Geo-processing workflow driven wildfire hot pixel detection under sensor web environment

Nengcheng Chen\textsuperscript{a,b}, Liping Di\textsuperscript{a,\textasteriskcentered}, Genong Yu\textsuperscript{a}, Jianya Gong\textsuperscript{b}

\textsuperscript{a} Center for Spatial Information Science and Systems (CSISS), George Mason University, 6301 Ivy Lane, Suite 620, Greenbelt, MD 20770, USA
\textsuperscript{b} State Key Lab for Information Engineering in Surveying, Mapping and Remote Sensing (LIESMARS), Wuhan University, 129 Luoyu Road, Wuhan 430079, China

1. Introduction

Significant progress has been made in collecting, analyzing, and modeling large amounts of diverse distributed geospatial data for Earth science research and applications. Di et al. (2006) show that a product virtualization approach can be implemented in a Grid service environment. Bai et al. (2007) discuss how federation catalogue services for three distinct geospatial catalogue services can retrieve heterogeneous metadata. Yang et al. (2007) discuss the concepts and applications of the Spatial Web Portal. However, effectively processing large volumes of Sensor Web data to feed application-specific information to those geospatial service systems enabled for a Geo-Processing Workflow (GPW) remains a problem. This paper discusses how to couple the geospatial Sensor Web information model and service specification with GPW to generate a live sensor product in a geospatial Sensor Web environment.

1.1. Geo-Processing Workflow (CPW)

The current workflow description languages in a Web service environment follow two separate standards: the Web Service Flow Language (WSFL) of IBM and XML LANGuage (XLANG) of Microsoft. The Business Process Execution Language (BPEL) and its extension for Web Services (BPEL4WS) is a new language of the above separate standards, and based on technologies like the Web Service Description Language (WSDL), XML Schema, and XPath. WSDL has the greatest influence on BPEL because the BPEL process model is built upon the WSDL specification. BPEL itself defines the syntax for an XML-based programming language (Curbera et al., 2006; Juric et al., 2004). Although it has some limitations, BPEL is nevertheless an adequate programming language because it has properties like loops (while), algorithm branching (switch) and explicit definition of variable type, e.g. Integer, Float. The result of the complete definition of a process is a BPEL script that will be interpreted by an orchestration engine like Oracle BPEL Process Manager or Active BPEL (Active Endpoints). The engine can be seen as a runtime environment that interprets the BPEL script.

In 2004, Open Geospatial Web Service (OWS)-2 offered some suggestions for the construction of GPW. There have since been...
The Geo-Processing Workflow plays an important role in the real-time or near real-time discovery and retrieval of remote-sensing observations in the Sensor Web environment. However, integration of SWE with GPW has the following problems:

1. How do applications tie a suite of services together for processing data flowing from the Sensor Web environment?
2. What is the model for mapping application data requirements onto individual services?
3. How are services instantiated and managed?

Therefore, organizing SPS, SOS, WPS, and WCS as a service chain to produce a real-time or near real-time sensor map has become a challenge in Sensor Web-based applications. This paper mainly tries to explain how to use GPW to process real-time remote-sensing observations.

1.4. Organization

This paper presents a GPW approach for using the SWE and OGC transactional sensor data services to generate live geospatial Sensor Web data service chains. The design considerations for and architecture of the abstract GPW model are presented in Section 2. The main components and different viewpoints of the concrete framework are described in Section 3. The implementation of the GPW under Sensor Web model design, instantiation, and execution is discussed in Section 4. The GPW prototype and use in a wildfire disaster emergency response system is presented in Section 5. Section 6 presents experiments to evaluate the performance of the wildfire hot pixel detection GPW for EO-1.
sensor data. The benefits and high performance of the proposed framework are discussed in Section 7. Finally, Section 8 summarizes the conclusions and discusses the next steps that are needed.

