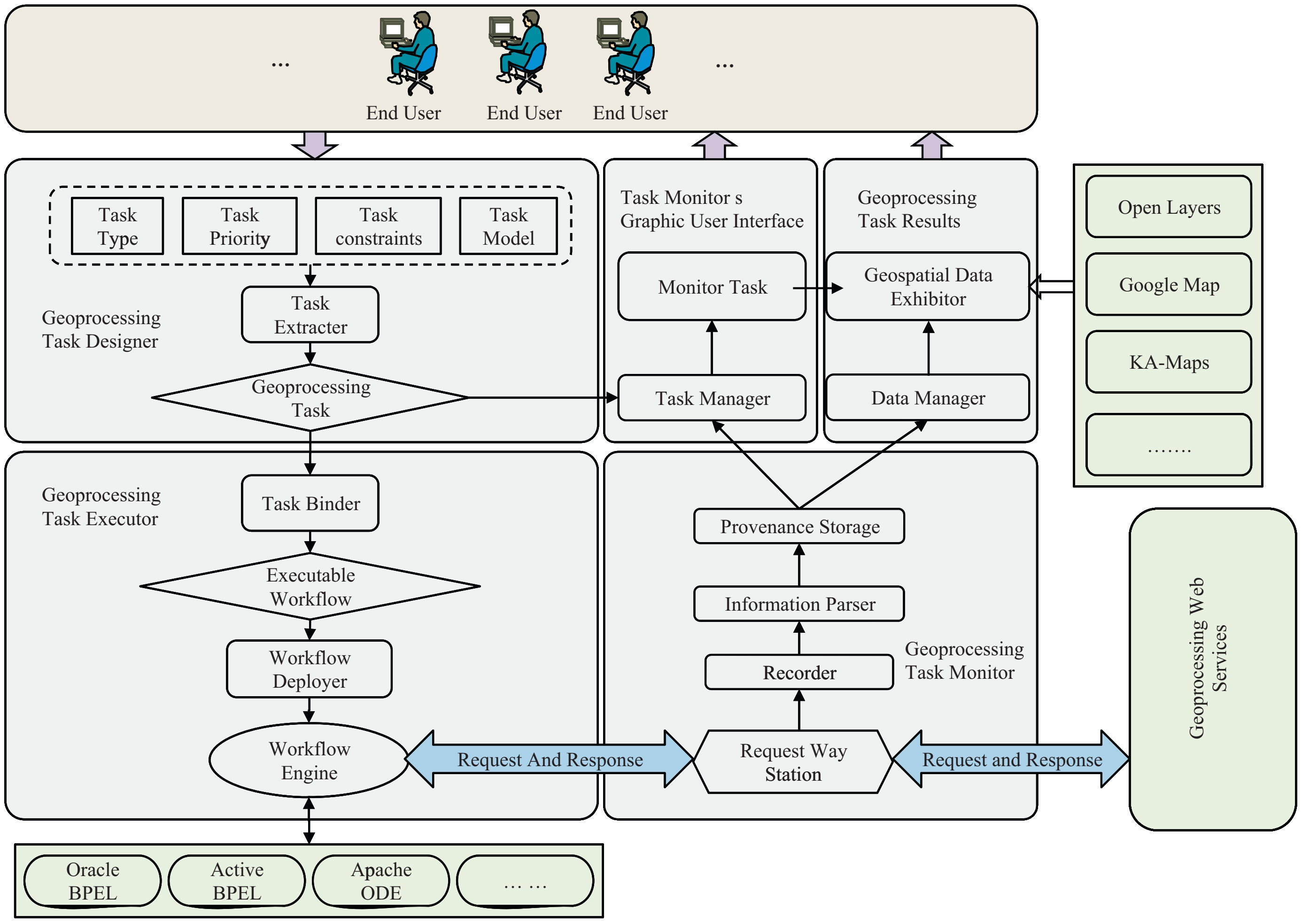


Fig. 4. Architecture for the management of a geoprocessing task.

of algorithms or functions. When connecting two components in a workflow, users are required to specify a data flow by connecting one input of the second component to the output of the first component. Finally, the model and other task information can be encoded in an XML file, describing task information on where, when, and how to geoprocess. The task model, i.e., the information on how to geoprocess, is the most important one, as it guides the transformation from task to executable workflow in the next section (Section 4.2).

4.2. Geoprocessing task executor

The geoprocessing task is transformed into an executable workflow using the geoprocessing task executor. A workflow can be characterized as a collection of activities with control and data flow relationships. The approach on mapping task model into an executable workflow includes the transformation of the control flow and data flow (Scherp and Hasselbring, 2010; Sun and Yue, 2010). Fig. 5 shows the process of transforming geoprocessing tasks into executable workflows. Each component in the task model is an abstract activity, which can be instantiated by a service. The order of services in the executable workflow follows the control flow in the task model. Services can be OGC services (e.g., WPS), W3C services, or RESTful services (Reed, 2009). Adapters for services need to be developed to support the mappings from activities to services. An adapter includes a request producer and a response parser. The request producer generates requests for the invocation of the service, and the response parser can parse the response of the service invocation. Fig. 6 shows how the output of the RasterMapcalcProcess is connected to the input of RasterBinaryProcess in generating the executable workflow, using BPEL as the workflow language. The adapter for the RasterMapcalcProcess generates a response expression by parsing the WPS response. This expression, according to the data flow, is assigned to the request body generated by the adapter of RasterBinaryProcess, as shown in Fig. 6. Thus, the connection between the input and output of dependent services are created in the executable workflow.

The final executable workflow may be executed using a mature business workflow engine. If the business workflow language is BPEL, mature engines include ActiveBPEL (now called ActiveVOS), Oracle BPEL (Juric and Krizevnik, 2010) (a product in the Oracle SOA Suite), and Apache ODE (Apache ODE, 2007). All these engines can receive a HTTP POST/GET request to deploy a new workflow and publish it as a value-added Web service, which can be visited by network users. People can send requests to this Web service to activate the workflow and get the desired results (OASIS, 2007). This mechanism separates users' attention from the execution of the workflow, and users need only ensure the correctness and availability of the input data. This mechanism prevents users from directly managing or monitoring the process of a workflow, as it wraps all the activities in a Web service that is exposed only at an invocation interface. A method will be proposed in the next module to solve this problem.

