INFORMATION TECHNOLOGY JOURNAL

Information Technology Journal 10 (6): 1126-1137, 2011
ISSN 1812-5638 / DOI: 10.3923/itj.2011.1126.1137
© 2011 Asian Network for Scientific Information

Interface Independent Geospatial Services Orchestration

Soravis Supavetch and Sanphet Chunithipaisan
Department of Survey Engineering, Geo-Image Technology Research Unit, Faculty of Engineering, Chulalongkorn University, Phayathai Road, Pathumwan, Bangkok, 10330, Thailand

Abstract: The complexity of geospatial services integration in modern GIS application is the key challenge of various recent researches in geospatial community. The integration of cross-organizational geospatial services, which are different standard interfaces, is one of problems that limit the use of such orchestration technology as BPEL for assembling geospatial services into a complex geospatial workflow. Because that geospatial services can be implemented over a number of standards and technologies that meet their requirements such as Web service, OGC Web service, or REST. In this study, the necessity and requirements of interface independent orchestration are addressed. The orchestration language is also designed response to interface independent orchestration purpose. Then multi-standard service interfaces over HTTP protocol can be assembled without the need of mediator (such as interface wrapping service). The language interpreter and execution, so called orchestration engine, is implemented behind OGC Web processing service for demonstrating the interface independent orchestration by client requests an execution. "Site selection" use case in which common problem of GIS projects are selected and used to define an example of orchestration script that contains a sequence of processes with different service interface standards. The result of execution is displayed with verified that cross-organizational geospatial services which is a different service interface, can be orchestrated through the implemented engine. For this solution, open geospatial information services from public organizations and authorities in which different service interface can be orchestrated without the need of mediator implementation.

Key words: Geospatial services chaining, geospatial services orchestration, geospatial services composition, workflow enactment service, transaction web processing service

INTRODUCTION

Geospatial information plays an important role at all levels of scientific disciplines. OGC (2008a) noted that "Geography is a foundation property for modeling the world in a coherent, intuitive way that location and time can be exploited as a unifying theme to better understand the context of most real and abstract phenomena". The geospatial information which is extracted the knowledge of the Earth can be used in many purposes e.g., hydrological analysis, wildfire prediction, air pollution monitoring, water and land resource management, agricultural production estimation, environmental management, disaster management and so on. Scientists use those geospatial-data to understand what is needed to utilize the data and associated information in the on-going scientific research.

Now-a-days, the numerous geospatial data are produced and provided through the web by public organizations and authorities at various levels of detail (local, regional, national and global) (Kiehle, 2006). The examples of those publication services are, JPL (Jet Propulsion Laboratory) Global Imagery Service, Microsoft TerraServer Map Server, Pacific and Yukon Region WFS for water quality monitoring stations, Geospatial Web Services from GeoBrain (Di, 2004a, INSPIRE Geo-portal for environment information, OK-GIS for disaster management, and so on. Facility of the Internet is a key of that increasing and chaining the way business in conducted, then companies are moving their main operations to the web for more automation, efficient business processes and global visibility (Tsalgatidou and Pilioura, 2002). The effect of this also gives scientific knowledge to be worldwide interdisciplinary as well.

Beyond geospatial services, the key success of these technologies is the interoperability because "no single organization produces all the data (so it's inconsistent) and no single vendor provides all the systems" (OGC, 2008a). The essential of this interoperability is usually appeared though a number of researches today in term of

Corresponding Author: Sanphet Chunithipaisan, Department of Survey Engineering, Geo-Image Technology Research Unit, Faculty of Engineering, Chulalongkorn University, Phayathai Road, Pathumwan, Bangkok, 10330, Thailand Tel: 066 2218 6651 Fax: 066 2218 6650

services composition, services chaining, services orchestration and services integration. Ability of compositing more complex processing tasks is the greatest value of geospatial web services (Einspanier *et al*., 2003).

The different type of geospatial services naturally published via different interfaces. To integrate those services, it requires the mediate component to collaborate those heterogeneous services for achieving a geo-processing solution. For examples, WFS (Web Feature Service) itself cannot coordinate with WMS (Web Map Service), WCS (Web Coverage Service) itself also cannot coordinate to WFS; therefore, the third entity (mediator) such as client-coordinated application/service, aggregate service, or workflow-managed service (Alameh, 2003) is necessary.