2. Abstract GPW framework

2.1. Design consideration

2.1.1. Interoperability

The IEEE Standard Computer Dictionary (IEEE, 1990) definition of interoperability, used by ISO/IEC 2382-01, Information Technology Vocabulary, Fundamental Terms, is as follows: “The capability to communicate, execute programs, or transfer data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units”. With respect to the general geospatial Sensor Web data service framework, the term interoperability is used to describe the capability of different services to exchange data with the same standard protocols using a common set of standard information models.

2.1.2. Flexibility

Flexibility refers to designs that can adapt when external changes occur. In the context of designing GPW systems for geospatial sensors, one can define flexibility as the ability of a system to respond to potential data node changes affecting the delivery of values.

2.1.3. Reusability

Reusability is the likelihood that a segment of a service component can be used with slight or no modification when a new service-based system is deployed. Reusable components reduce implementation time, increase the likelihood that prior testing and use has eliminated bugs, and localize code modifications when an implementation must be changed.

2.2. GPW model

As shown in Fig. 1, the life cycle for modeling a geospatial process consists of three phases: knowledge, information and data phase.

(1) Knowledge phase—Building a geospatial processing model by composing a composite geospatial process.

There are three kinds of model construction, transparent, translucent, and opaque.

(a) User-defined (transparent): The user queries the catalogue service with specific details on the different geospatial service types to define and manage the model.

(b) Workflow-managed (translucent): The user queries the catalogue service to find a given problem, and the knowledgebase then assists the user in selecting and configuring the most suitable geospatial service types for each step of model construction.

(c) Aggregate (opaque): The user presents a problem, and the knowledgebase uses the catalogue service to build a geospatial model with the best geospatial service type without the user’s intervention.

The “transparent” and the “translucent” approaches have been implemented in the GeoBrain model designer. Development of the “opaque” approach is in progress. The model designer provides a graphic user interface allowing the user to drag and drop “data type” and “service type” (Chen et al., 2009a) to build the model. Once the model has been created, it can be registered in the catalogue service as a service type for future use. It has its own inputs, outputs, and spatial and temporal scopes like other service types. Ontology and the Semantic Web (Zhao et al., 2009) play very important roles in the model building process. They suggest what to do and what to use by semantic matching, for example, locating data types for a specific topic and then finding service types for a desired data type, a particular method, or a specified geospatial scientific task.

(2) Information phase—Instantiating a geospatial process into a geospatial service chain.

In this phase, instance information for registered services is used to instantiate a geospatial model into a geospatial Web service chain. Such a service chain represents the information of how to derive the exact data product. A virtual data service is implemented to fulfill this phase:

(a) Service discovery: Since every service instance registered in the catalogue service has an association to a service type, it is easy to find a service instance for each service type in the geospatial model. If more than one service instance is available, quality of service (QoS) is used as a selection criterion. Of course, the level of match of services and data should be considered first in the following sequence of relevance: exact > exact > plug in > subsume. Other functional factors and conditions, such as accuracy, time, data format, and data projection should also be considered. If no service instance is discovered, this phase will be said to fail, and processing will stop here.

Fig. 1. Abstract GPW framework.
(b) Data discovery and fusion: In a geospatial model, there is no indication of what provides the inputs to the services. With the help of the catalogue service, the virtual data service automatically adds a relevant data service instance, which provides such input data, at the beginning of the service chain. If the outputs and inputs of adjacent services differ on data format and data projection, a data fusion service automatically deals with these differences. Examples of such data fusion services are the Web Coordinate Transformation Service and Data Format Translation Service.

(c) Representation of service chains: The representation of a service chain is critical to its materialization and reuse. Some industry initiatives have been developed to address the coordination requirements for sequencing and execution of services. We adopt the widely used Business Process Execution Language for Web Services, a language for the formal specification of business processes and business interaction protocols, to represent service chains. Although BPEL4WS was created for business process, the system designed here shows that it can meet the requirements of scientific processes.