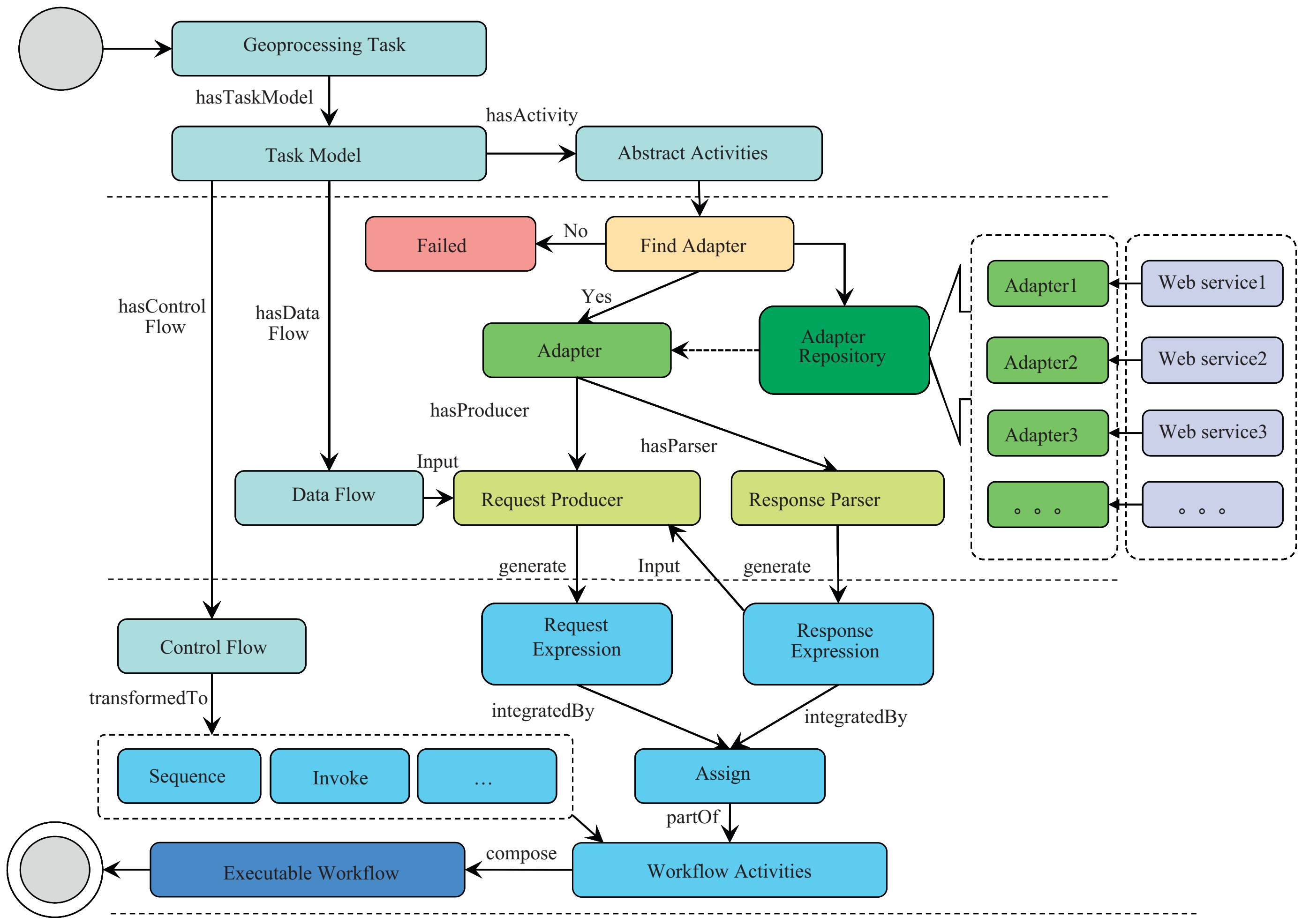


Fig. 5. The process of transforming geoprocessing tasks into executable workflows.

4.3. *Geoprocessing task monitor*

In order to let users monitor task execution, the status of workflow execution must be captured and used to update the status of the task execution. Most existing approaches on monitoring workflow execution suggest the insertion of monitoring activities in the workflow (Sun et al., 2009) (Fig. 7a). These monitoring activities capture the runtime information of service activities in the workflow, and send it to a certain listening application. As shown in Fig. 7a, two monitor activities are inserted into the business logic of the workflow. The MonitorProcess1 is used to collect the information of the Process1, while the MonitorProcess2 is used to collect the information of the Process2. Such approaches require the extensive changes to the business activities in the workflow.

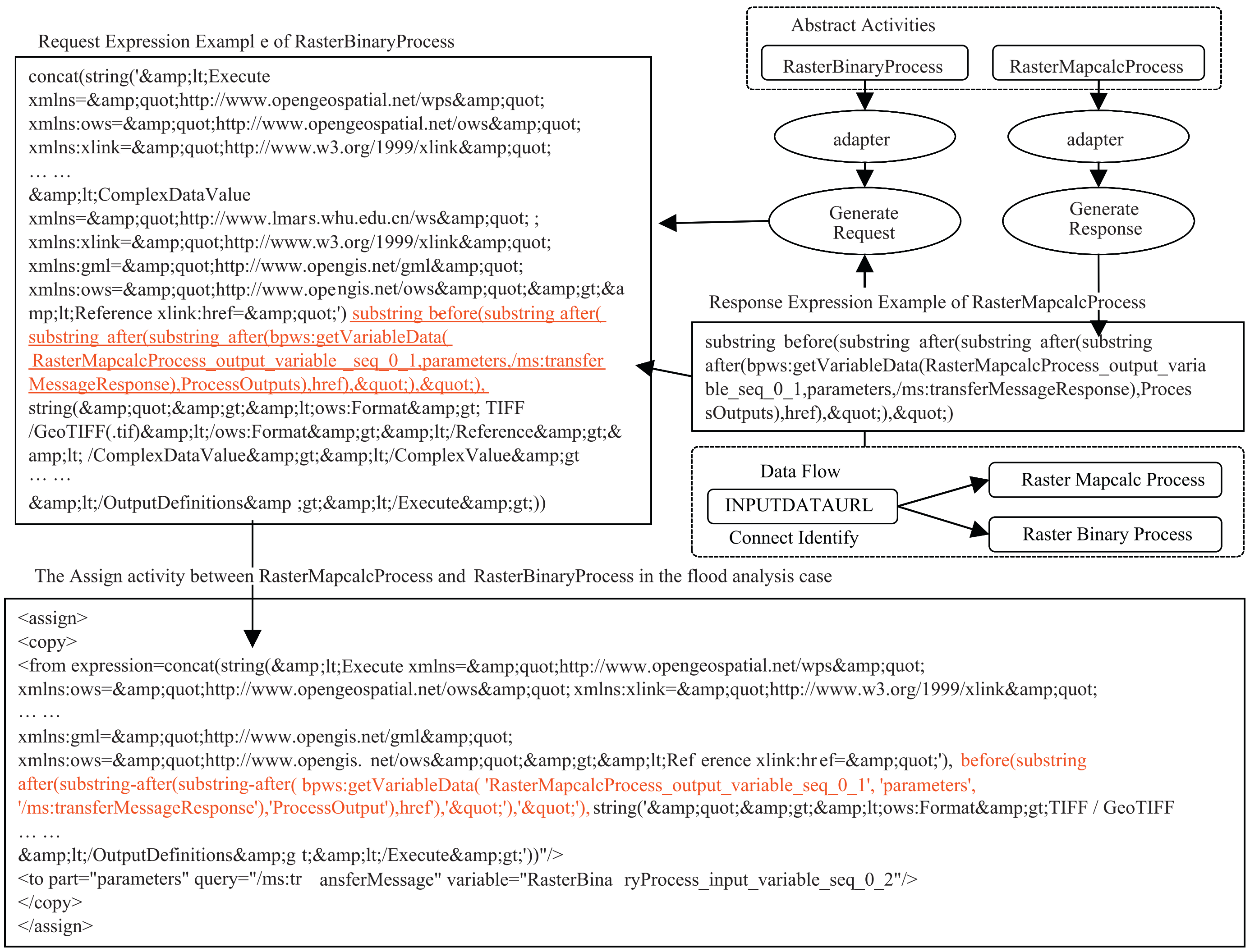
The work in this paper takes a different approach to monitor the workflow, by adding a request transfer station for the service invocation (Fig. 7b). The workflow engine sends all requests to the transfer station, which is responsible for collecting, analyzing, and storing requests before forwarding them to the target services, and then receiving, analyzing, and storing responses from the target services before sending them back to the engine. As shown in Fig. 7b, the monitoring logic is handled by the transfer station. One advantage of this method is that monitor activities are no longer needed to be entangled in the workflow logic. The change to the workflow is minimized and the monitor activity is more independent and manageable.