Several protocols, profiles, specifications, and standards are developed to service the geospatial information through Internet technologies. Open Geospatial Consortium (OGC) and ISO/TC 211, the international organization for standardization, have jointly released a number of specifications for interoperable geographic information services and message encodings. Those standard specifications allow developer to build reliable and sustainable geospatial services supporting geospatial users in multiple disciplines (Granell *et al*., 2010). The standardized geospatial web services (so called OGC Web Services) are defined using open non-proprietary Internet standards in particular the WWW standards of HTTP, Uniform Resource Locators (URLs), Multipurpose Internet Mail Extensions (MIME) types and Extensible Markup Language (XML).

OGC, WPS and WCPS, a standardized geo-processing service for vector based and raster based, are designed to define an interface that facilitates the publishing of geospatial processes to clients. OGC said that WPS or WCPS processes can be incorporated into service chains and those services are designed to call a sequence of web services, thus acting as the service chaining engine (OGC, 2008b).

Moreover, OGC have acknowledged on using BPEL (Business Process Execution Language) which is an OASIS standard executable language for generating recommendations and guidelines for geospatial services orchestration (OGC, 2009). BPEL aim is to enable “programming in the large” which programmed by larger or smaller groups of people over the long time periods and coding managers place emphasis on partitioning work into modules, possibly written by different people with precisely-specified interactions (DeRemer and Kron, 1975). BPEL describes the automated arrangement, coordination, and management of web services and focuses on the collection of related, structured activities or tasks that produce a specific service (OASIS, 2007). BPEL implements on such technologies as WSDL, XML Schema, XPath, XSLT and XML Information Set. Using BPEL for orchestrating OWS services is proprietary to those Web Service standards (Gone and Schade, 2008). For example, OWS services, which are orchestrated by BPEL engine, must describe service interface through WSDL and provide message exchange through SOAP protocol.

BPEL is relied on SOAP and WSDL while some public and open geospatial web services are not strictly relied on WSDL and SOAP. The differences between those OGC and W3C web services are addressed and described by Ioup *et al*. (2008). For this reason, geospatial services orchestration by using BPEL is invalid in case of particular services in a chain do not support SOAP and WSDL. In addition, some geospatial services such as WMS and WCS response a geospatial message in a byte-stream via HTTP transport protocol.

Weiser and Zipf (2007), Fleuren and Muller (2008) and Sancho-Jimenez *et al*. (March 25, 2011 2008) proposed the mediator (e.g., Proxy OGC Web Service) as a message exchanging service to associate those different interfaces into homogenous workflow. That proxy service provides a way to integrate OGC Web Services into the BPEL process.

In this research perspective, constrain on above heterogeneous geospatial services integration is an interface. Not only W3C/OASIS and OGC Web Service interface but also interface adaptation (i.e., WPS 0.4.0 and WPS 1.0.0). Then the interface independent orchestration language is designed and implemented. This study also evaluates the current services orchestration and examines the public geospatial information services for improving interface independent orchestration. Then the achieving solution is reported by developing the workflow enactment service in which an interface independent orchestration language is executed. That several service interfaces in which delivered over HTTP transportation mechanism can be orchestrated by our orchestration engine such as HTTP/GET KVP, HTTP/POST XML-based document (with SOAP or non SOAP), HTTP/POST byte-stream body and REST protocol. This workflow enactment service allows user to pre-define a services chain across the multiple standards interface of geospatial messaging. In testing scenario a service prototype is examined for addressing composition problem and its solution is explained in the conclusion.

TOWARD METHODOLOGY FOR GEOSPATIAL SERVICES CHAINING

Interoperability reference model: In ISO19119 (2001), geographic interoperability is the ability of information systems that run software cooperatively over the network for manipulating such spatial information about the objects and phenomena on, above, and below the Earth's surface. The ability of that interoperability is about to access, integrate, analyze and present geospatial data across a distributed computing environment by machines and personal devices such as cell phones, PDAs and smart phones (Lee and Percivall, 2008).