3. Data phase—Executing a geospatial service chain to generate the geospatial data.

In this phase, a geospatial service chain is executed to derive the desired data products. For this purpose, we have the BPELPower, a service chain engine based on mainstream standards, such as BPEL, WSDL, SOAP, and J2EE, has been developed. It can run on the top of popular application servers, such as Tomcat, JBoss, Weblogic, and WebSphere.

4. Concrete hot pixel detection GPW

In the enterprise viewpoint, the general framework is to form a mechanism to dynamically feed and use live sensor data in Geo-Processing Workflows. Fig. 2 shows that the service framework for Geospatial Sensor Web data consists of a data service node, a processing service node, a presentation service node and a workflow engine. The data service node is responsible for feeding live sensor data to WPS and making use of the geospatial product. It can vary from a simple WCS server to a complex transactional WCS coupled with a Sensor Observation Service. The data service node is a Sensor Planning Service (SPS), a Sensor Observation Service (SOS), and a Transactional Web Coverage Service (WCS_T) or a Transactional Web Feature Service (WFS_T). The processing service node is a series of atomic sensor data processes packaged into a Web Processing Service (WPS). The presentation service node is a Web Map Service (WMS).

Fig. 2 also shows that, from the information viewpoint, real-time sensor data can be served at four different levels—raw sensor data, coverage or feature data, a geospatial product, and a sensor map. For sensor data, a combination of the SOS and SPS workflows provides live sensor raw data service. The raw observation data are served in O&M encoded format by the “GetObservation” operation. Sensor model data are served in SML encoded format by the “DescribeSensor” operation in the SOS implementation. The observed property may be any property associated with the type of the feature of interest. For coverage or feature data, an SPS+SOS+WCS_T/WFS_T workflow provides on-demand live sensor data service. The data are served using OGC-compliant formats and services. This allows operations on the dataset, including spatial subset, temporal subset, and re-projection, to be more flexible. For geospatial products, an SPS+SOS+WCS_T/WFS_T/WPS+WCS_T/WFS_T workflow provides on-demand live sensor product service. A specific process operates on sensor data. Such workflows yield geospatially meaningful products. For example, if a landslide susceptibility Web Processing Service (WPS) is chained in the workflow, the final product is landslide susceptibility data—a product meaningful for disaster responders. For a sensor map, an SPS+SOS+WCS_T/WFS_T/WPS+WCS_T/WFS_T+WMS workflow is used to provide on-demand live sensor mapping service. The result is one step from a geospatial product that can be presented to users with a visual map.

4. Implementation

4.1. Defining the WSDLs for SOS, SPS and WPS

WSDL provides a model and an XML format for describing Web services. It permits separation of the description of the abstract functionality offered by a service from the concrete details of a service description, such as “how” and “where” that functionality is offered. In the collaborative wildfire monitoring use case, WSDL has been defined for SOS, SPS, and WPS. The EO-1 SOS (Chen et al., 2009b), EO-1 SPS, JPL WPS, GMU WCS, and GMU CSW services have been set up.
The SOS WSDL has two transportation port types (HTTP GET and POST) and three mandatory operations (GetCapabilities, DescribeSensor, and GetObservation). The CSISS Sensor Web project hosts the XML document for SOS at: http://csiss.gmu.edu/sensorweb/wsdl/aist/eo1sos.wsdl.

The SPS WSDL has two transportation port types (HTTP GET and POST) and nine operations (GetCapabilities, DescribeGetFeasibility, GetFeasibility, DescribeSubmit, Submit, GetStatus, DescribeResultAccess, Update, and Cancel). The CSISS Sensor Web project hosts the XML document for SPS at: http://csiss.gmu.edu/sensorweb/wsdl/aist/eo1spws.wsdl.

The WPS WSDL has two transportation port types (HTTP GET and POST) and three mandatory operations (GetCapabilities, DescribeProcess, and Execute). The CSISS Sensor Web project hosts the XML document for WPS at: http://csiss.gmu.edu/sensorweb/wsdl/aist/jplwps.wsdl.