a. The conventional monitoring method.
b. The monitoring method.in this paper

The execution monitoring also helps to record the provenance. The provenance records the derivation history of the data products. For example, the request and response of each Web service, collected by the transfer station, can contribute the data products' provenance by linking themselves to the metadata for workflows and their executions. The emphasis of this paper is on the task-oriented architecture. For a detailed account on provenance, please refer to (Yue et al., 2011).

4.4. *Geoprocessing task results*

This module focuses on the visualization of the execution result of tasks. There are many tools available for the visualization of geospatial data, such as OpenLayers (OpenLayers, 2005), Google Maps (Google Maps API, 2005), KA-MAPS (ka-Map, 2007), CartoWeb (CartoWeb, 2005), and MapLab (MapLab, 2003). Most of these tools are open source software so that they can be easily integrated into new applications. The visualization of execution results can be loaded into these tools for evaluation.



Request Expression Exampl e of RasterBinaryProcess

```
concat(string('&amp;lt;Execute
xmlns=&amp;quot;http://www.opengeospatial.net/wps&amp;quot;
xmlns:ows=&amp;quot;http://www.opengeospatial.net/ows&amp;quot;
xmlns:xlink=&amp;quot;http://www.w3.org/1999/xlink&amp;quot;
… …
&amp;lt;ComplexDataValue
xmlns=&amp;quot;http://www.lmars.whu.edu.cn/ws&amp;quot; ;
xmlns:xlink=&amp;quot;http://www.w3.org/1999/xlink&amp;quot;
xmlns:gml=&amp;quot;http://www.opengis.net/gml&amp;quot;
xmlns:ows=&amp;quot;http://www.opengis.net/ows&amp;quot;&amp;gt;&a
mp;lt;Reference xlink:href=&amp;quot;') substring_before(substring_after(
substring_after(substring_after(bpws:getVariableData(
 RasterMapcalcProcess_output_variable _seq_0_1,parameters,/ms:transfer
MessageResponse),ProcessOutputs),href),&quot;),&quot;),
string(&amp;quot;&amp;gt;&amp;lt;ows:Format&amp;gt; TIFF
/GeoTIFF(.tif)&amp;lt;/ows:Format&amp;gt;&amp;lt;/Reference&amp;gt;&
amp;lt; /ComplexDataValue&amp;gt;&amp;lt;/ComplexValue&amp;gt
… …
&amp;lt;/OutputDefinitions&amp ;gt;&amp;lt;/Execute&amp;gt;'))
```

Response Expression Example of RasterMapcalcProcess

```
substring_before(substring_after(substring_after(substring
after(bpws:getVariableData(RasterMapcalcProcess_output_varia
ble_seq_0_1,parameters,/ms:transferMessageResponse),Proces
sOutputs),href),&quot;),&quot;)
```

The Assign activity between RasterMapcalcProcess and RasterBinaryProcess in the flood analysis case

```
<assign>
<copy>
<from expression=concat(string(&amp;lt;Execute xmlns=&amp;quot;http://www.opengeospatial.net/wps&amp;quot;
xmlns:ows=&amp;quot;http://www.opengeospatial.net/ows&amp;quot; xmlns:xlink=&amp;quot;http://www.w3.org/1999/xlink&amp;quot;
… …
xmlns:gml=&amp;quot;http://www.opengis.net/gml&amp;quot;
xmlns:ows=&amp;quot;http://www.opengis. net/ows&amp;quot;&amp;gt;&amp;lt;Ref erence xlink:hr ef=&amp;quot;'), before(substring
after(substring-after(substring-after( bpws:getVariableData( 'RasterMapcalcProcess_output_variable_seq_0_1', 'parameters',
'/ms:transferMessageResponse'),'ProcessOutput'),href'),'&quot;'),'&quot;'), string('&amp;quot;&amp;gt;&amp;lt;ows:Format&amp;gt;TIFF / GeoTIFF
… …
&amp;lt;/OutputDefinitions&amp;g t;&amp;lt;/Execute&amp;gt;'))"/>
<to part="parameters" query="/ms:tr   ansferMessage" variable="RasterBina  ryProcess_input_variable_seq_0_2"/>
</copy>
</assign>
```

Fig. 6. An example of the mapping between the input and output of activities.

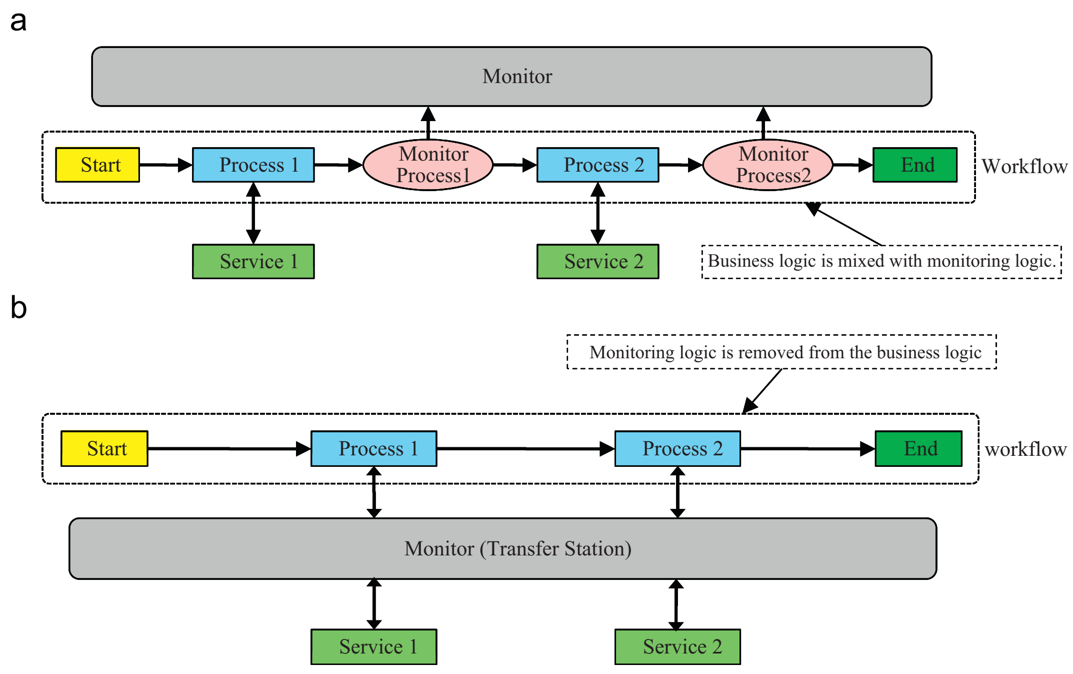


Fig. 7. The methods monitoring a workflow. The workflow includes two activities: Process 1 and Process 2, which are executed by Service 1 and Service 2 respectively.