In Web service architecture, a loosely-coupled in distributed interaction between web services is designed that is easier to develop interoperability among different web services in which supporting different interfaces and protocols. For example, the interaction between two standard web services that use SOAP over HTTP as transport mechanism, may provide the different of operations, parameters and business protocols (different constrains) (Benatallah *et al*., 2005).

- In OGC Web service, the services such as WMS, WFS, and WCS cannot coordinate to each others. The mediator is needed to perform interoperability. (ISO19119, 2001) defined User defined (transparent) orchestration. The user acts as mediator and manually composes and executes the geospatial services
- Workflow-Managed (translucent) orchestration. A sequence geospatial services are executed and controlled by a workflow service that acts as a mediator
- Aggregate Service (opaque-orchestration). The sequence geospatial services appear to the user as a single service. The aggregate service acts as the mediator that services integration performs in the back-end

The needs of above three approaches in geospatial community come from three sources: 1) the large heterogeneity of open geospatial services; 2) more and more complex geospatial problems can be solved by an interoperability of a number of geospatial services and 3) a growing number of users in geosciences disciplines.

Interface standards: "OGC has been dedicated to the standardization of the interfaces of geospatial Web services to enable interoperability" (Zhao *et al*., 2007). In Web service, the structure of information data in which participant exchanges depends on specific domain and implementation. SOAP provides only the way for defining what information gets sent and how but in geospatial data the requirement for a common specific structure for services interoperability is necessary. Thus, OGC decided to design geospatial web service interface to common model and share some unified characteristics. The same type of OWS service is implemented and published via the same interface; for examples, WMS provide GetCapabilities, GetFeatureInfo and GetMap operations with unified c parameters for servicing map image, WFS provides GetCapabilities, GetFeature, DescribeFeatureType operations for distributed geospatial features, WCS provides GetCapabilities, GetCoverage, DescribeCoverage operations for publishing geo-referenced image.

Additionally, Amirian and Alesheikh (2008) noted that Geospatial Web services and Web services differ in a way that Web services are composed of particular set of technologies and protocols. But Geospatial Web services are comprised of defined set of interface implementation specifications which can be implemented with diverse technologies.

Standard technologies for geospatial services orchestration: The growing of orchestration software tools and technologies is convincing the web service interface to SOAP and WSDL implementation even though OGC Web services. Those standards for enabling OGC Web Services orchestration are focused and elaborated for a long time by OGC since OWS-2 testbed (OGC, n.d.). OGC proposed to use BPEL hidden behind the vendor specific WfCS (Workflow Chaining Service) interface such as WPS (OGC, 2009). OWS services that appear in BPEL script must adapted to service via SOAP and WSDL (OGC, 2003, 2005, 2008a). Finally, a more complicated geospatial model and process flow can be defined into a service chain such as BPEL for complex geospatial applications and knowledge discovery (Zhao *et al*., 2007). For more discussion and detail about orchestration software tools and technologies are founded in the study of Peltz (2003).

In addition, WPS is extended to support the deployment/undeployment for any algorithm/workflow called Transactional WPS (WPS-T). Then BPEL profile can be deployed for executing as a single WPS process. In working of this research, no standardized way to package the workflow model to be exchanged in GI community, but WPS-T which is the research study of Schaeffer (2008) is a starting point for standardizing way to deploy geo-processing model to WPS. Even more, an executable JAR files or other XML-based workflow descriptions can be deployed to WPS-T as well.

Wrapper or mediator service: In open OWSs, which are platform-neutral, require a mediator service for wrapping service interface to facilitate the OWSs orchestration by BPEL. The deep description of that implementation is found in the research of (Sancho-Jimenez *et al.*, 2008). The mediator (or wrapper) services should acts as a proxy from SOAP interface to OWS interface and vice versa. The aim of that mediator is a bridging gap between OWS services and W3C/OASIS services for using SOAP and WSDL above OWS without code modification. In additional point of view, the mediator services or “communication services” described in “Technology viewpoint (a basis for cross platform interoperability)” (ISO19119, 2001) are hardware and software components in distributed system which is responsible for routing service request from client object to server object. The interoperations between those components are needed to achieve interoperability.