4.2. Design of the abstract model

An abstract model designer enables domain experts to use data types, service types, and existing abstract models as basic components for constructing new abstract models by clicking and dragging. The models represent the domain knowledge of the experts. The experts can choose to validate the abstract models and register those models into a catalogue compliant with the Catalogue Service–Web Profile (CSW) (Nebert, et al., 2007) specification, which will make the abstract models available for later use. An abstract model can be instantiated and transformed into a concrete BPEL process, which can be executed in a BPEL engine (e.g. BPELPower). To enable domain-specific modeling, ontology is used to support semantic matching. Data are externally described and categorized following geoscience domain ontology. The ontology is internally managed by an ebRIM classification, expressed in a tree. The basic ontology is derived from GCMD earth science keywords. Once the model is created, it can be registered in the catalogue service as a service type for future use. Like other service types, it has its own inputs, outputs, and spatial and temporal scopes.

In the collaborative wildfire monitoring experiment, using data from EO-1 Hyperion 242 bands, additional sub-categories under “Earth Science/Land Surface/Surface Radiative Property/Land Surface Reflectance/” were added to the geoscience domain ontology. The abstract model designer developed by CSISS (http://laits.gmu.edu/vdp/) is used to produce the EO-1 fire classification abstract model and register the model in the CSISS Catalogue Service for the Web (CSW) (http://laits.gmu.edu:8099/csw-echo/Publication).

4.3. Dynamic generation of the concrete workflow

In this phase, a geospatial model is instantiated into a geospatial Web service chain with registered service instance information. Such a service chain represents the information on how to derive the exact data product. A virtual data service is implemented to fulfill this phase. It has the following steps:

1. Service discovery: Since every service instance registered in the catalogue service has an association to service types, it is easy to find a service instance that meets the requirement of each abstract service type request in the geospatial model.

2. Data discovery and fusion: A geospatial model has no information on the sources of the inputs to the services in the model. With the help of a catalogue service, a virtual data service automatically adds a relevant data service instance, which provides such input data, at the beginning of a service chain.

3. Representation of service chains: The representation of a service chain is critical to its development and reuse. For the system described here, the widely used BPEL4WS, a language for the formal specification of business processes and business interaction protocols, is adopted to represent the service chain.

From the above three steps, the concrete BPEL workflow can be generated dynamically. The BPEL workflow includes processes. One process contains partnerLinks, variables, and a sequence of virtual WCS. Fig. 3 shows the workflow for the collaborative workflow.
wildfire monitoring experiment. The hot pixel detection procedure can be expressed as an “AIST_GMU_JPL_Fire” process element. The “AIST_GMU_JPL_Fire” process element contains “client” and “WPS” partnerLink elements. The “WPS” partnerLink element contains “Execute_InputVariable”, “Execute_OutputVariable”, and “vwcs_sequence” sequence elements. The “vwcs_sequence” sequence element contains the “vwcs_receive” element (it is the “receive” type) and the “WPS_Execute_Input” and “WPS_Execute_Output” elements (they are “assign” type). The “WPS_Execute_Input” element contains one or more “copy” elements.

4.4. Virtual WCS based on secure BPELPower

In this phase, a geospatial service chain is executed to derive the desired data products. For this purpose, a secure BPELPower has been developed and the geospatial service chain packaged as a virtual WCS. For example, in the collaborative wildfire monitoring experiment, the Virtual WCS URL is “http://lait.s.gmu.edu:8099/VWCS_AQ/wcs1020?service=WCS&request=GetCoverage&coverage=VIRTUAL:EO1_Fire_Model&bbox=-74.3785,40.1898,-74.012,41.1275&crs=EPSG:4326&width=200&height=200&format=GTiff”. This is called a Virtual WCS (VWCS). The data extent and type served through this VWCS are registered into the LAITS catalogue service. The coverage name uniquely identifies the registered abstract model. For this case, one model was registered as the VIRTUAL:EO1_Fire_Model. A user may create a model and register it in the catalogue service. The abstract model designer is at http://lait.s.gmu.edu/vdp/. The applicable extent of the model is normally defined when the model is designed and registered. The default extent is global, such that in principle, data for any area on the Earth could be retrieved. However, this depends on the actual availability of input data for the model. Two projections are supported: geographic Lat/Lon and UTM. Code EPSG:4326 is for lat/lon. The code for UTM depends on the zone where data is available; for the data example, which is in zone 18, the code is EPSG:32618.