In addition, the monitoring result of task execution can also be visualized in this module.

5. Implementation

We have developed a prototype system named GeoPWTManager[1] (also called GeoPWTaskManager) to implement the model. It uses a number of techniques and tools such as Asynchronous JavaScript (Ajax) and XML (Garrett, 2005), Java servlet (Coward and Yoshida, 2003), Web service, JSON (Crockford, 2001), ExtJS (ExtJS, 2008), RaphealJS (Baranovskiy, 2009), OpenLayers, ActiveBPEL Server, Apache Axis2 (Apache Axis2, 2004). It can be deployed on a server and visited by any Web users in the Web browsers.

The task designer provides a novel useful tool for users to drag, drop, and compose the geoprocessing Web services following the WPS standard, and get a desired logical geoprocessing workflow described in XML. This tool is implemented already (Sun and Yue, 2010) and used here to support the task management. Other complementary information on tasks is also recorded in XML. Once a new geoprocessing task is successfully created, it will be delivered to the task manager, which will take charge of all activities of the task including validation, execution, supervision, and result demonstrability. The workflow, encoded using XML, can be instantiated as an executable workflow expressed by BPEL, deployed on a running ActiveBPEL Server. The task manager can also activate the workflow deployed on the engine with valid inputs, and then supervise the execution and display the results on the Graphical User Interface (GUI) using OpenLayers.

As shown in Fig. 8, we have helped Mr. Li create a geoprocessing task for the flood analysis case, perform it, and get the initial required output in the end. The net of graphical ellipses connected with each other represents the geoprocessing Web services used in this geoprocessing task. The numbers in the small circles near the ellipses label the execution order of the geoprocessing Web services. The lower part is an OpenLayers window to exhibit the resultant image. The background is provided by a WMS service containing tiles showing the whole world. The image in the central part is the result, which uses different colors to show different situations. For example, green means the region is covered by water in all the three periods, but pink means that the region is covered by water on 31 May and 1 August, but not covered by water on 27 August. GeoPWTManager also supports users to check provenance of each task, and Mr. Li can inspect the data in every intermediate step by double-clicking the ellipses or lines in the page. Fig. 9 shows the intermediate processing results of images in the case.

6. Evaluation and discussion

In this section, the performance of the prototype system is evaluated (Section 6.1). The approach is compared with other workflow-based systems, and advantages of the task-oriented approach are discussed (Section 6.2).

6.1. Evaluation

To evaluate the system performance, we create four tasks (Table 3), in which two tasks use only individual Web services and the other two tasks use workflows composed of services. Each task is executed in twenty times. The time for both the task execution and services/chains execution are recorded each time. The difference between the two time values can represent the performance of GeoPWTManager itself.

Figs. 10 and 11 shows test results of the four tasks. The Difference1 and Difference2 in Fig. 10 show the time difference between task executions and service executions, while the Difference3 and Difference4 shows the time difference between task executions and executions of service chains. The following observations can be achieved:

(1) The execution time of services or service chains is the major factor affecting the performance of the task execution. For example, the blend service in Task1 takes more time than the color service in Task2, which makes the execution time of Task1 is much higher than that of Task2.
(2) The execution time is fluctuating for each execution of tasks or services/chains. This is due to the changing status of the network and processing environments.
(3) The execution time of Task3 and Task4 is near to the sum of the execution time for the contained Web services, it demonstrates that the execution time of complex tasks depends on the execution time of individual services in the service chain.
(4) The Difference1, Difference2, Difference3, and Difference4 are almost stable, and the Difference3 and Difference4 are a little higher than Difference1 and Difference2. The reason is that complex tasks take the workflow engine more time to run the corresponding service chains and transport the messages between the workflow activities. Since the main programs of GeoPWTaskManager, the client code in Web browser and the transfer Web service, have a small and stable cost, it means that the performance of the GeoPWTManager is almost not affected by different tasks.

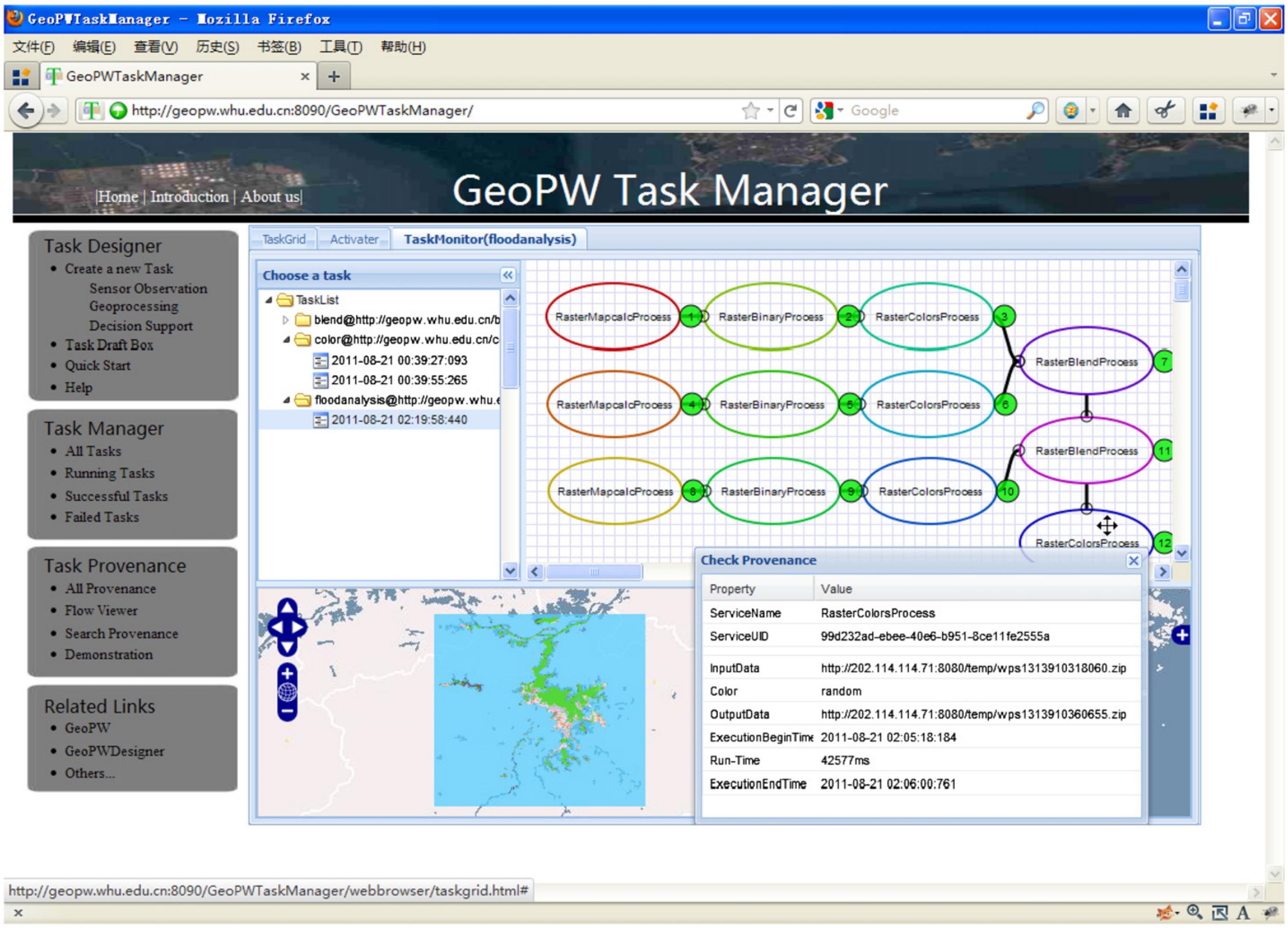
The GeoPWTManager provides a user-friendly environment to bridge the users' requirements and service/chains. The execution time of the GeoPWTManager is very low and can be ignored, compared to the execution time of services/chains. That means there is no decrease in performance when adding GeoPWTManager on top of services/chains. Therefore, the performance optimization of the GeoPWTManager is not the major concern of this paper. The performance of task executions can be improved mainly by the enhancement of underlying services and chains.