INTERFACE INDEPENDENT ORCHESTRATION

Among the variety of service interfaces, the common patterns can be captured. In a simple service call, HTTP GET is commonly used to invoke the service such as HTTP/GET KVP or HTTP/GET in REST. In this case XML is ignored for encoding request document. When special condition is needed for retrieving specific response, the standard XML-based request document is used to encode the constrain information then HTTP POST is required. For example, Filter Encoding statement is embedded in WFS GetFeature document for retrieving geospatial features. We can say that general services, which are publishing via Internet through HTTP operations, use those operations as both interface and transfer protocol.

The orchestration language in which supports control and manipulation on HTTP operations is able to say that the orchestration capability is independent from interface. Additionally, the component such as statement manipulations i.e., insert, delete, replace, rename and move for XML data manipulation allows two different services interfaces be transformed (Florescu *et al.*, 2003). For example, WFS response (wfs: Feature Collections) can be transformed to SOAP request for next service participant.

EXAMPLE OF INTERFACE INDEPENDENT ORCHESTRATION LANGUAGE

There are at least four components for a language that can orchestrate multiple services interface. (1) Geospatial data handling (2) Geospatial services coordination (3) Geospatial message control and manipulation and (4) Exception handling. Each component is not designed based on any standard service interface, but is focused in access mechanism for using technologies that are implemented in each component ranging from data format (e.g., GML, GeoJSON, GeoRSS, Shapefile and GeoTiff.) to communication related techniques (e.g., HTTP/GET KVP, HTTP/POST SOAP, HTTP/POST WFS, HTTP/POST ATOM and REST).

Difference geospatial services interface can be orchestrated through these four language components as the following example:

Geospatial data handling: Variety of geospatial information messages are ranging from plain text and XML-based document e.g., wms:GetFeatureInfo, GML, CityGML, or wfs:FeatureCollection to bytes stream object such as GeoTiff, Shapefile, or Image/png. Orchestration engine holds these geospatial messages through pre-defined variables. These variables provide the means for holding geospatial messages that constitute a part of the state of a geospatial processes. Each variable can be declared for handling geospatial data as the following syntax:

```
<Variable name=”NCName” mimeType=”QName”
portType=”QName”>
```

Most web services require XML-encoded to request service operation. The request document can be described in RPC (Remote Procedure Call) style or Document style such as SOAP RPC or wfs: Get Feature. The content of that document should not be strict by language. Client/User who writes the script may embeds that document as a child element of <Variable>. With this kind of this variable, user is responsible to aware that document is valid to use such as the following example:

```
<Variable name="getHosptial" mimeType="text/xml" portType="wfs:GetFeatureType">
    <wfs:GetFeature service="WFS" version="1.1.0"
        xmlns:cuwps="http://www.sv.eng.chula.ac.th/cuwps"
        xmlns:wfs="http://www.opengis.net/wfs"
        xmlns:ogc="http://www.opengis.net/ogc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.opengis.net/wfs
        http://schemas.opengis.net/wfs/1.1.0/wfs.xsd">
        <wfs:Query typeName="cuwps:bma_admin_poly">
            <ogc:Filter>
                <ogc:PropertyIsEqualsTo>
                    <ogc:PropertyName>LOCATION_TYPE</ogc:PropertyName>
                    <ogc:Literal>HOSPITAL</ogc:Literal>
                </ogc:PropertyIsEqualsTo>
            </ogc:Filter>
        </wfs:Query>
    </wfs:GetFeature>
</variable>
```

Geospatial services coordination: Most geospatial web services exchange geospatial data via HTTP transport protocol. For example, geospatial information services are delivered via HTTP GET/KVP, HTTP POST/XML-based, SOAP via HTTP/POST and REST (i.e., HTTP GET/PUT /DELETE/POST/HEAD). Those services implementation depend on their business gold and technologies related. Among those varieties of service interface such as WFS, WMS, WCS, WPS and WCPS, the common HTTP operations are used to services coordination.

To implement engine which contains sustainable capability for interface independent orchestration, the engine must provide service interaction by two sequence modules: The coordination module (Geospatial Services Coordination) concentrates on HTTP transportation mechanism and the interface manipulation module (Geospatial Message Control and Manipulation) concentrates on syntactic and semantic level of service interface.