5. GPW prototype

5.1. Implementation

Web services and Java technology have been used to implement SOS registry service middleware compliant with the OGC CSW and SOS standards. A demonstration of this implementation has been deployed in the “GeoBrain” server. It can be accessed at
"http://csiss.gmu.edu/sensorweb/demo.html". It consists of the following two components:

1. **GPW designer**: Fig. 4 shows how the model designer is used to build a wildfire hot pixel detection model. The left column shows the data type and service type registered with the catalogue service, and the right column is the model graphic representation. The process, carried out automatically by the designer, is as follows: select the "Fire_Ocurrence" data type; then find a service type whose output type is "Fire_Ocurrence". Only the service types that satisfy the output type are listed and selectable so that match errors between service type and data type are impossible. The designer finds that the "Thermal_Classifier" service type generates "Fire_Ocurrence" as output. It selects the "Thermal_Classifier" service type. The input data types to "Thermal_Classifier" are "EO1_Hyperion_Band110", "EO1_Hyperion_Band150", "EO1_Hyperion_Band210", and "EO1_Hyperion_Band213". If more than one service satisfies the output type, the designer allows the user to view their metadata to assist in selection; the last step is to find those "EO-1 SOS" service types whose output data types are "EO1_Hyperion_Band110", "EO1_Hyperion_Band150", "EO1_Hyperion_Band210", and "EO1_Hyperion_Band213" as step 2.

2. **GPW engine-BPELPower**: Fig. 5 shows its user interface. WSDL-based web services and BPEL-based web services chains can be deployed and executed dynamically in BPELPower, where their validity is checked. Different invocations (e.g., HTTP POST/GET and SOAP document/rpc) are well supported.

### 5.2. Use in a wildfire disaster emergency response system

The proposed GPW framework can be used in the wildfire disasters emergency response system. The wildfire hot pixel detection workflow in the Sensor Web environment has been presented as part of the EO Geo-Processing Workflow (GPW) at the OGC OWS-5 meeting. This wildfire use scenario describes a real-time collaboration between agencies using different platforms and sensors. Geologists are using the proposed GPW for fire location detection. The Expert designs an abstract Geo-Processing Model for fire location detection using the abstract model designer and registers the model in the ebRIM catalogue server. The GMU Business Process Execution Language (BPEL) engine BPELPower passes the data to a NASA JPL WPS for extracting fire information. The resulting data is ingested by middleware a WCS. The data or desired subset is then ready for the end users in a specified format and projection.

### 6. Case study

#### 6.1. The EO-1 live Hyperion data fire classification service

This study is a real-time collaboration between different agencies using different platforms and sensors. The CSISS team is focused especially on development of an asynchronous model for the Sensor Web data. An asynchronous model can be interpreted as the workflow if the concept of a geospatial virtual product developed by CSISS is applied. End users interact with a standard CSW for fire extent data. The CSW seeks and finds a virtual product that can be served through a standard WCS. The WCS initiates execution of the model workflow after data has been received from the EO-1 SOS. Once the data is available in the SOS, it is ingested into a WCS through an SOS registry operation. First, the BPEL engine passes the data to a JPL WPS to extract fire information. The resulting data is ingested to the WCS and the data or desired subset is ready for the end users in a specified format and projection. All these processes are completed automatically. This avoids human delays in passing around data and information.