After GeoPWTManager is available to the public, a questionnaire is provided to collect the feedback from users. The users with a background in the geoinformatic domain are invited to use the system. Finally, 27 people submit their feedback. In the investigation, 92.6 percent of users agree that GeoPWTManager can help geoscientific problem solving. 77.8 percent of users state that they could potentially use GeoPWTManager in their scientific research. Over half of users think that the performance of GeoPWTManager is acceptable. 96.3 percent of users agree that the task-oriented design makes it easier to use geoprocessing Web services. 85.2 percent of users also think that an impressive advantage of GeoPWTManager is that it needs not be installed in their computers, and 55.6 percent of users state that the major disadvantage of GeoPWTManager is that the types of services supported in the system are not rich. Some additional suggestions will be addressed in the final section on the future work.

6.2. Discussion

There are already a number of workflow approaches on scientific problem solving (Dustdar and Schreiner, 2005; Jaeger et al., 2005; H. ter Beek, Bucchiarone et al., 2006). Here we list several representative approaches, and compare them with GeoPWTManager.

[1] GeoPWTManager online: http://geopw.whu.edu.cn:8090/GeoPWTaskManager.



Property	Value
ServiceName	RasterColorsProcess
ServiceUID	99d232ad-ebee-40e6-b951-8ce11fe2555a
InputData	http://202.114.114.71:8080/temp/wps1313910318060.zip
Color	random
OutputData	http://202.114.114.71:8080/temp/wps1313910360655.zip
ExecutionBeginTime	2011-08-21 02:05:18:184
Run-Time	42577ms
ExecutionEndTime	2011-08-21 02:06:00:761

Fig. 8. The task on flood analysis in GeoPWTManager.

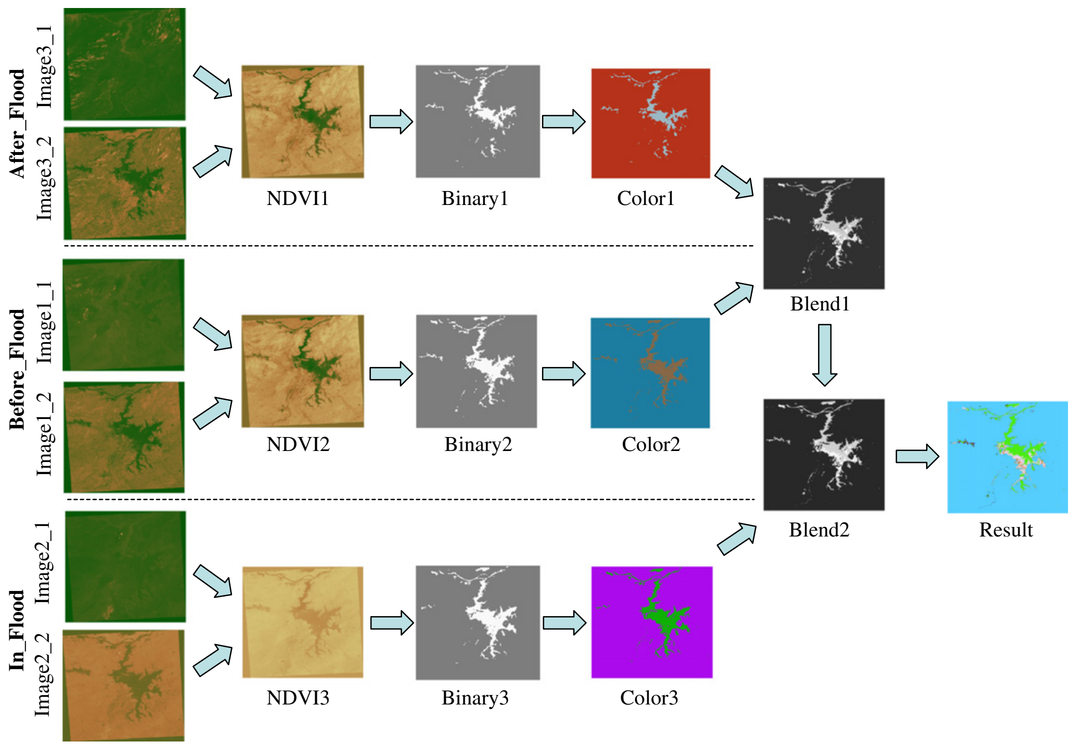
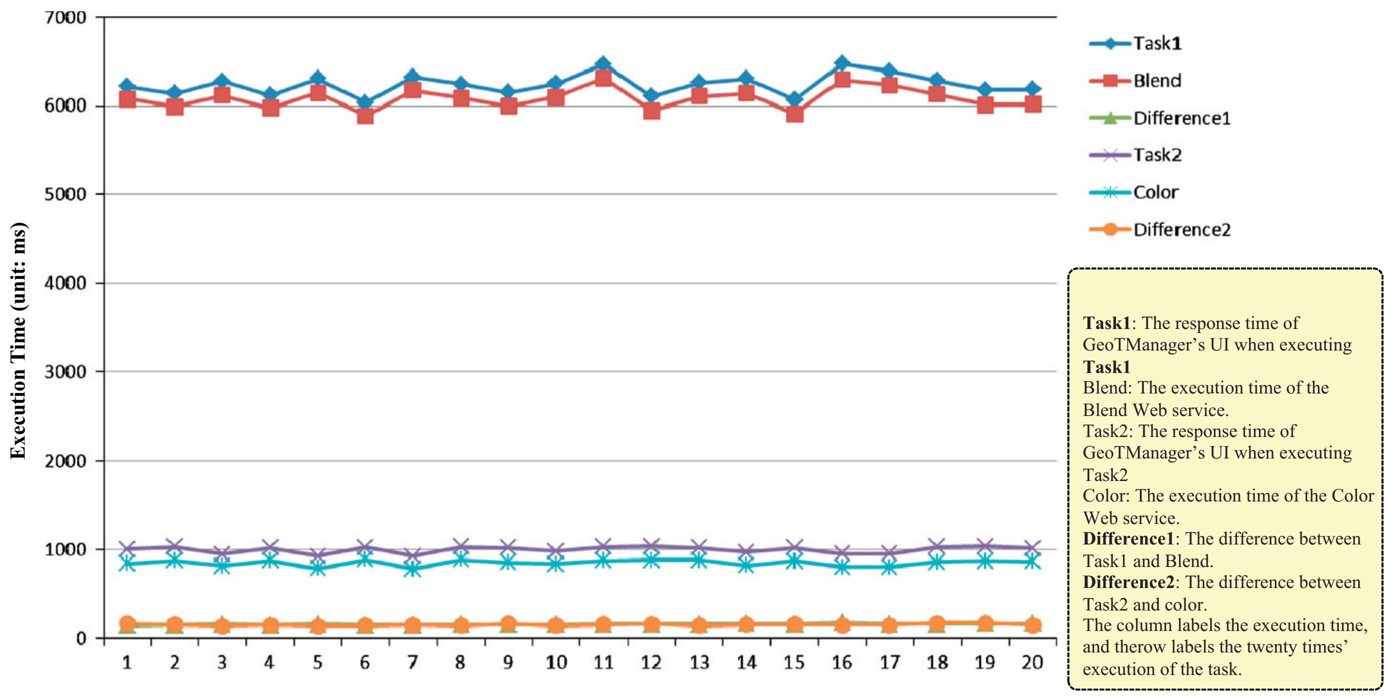


Fig. 9. The processing results in the flood analysis case.

Table 3
The four tasks and input data for the performance experiments.

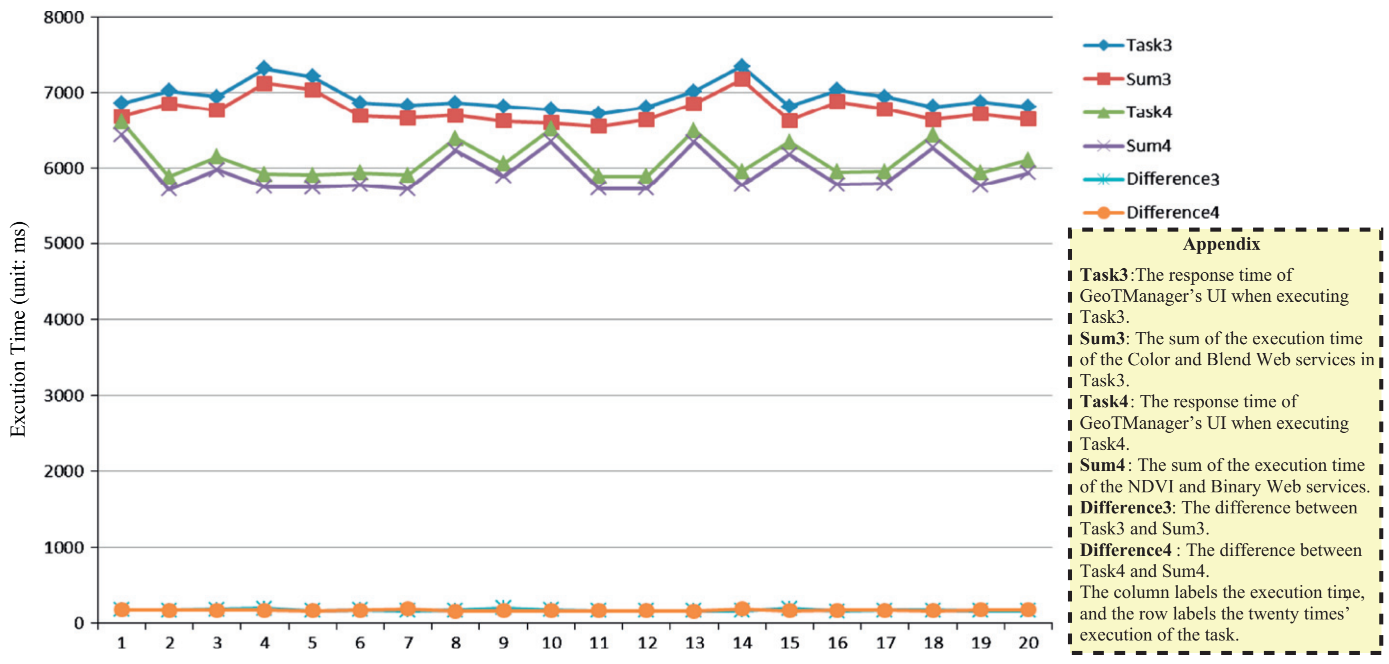
Task name	Web services	Input data (from Fig. 9)
Task1	RasterBlendProcess	Color1(817 KB), Color2(817 KB)
Task2	RasterColorsProcess	Binary1 (770 KB)
Task3	RasterColorsProcess → RasterBlendProcess	Binary1, Color2
Task4	RasterMapcalcProcess → RasterBinaryProcess	Image1_1(2.96 MB), Image1_2(2.96 MB)

The Business Process Modeling Notation (BPMN) is mainly used in business domain and can cover many types of modeling of business activities and flows (White, 2004). The BPMN models are not executable and must be instantiated (Ouyang et al., 2006). The workflow tools supporting BPMN include ARIS Express (ARIS Community, 2011) and ActiveVOS. Although both BPMN and tasks in this paper can be instantiated and executed by BPEL, the BPMN is designed for the business domain, while the task-oriented approach is designed for geospatial applications. BPEL is often regarded as an approach for static service composition. It relies on the Web Services Description Language (WSDL) and exposes



Task1: The response time of GeoTManager's UI when executing **Task1**
Blend: The execution time of the Blend Web service.
Task2: The response time of GeoTManager's UI when executing Task2
Color: The execution time of the Color Web service.
Difference1: The difference between Task1 and Blend.
Difference2: The difference between Task2 and color.
The column labels the execution time, and therow labels the twenty times' execution of the task.

Fig. 10. Performance tests of GeoPWTManager using Task1 and Task2.



Appendix
Task3:The response time of GeoTManager's UI when executing Task3.
Sum3: The sum of the execution time of the Color and Blend Web services in Task3.
Task4: The response time of GeoTManager's UI when executing Task4.
Sum4: The sum of the execution time of the NDVI and Binary Web services.
Difference3: The difference between Task3 and Sum3.
Difference4 : The difference between Task4 and Sum4.
The column labels the execution time, and the row labels the twenty times' execution of the task.