#### 6.1.1. Data Set

EO-1 provides a Sensor Observation Service—SOS (http://eo1.geobliki.com/sos/) and a Sensor Planning Service—SPS (http://eo1.geobliki.com/sps/). Observations can be scheduled by SPS, while the basic service information and offering capability can be obtained through the SOS “GetCapabilities” operation. The high spectrum image data from the Hyperion instrument can be obtained through the SOS “GetObservation” operation, accepting only data more recent than 2005-10-18T15:19:39.000Z.

#### 6.1.2. Approach

Fig. 6 shows a fire classification scenario using near real-time Hyperion data. The workflow retrieves data in the four EO-1 SOS bands (band name is 110, 150, 210 and 213) and generates the fire location map shown in Fig. 6 by the following four steps:

1. **Model design for fire location detection**: The expert designs an abstract Geo-Processing Model for fire location detection using the abstract model designer and registers the model in the ebRIM catalogue server.

![Fig. 6. A wildfire hot pixel detection scenario using near real-time EO-1 Hyperion data.](image-url)
Table 1
Service node of EO-1 case study.

<table>
<thead>
<tr>
<th>Service node</th>
<th>Service provider</th>
<th>Operation</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOS</td>
<td>EO-1</td>
<td>GetObservation</td>
<td>Time-space</td>
<td>O&amp;M</td>
</tr>
<tr>
<td>CSW</td>
<td>LAITS</td>
<td>Publish</td>
<td>O&amp;M</td>
<td>Coverage</td>
</tr>
<tr>
<td>WCS</td>
<td>CSISS</td>
<td>GetCoverage</td>
<td>GeoTIFF</td>
<td></td>
</tr>
<tr>
<td>WPS</td>
<td>JPL</td>
<td>Execute</td>
<td>GeoTIFF</td>
<td></td>
</tr>
<tr>
<td>BPELPower</td>
<td>CSISS</td>
<td>Execute</td>
<td>BPEL</td>
<td></td>
</tr>
<tr>
<td>WPS</td>
<td>CSISS</td>
<td>GetCoverage</td>
<td>BPEL</td>
<td></td>
</tr>
</tbody>
</table>

Table 2
EO-1 Hyperion data set.

| Dataset Width Height Size (M) General GPW time (s) Wireless Wire |
|-----------------|-----------------|-----------------|
| 1 100 100 0.08  48.1 52.7 |
| 2 200 200 0.32  50.97 57.5 |
| 3 300 300 0.72  49.2 55.1 |
| 4 400 400 1.28  51.7 54.3 |
| 5 500 500 2.00  53.8 54.4 |
| 6 600 600 2.84  52.6 54.4 |
| 7 700 700 3.60  55.1 56.9 |
| 8 800 800 4.80  60.6 60.8 |
| 9 900 900 6.20  59.3 59.1 |
| 10 1000 1000 7.64  61.9 59.8 |
| 11 1500 1500 11.44  68.5 68.1 |
| 12 2000 2000 14.28  75.4 76.3 |
| 13 2500 2500 19.12  79.7 84.8 |
| 14 3000 3000 22.9  85.2 86.7 |
| 15 3461 3461 32.32  121.1 123.45 |

(2) Workflow instantiation for fire location detection: The decision support system (DSS) discovers the model using the “fire detection” keywords, instantiates the model and generates a concrete Geo-Processing workflow. The workflow chains EO-1 SOS, GMU WCS, and JPL WPS to generate a fire location map.

(3) Workflow execution for fire location detection: A secure BPELPower engine (http://geobrain.laits.gmu.edu:8099/bpel) deployed in the GeoBrain server executes the workflow. As shown in Table 1, the near real-time encoded observation data is

(a) Retrieved from EO-1 SOS using the “GetObservation” operation with the data described by the O&M specification,
(b) Registered as WCS coverage using the LAITS CSW (http://laits.gmu.edu:8099/LAITSCSWVM2/discovery) server through the “Publish” operation,
(c) Fed from the transactional WCS (http://geobrain.laits.gmu.edu/cgi-bin/gdalwcs/gdalwcs) deployed in the GeoBrain server to the JPL WPS (http://aiweb.jpl.nasa.gov/wps/cgi-bin/wps.py).