Fig. 11. Performance tests of GeoPWTManager using Task3 and Task4.

much technical details. GeoPWTManger leverages BPEL for the workflow execution but hides the syntactic details of BPEL in the background, which makes it easier for geospatial experts to understand and use geoprocessing Web services.

ArcGIS ModelBuilder is a geoprocessing tool in ArcGIS software. It can help compose spatial analysis functions in the ArcToolBox into a workflow in a user-friendly way (Manegold, 2003). However, ArcGIS ModelBuilder is a desktop-based tool, while GeoPWTManager can run in the Web browser. The workflow activities in the ArcGIS ModelBuilder are proprietary geoprocessing functions, while in GeoPWTManager workflow activities are open and interoperable geoprocessing Web services.

Kepler is a workflow system for the analysis and modeling of scientific data (Kepler, 2008). It is not domain-specific and supports the composition of both local and remote resources (e.g., through Web service invocation). Taverna is a similar scientific workflow system and can be used in the grid-computing environment (Taverna, 2009). However, these workflow systems are not ready for geospatial domain users. Significant development is required to support geoprocessing Web services. On the other hand, the task-oriented approach in this paper wraps the workflow technologies in the background and transforms geoscientific problem solving into the creation and execution of geospatial tasks. The concept of geoprocessing task facilitates the easy expression of users' requirement and improves the users' interaction with the Web geoprocessing system. The separation of task and task instance bridges the users' requirements and the underlying technologies. The end-users work in a task-oriented environment, where they can focus on their domain problems using knowledge in the task context. The complexity of technologies is hidden. In the background, the use of geoprocessing workflow is flexible since the task model can be bound to concrete workflows dynamically. Interoperable geospatial services can be easily plugged into the system using adapters. The task-oriented architecture allows the monitoring of geoprocessing workflows in this architecture. The monitoring method suits the workflow in SOA well, and is independent of workflow languages or engines. Therefore, the approach in this paper facilitates the expression of users' requirements, allows the monitoring of task execution, and hides the complexity of technical details. The Web characteristics of the tool do not require users to install software on their personal computers. In addition, users no longer need to handle complex XMLs or XML schemas when composing Web service. The approach makes users stay at a conceptual level and focus on domain problems. The Web-based designer interface, and task-oriented execution and monitoring environment are more user-friendly.

7. Conclusion and future work

This paper proposes a task-oriented architecture for Web geoprocessing systems, which leverages Web service and workflow technologies to design and execute tasks, and monitor and visualize the execution of tasks. A prototype system, named GeoPWTManager, is implemented to demonstrate the applicability of the approach. The system is available online and accessible in the Web browser. We evaluate the approach by designing experiments to test the performance of the prototype system, and compare the approach with other traditional workflow strategies. The approach facilitates the expression of users' requirement, allows the monitoring of task execution, and hides the complexity of technical details.

The current implementation of the GeoPWTManager supports only geoprocessing tasks. In the future, other types of tasks, such as an observation task on sensor planning or observing, will be supported. These tasks, connected with geoprocessing tasks, can help achieve live geoprocessing in a Sensor Web environment. The connections among these tasks can be formalized using an ontology-based approach. Ontology denotes a specification of the terms in a shared domain and relations among them (Gruber, 1993; Noy and McGuinness, 2001). A semantic approach using task ontologies will be added in the system to allow automatic task decomposition, geospatial knowledge reuse, and semantic interoperability. In addition, future implementation will include the incorporation of RESTful services and improvement of the service performance.

Acknowledgements

We are grateful to the anonymous reviewers for their comments. This work was funded jointly by National Basic Research Program of China (2011CB707105), Project 40801153 and 41023001 supported by NSFC, LIESMARS (Wuhan University) Special Research Fundings.

References

Albrecht, J., 1994. Universal elementary GIS tasks – beyond low-level commands. In: Proceedings of the 6th Symposium of Spatial Data Handling, Edinburgh. pp. 209–222.
Albrecht, J., 1995. Semantic net of universal elementary GIS functions. In: Proceedings, 12th International Symposium on Computer-Assisted Cartography (Auto-Carto12), Charlotte, NC, pp. 232–241.
Albrecht, J., 1996. Universal GIS-Operations: A Task-Oriented Systematization of Data Structure-Independent GIS Functionality Leading Towards a Geographic Modeling Language. ISPA, University of Vechta, Germany.
Albrecht, J., 1997. Universal analytical GIS operations: A task-oriented systematization of datastructure-independent GIS functionality. Geographic Information Research: Transatlantic Perspectives. Taylor and Francis, London pp. 577–591.
Allen, G., Bogden, P., Creager, G., Dekate, C., Jesch, C., Kaiser, H., McLaren, J., Perrie, W., Stone, G.W., Zhang, X., 2008. Towards an integrated GIS-based coastal forecast workflow. Concurrency and Computation: Practice & Experience 20, 1637–1651.
Apache Axis2, 2004. Apache Axis2. the Apache Software Foundation, 〈http://axis.apache.org/axis2/java/core/〉, (accessed 08.06.10).
Apache ODE. 2007. Apache ODE. the Apache software foundation. 〈http://ode.apache.org/〉, (accessed 23.06.10).
ARIS Community, 2011. A. ARIS Express. 〈www.ariscommunity.com/aris-express〉, (accessed 25.08.11).
Baranovskiy, D. 2009. Raphaël: a JavaScript API for SVG. 〈http://dev.opera.com/articles/view/raphael-a-javascript-api-for-svg/〉, (accessed 27.11.10).
Baresi, L., Guinea, S., Nano, O., Spanoudakis, G., 2010. Comprehensive monitoring of BPEL processes. Internet Computing, IEEE 14, 50–57.
Brauner, J., Schaeffer, B. 2008. Integration of GRASS functionality in web based SDI service chains. In FOSS4G Conference. Cape Town,USA.
Cardoso, J., Sheth, A., Miller, J., Arnold, J., Kochut, K., 2004. Quality of service for workflows and web service processes. Web Semantics: Science, Services and Agents on the World Wide Web 1, 281–308.
CartoWeb, 2005. CartoWeb. Camptocamp SA, 〈http://www.cartoweb.org/〉, (accessed 2010.10.27).
Chaolin, W., Jianqin, W., Yong, X., Jianping, G., 2005. Study of task managing strategy in remote sensing information analysis and service grid node. In Geoscience and Remote Sensing Symposium, 2005. IGARSS '05. Proceedings. 2005 IEEE International, 4 pp.
Chen, A., Di, L., Wei, Y., Bai, Y., 2009. Use of grid computing to model virtual geospatial products. International Journal of Geographical Information Science 23 (5), 581–604.
Chen, N., Di, L., Yu, G., Gong, J., 2010. Geo-processing workflow driven wildfire hot pixel detection under sensor web environment. Computers and Geosciences 36, 362–372.
Coward, D., Yoshida, Y., 2003. Java(tm) Servlet Specification Version 2.4. Sun Microsystems, Inc., California, USA (November).
Crockford, D., 2001. JSON. 〈http://www.json.org〉, (accessed 05.08.09).
Di, L., 2005a. The implementation of geospatial Web services at GeoBrain. In: Proceedings of 2005 NASA Earth Science Technology Conference.
Di, L., 2005b. Customizable virtual geospatial products at web/grid service environment. In: Geoscience and Remote Sensing Symposium, 2005. IGARSS '05. Proceedings. 2005 IEEE International, pp. 4215–4218.
Dustdar, S., Schreiner, W., 2005. A survey on web services composition. International Journal of Web and Grid Services 1 (1), 1–30.
ExtJS, 2008. ExtJS. Sencha, 〈http://www.sencha.com/products/extjs/〉, (accessed 14.12.10).