(4) Workflow product distribution for fire location detection: The process result is served by CSISS WCS.

6.1.3. Response time
The study was of the response time for common GPW in wire or wireless network mode. As shown in Table 2, the width of the requested coverage varies from 100 to 1221 pixels while the height varies from 100 to 3461 pixels. The data size varies from 0.08 to 32.32 Mbytes. The general GPW execution time was measured under both wireless and wire environments. In the wire network mode, the client used a wire network to fetch data from the VWCS server, while, in the wireless network mode, a wireless network was used to retrieve data from the VWCS server. Fig. 7 shows the results. The response time in both modes increases linearly with the data size. The growth rate of the response time as a function of the amount of data is similar for both modes.

6.1.4. Times for different processing stages
In order to analyze the factors affecting performance of the workflow in EO-1 Hyperion data fire classification, the times for five main processing stages were observed. Fig. 7 shows the response times for deriving SOS service information and the sensor information register from SOS (Register), constructing a sensor data processing model and generating a BPEL script (Model), fetching live sensor observation data from SOS using WCS_T (WCS_T), classifying fires using the JPL WPS “Execute” operation (WPS) and executing the secure BPEL engine (BPEL).

The “WCS_T” time tends to be comparable to that for “WPS”, the “Register” time is between those for “BPEL” and “Model”, and the “BPEL” time is more than that for “Model”. Moreover, the variation of the response time with data size in the “Model” and “WPS” phases is smaller than that of “Register”, “WCS_T”, and “BPEL”. The timing for the “Model” and “WPS” procedures is robust in that all of the process was conducted on local machines. However, the times for “Register”, “WCS_T”, and “BPEL” are unstable because of uncontrolled network bandwidth, especially the “BPEL” procedure, which must repeatedly retrieve and reload the WSDL and schema involved in the general framework from different remote service nodes.

The results converted into percentages of the processing time, are shown in Fig. 8. “Register” time is about 16.71 s and occupies 14% of the time, “Model” time is about 6.17 s and occupies 5%, “WCS_T” time is about 27.1 s and occupies 22%, “WPS” time is about 30.61 s and occupies 25%, and “BPEL” time is about 42.2 s and occupies 34%. The “WCS_T”, “WPS”, and “BPEL” times are the dominant influences on the response time.

6.1.5. SBIR Life cycle
Fig. 9 shows that the added SBIR (Search, Binding, Invoke, and Register) cycle enhances the framework, four kinds of sensor product are cached and registered as data types in CSW, and CSW plays the most important role in the SBIR cycle.

For example, in the WCS_T phase, the framework searches the CSW to find the sensor data or an SOS instance. Once the SOS has been identified, the CSW binds the SOS and the framework invokes the SOS. When the sensor data has been obtained, the framework feeds the sensor data to the WPS, for the space extent and data-encoding format specified. The O&M observation data is registered in CSW as “WCSCoverage” or “WFSLayer” and can be reused by the next process. The mechanism of other phases is similar to the WCS_T phase; thus, the procedure can be simplified and the quality of GPW improved.

Let TR represent Register time, TM Model time, TC WCS_T time, TP WPS time, TB BPEL time, and TV geospatial visualization time. If the client sends n sensor map data requests to the framework, the response time of the general framework mode is TG(n)=n*(TR+TM+TC+TP+TB+TV). Once the virtual data becomes a real product and is registered in the CSW, the response time of later sensor map data requests is the TV time, therefore, the best response time of the enhanced framework mode is TE(n)=TR+TM+TC+TP+TB+n*n*TV. Consider the “1221x3461” dataset use case. Fig. 10 shows that “TR+TM+TC+TP+TB” equals 123.45 s, while “TV” equals 28.6 s. The response time of the general framework mode would be TG(1)=152.05, TG(5)=760.25, TG(10)=1520.5, TG(15)=2280.75, and TG(20)=3041, while the response time of the enhanced framework mode is TE(1)=152.05, TE(5)=266.45, TE(10)=409.45, TE(15)=552.45, TE(20)=695.45. If R(n)=TE(n)/TG(n) represents the ratio
of TE to TG, then $R(1)=1$, $R(5)=0.35$, $R(10)=0.27$, $R(15)=0.24$, $R(20)=0.23$. As $n$ tends to infinity, $R(n)$ tends to $TV/TG(1)$. In this use case, $R(\infty)=0.188$.