Friis-Christensen, A., Lucchi, R., Lutz, M., Ostlander, N., 2009. Service chaining architectures for applications implementing distributed geographic information processing. International Journal of Geographical Information Science 23, 561–580.

Garrett, J.J., 2005. Ajax: A new approach to web applications. 〈http://www.adaptivepath.com/publications/essays/archives/000385.php〉, (accessed in 10.07).

Google Maps API, 2005. Google Maps API Family. Google, Inc. Google, Inc., California, USA. 〈http://code.google.com/apis/maps/index.html〉, (accessed 15.11.10).

Gorton, S., Reiff-Marganiec, S., 2006. Towards a task-oriented, policy-driven business requirements specification for Web services. In: 4th International Conference on Business Process Management, BPM 2006, September 5, 2006–September 7, 2006, Springer Verlag, Vienna, Austria, pp. 465–470.

Gruber, T.R., 1993. A translation approach to portable ontology specifications. Knowledge Acquisition 5 (2), 199–220.

ter Beek, H., Bucchiarone, M., A., Gnesi, S., 2006. A survey on service composition approaches: From industrial standards to formal methods. Technical Report 2006-TR-15.

Heloisa Martins, S., Tseng, M.M., 1996. Workflow technology-based monitoring and control for business process and project management. International Journal of Project Management 14, 373–378.

Jaeger, E., Altintas, I., Zhang, J., Lud, B., Pennington, D., Michener, W., 2005. 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, Santa Barbara, CA: Lawrence Berkeley Laboratory, pp. 87–90.

Jiang, L., Gong, J., Li, B., 2007. Research on Technology of Automatic Geospatial Web Service Composition Based on Semantic Web. doctoral dissertation of Wuhan University.

Juric, M.B., Krizevnik, M., 2010. WS-BPEL 2.0 for SOA Composite Applications with Oracle SOA Suite 11g. Packt Publishing Ltd., Birmingham, UK.

ka-Map, 2007. ka-Map 1.0. DM Solutions Group Inc., DM Solutions Group Inc. Ottawa, Ontario, Canada. 〈http://ka-map.maptools.org/〉, (accessed 20.12.10).

Kepler, 2008. Getting Started with Kepler, 〈https://code.kepler-project.org/code/kepler-docs/trunk/outreach/documentation/shipping/getting-started-guide.pdf〉, (accessed 10.07.11).

Kiehle, C., 2006. Business logic for geoprocessing of distributed geodata. Computers & Geosciences 32, 1746–1757.

Maathuis, B., Van Westen, C., 2005. Flood Hazard Analysis Using Multitemporal SPOT-XS Imagery. ILWIS Application Guide. International Institute for Aerospace Survey and Earth Sciences (ITC) 19C28. International Institute for Geo-Information Science and Earth Observation (ITC), The Netherlands.

Manegold, J., 2003. Using the ModelBuilder of ArcGIS 9 for Landscape Modeling. Trends in Landscape Modeling. In: Proceedings at Anhalt University of Applied Sciences, pp. 240–245.

MapLab, 2003. MapLab: manuel pour débuter. 〈http://www.maptools.org/maplab/index.phtml?page=maplab_tutorial_fr.html〉, (accessed 03.11.10).

Noy, N.F., McGuinness, D.L., 2001. Ontology development 101: A guide to creating your first ontology, Citeseer.

OASIS, 2007. Web Services Business Process Execution Language, Version 2.0.

OGC, 2007. OGC Web Services Common Specification, Version 1.1.0. OpenGIS® Implementation Specification.

OMG, 2002. OMG Model Driven Architecture. Object Management Group, Inc. 〈http://www.omg.org/mda/〉, (accessed 20.11.10).

OpenLayers, 2005. OpenLayers. Open Source Geospatial Foundation. 〈http://openlayers.org/〉, (accessed 25.09.10).

Ouyang, C., Van Der Aalst, W.M.P., Dumas, M., H.M. ter Hofstede, A., 2006. Translating bpmn to bpel. BPM Center Report BPM-06-02, BPMcenter.org.

Reed, C., 2009. The OGC and REST. 〈portal.opengeospatial.org/files/?artifact_id=32222〉, (accessed 22.08.11).

Scherp, G., Hasselbring, W., 2010. Towards a model-driven transformation framework for scientific workflows. Procedia Computer Science 1, 1519–1526.

Sun, Z., Ye, S., Wei, J., 2009. Implementation of visual BPEL process monitoring. Computer Systems & Applications, 134–142.

Sun, Z., Yue, P. 2010. The use of Web 2.0 and geoprocessing services to support geoscientific workflows. In Geoinformatics, 2010 18th International Conference on, pp. 1–5.

Taverna, 2009. Taverna. 〈http://www.taverna.org.uk/〉, (accessed 30.08.11).

Vuong Xuan, T., Tsuji, H., 2007. OWL-T: an ontology-based task template language for modeling business processes. In Software Engineering Research, Management & Applications, 2007. SERA 2007. 5th ACIS International Conference on, pp. 101–108.

Vuong Xuan, T., Tsuji, H., 2009. A task-oriented framework for automatic service composition. In Services – I, 2009 World Conference on, pp. 615–620.

White, S.A., 2004. Introduction to BPMN. IBM Corporation. IBM Corporation White Plains, NY, USA, pp. 2008–2029.

Yu, C., Hexin, L., Ji, G., Huamao, G., 2007. Task oriented Web service requirement framework. In e-Business Engineering, 2007. ICEBE 2007. IEEE International Conference on, pp. 503–506.

Yue, P., Di, L., Yang, W., Yu, G., Zhao, P., 2007. Semantics-based automatic composition of geospatial Web service chains. Computers & Geosciences 33, 649–665.

Yue, P., Gong, J., Di, L., Yuan, J., Sun, L., Sun, Z., Wang, Q., 2010. GeoPW: laying blocks for the geospatial processing web. Transactions in GIS 14, 755–772.

Yue, P., Sun, Z., Gong, J., Di, L., Lu, X., 2011. A provenance framework for Web geoprocessing workflows. The IEEE International Geoscience and Remote Sensing Symposium (IGARSS) 2011, Vancouver, Canada.

Zhao, P., Di, L.,Wei, Y., 2006. A virtual data product toolkit based on geospatial Web service orchestration. In: Proceedings Geoinformatics 2006, Reston, VA, USA.