7. Discussion

Integrating SWE services with Geo-Processing Workflow (GPW) is a complex problem. The proposed framework solves the problem by an automated method, using BPEL, Catalogue Services and other data and processing services.

7.1. The benefits of the proposed framework

To help satisfy the interoperability requirements in Section 2.1.1, the proposed framework adopts OGC SWE and OWS compliant service interfaces and information models for packaging, processing, and presenting remote-sensing data. The transaction Web Coverage Service was used between the Sensor Observation Service and the Web Processing Service to feed the live EO-1 Hyperion data into fire hot pixel detection WPS. The services interact with each other through standard operations and are chained together.

To address the flexibility requirements in Section 2.1.2, a variety of data, processing and presentation nodes are used to generate workflows for sensor data services in the Sensor Web environment. The GPWs tie a suite of services together for processing data flowing from the Sensor Web environment. The four different products of sensor data including raw sensor data, coverage or feature data, a geospatial product, and a sensor map can be acquired because of the flexibility of the service workflows.

To satisfy the reusability consideration of Section 2.1.3, the framework uses WSDL from the OGC services, an abstract model designer, and a VWCS based on the secure BPELPower engine to implement GPW model construction, instantiation, and execution, registered in CSW and reused by others.

7.2. The high performance of the proposed framework

The “BPEL” execution, “WPS” processing, and “WCS_T” times are the dominant influences on the total response time in
the proposed framework. In the test use case (discussed in Section 6.1.4), the “BPEL” time is about 42.2 s and occupies 34% of the response time, the “WPS” time is about 30.61 s and occupies 25% of the response time, and the “WCS_T” time is about 27.1 s and occupies 22% of the response time. The enhanced mode based on the SBIR cycle is more efficient than the general mode. For large data sets, the enhanced mode to general mode response time ratio is TV/TG(1), where TV is the time for sensor map visualization and the TG(1) is the time of full GPW execution. In the fire classification use scenario, this ratio equals to 0.188.

8. Conclusions and outlook

Achieving interoperability, flexibility, and reusability in Sensor Web data service for Geo-Processing Workflow in the Sensor Web environment is a major challenge. This paper proposes a general Sensor Web data service framework for GPW. The proposed framework consists of a data service node, a data processing node, a data presentation node, a Catalogue Service node and a BPEL engine. The framework has the following features: interoperability, flexibility, reusability, and high performance.

The next step will be to study how to provide the following components: an enhanced workflow engine to chain the diverse atomic processes into a uniform WPS, a decreased service schema round trip through the workflow engine to improve the quality of GPW, a workflow engine oriented to an OGC service interface to simplify the invocation of geospatial Web services, and a lightweight Sensor Model Language (SML) oriented workflow engine to process real-time and model-related sensor data.

Acknowledgements

This work was supported by grants from the US NASA ESTO/AIST Sensor Web program: A General Framework and System Prototypes for the Self-Adaptive Earth Predictive Systems (SEPS)—Dynamically Coupling Sensor Web with Earth System Models (No. NNX06AG04G, PI: Dr. Liping Di), the Chinese 863 program (No. 2007AA12Z230, PI: Dr. Nengcheng Chen) and National Nature Science Foundational of
The authors would like to thank our colleague, Dr. Barry Schlesinger, for proof reading the manuscript.

The authors would like to thank the editors and anonymous reviewer for their valuable comments and insightful ideas.

References