On service coordination, our definition language provides mechanism to access and manipulate HTTP transportation structure such as <GetHeader/> and <SetHeader/> elements. Then a document in which sending between service and engine such as XML document can be interpreted, managed and transformed by Geospatial Message Control and Manipulation module.

HTTP structure accessibility gives an engine to support both on SOA and ROA principles. In SOA Web Service, SOAP message is transported via HTTP POST between service participants. The focus of SOAP related technologies in web service is a syntactic and semantic of the context (or payload) but in ROA such as REST focuses on HTTP operations without concentration on context. Thus, the separation of transportation mechanism and semantic interface give service with compatibility with both Web services and platform-natural OGC Web services.

In definition language, BPEL activities are extended by adding HTTP Header accessibility mechanism. For examples, <Receive/> activity defines a receiving message from service client. <Reply/> activity defines a message responding mechanism to client. <Invoke/> activity for specifying request mechanism via HTTP GET/POST/ PUT/DELETE to service participant. The following elements are the examples of the above mentioned syntax.

```
<!—Receive client/user request document -->
<Receive operation="wps:Execute"
outputVariable="wpsExecuteRequest" />
…
<!—Begin services orchestration -->
<Invoke operation="wfs:GetFeature" inputVariable="getHospital"
outputVariable="responseHospital"port="http://161.200.86.131/geoserv
er/ wfs"
method="POST" />
…
<Invoke operation="wfs:GetFeature" inputVariable="getGmlFromShape"
outputVariable="responseGml"
port="http://geobrain.laits.gmu.edu/axis/services/Vector_SHP2GML"
method="POST">
    <!—Assign HTTP header -->
    <HTTPHeaders>
        <SetHeader>
            <Name>SOAPAction</Name>
            <Literal>shp2gml</Literal>
        </SetHeader>
    </HTTPHeaders>
</Invoke>
…
<!—Reply solution to client/user -->
<Reply outputVariable="intersectedPolygon" />
```

Geospatial message control and manipulation: The data control and manipulation are necessary for the preparation of input parameter for the next process sequence. Some operations such as Intersection, Union, and Dissolve require input parameters in which issued from different services. In BPEL, the <Assign/> activity can be used to copy data from one variable to another input parameter. XML technologies such as XPath, XSLT are used to access and transform XML-based content for input parameter preparation purposes.

Additionally, some geospatial processed results need more operation accessible. For example, in WFS provides Filter Encoding to extract demanded feature from features resource, but features which are resulted from WPS processing do not have Filter Encoding for extracting post process features. Unlike an XPath, features document such as GML need more semantic access than syntactic access to spatial data. XPath provides only syntactic access to XML node in document structure, but Filter Encoding provides more access in spatial relation. In BPEL, XPath provides preparation mechanism to assign a parameter. Thus, Filter Encoding also needs to provide preparation mechanism to spatial data as well.

Moreover, XML data manipulation can be applied to cooperate with such XML as used in a specific purpose, which found in a number of researches in Web Service area. For examples, SOAP header in which extended for web service users authentication (Kadry and Smaili, 2007), SOAP message in which securing high sensitive data encrypted (Nabi *et al*., 2010), WSDL which is enhanced for describing QoS properties to client (Yan-Ping and Zeng-Zhi, 2007) and XML-based metadata for agricultural spatial information sharing in personalized information service (Wang *et al*., 2010).

The following are the examples of preparation definitions for input parameter preparation in various cases.

Basic assignment:

```
<Assign name="assignHospitalToBuffering">
    <Copy>
      <From inputValue="responseHospital"/>
         <To outputValue="getBufferingHospital">{$hospitalToBeBuffered}</To>
    </Copy>
</Assign>
```

The above example demonstrates the assign element <Copy> that copies a message from one variable to another variable by replacing value at {$buildingToBeBuffered} markup in which declared in the output variable context. Some OGC Web services publish a sizeable geospatial message such as wfs: Feature Collection or wcs: Coverage, the engine must use that value by reference instead of instance value directly. <GetRefereceURL/> element is used to inform engine to store data into file system and uses that data by reference as follows example.

```
<Assign name="assignUserHospitalDistance">
    <Copy>
        <From inputValue="wpsExecuteRequest">
            <GetReferenceURL/>
        </From>
         <To outputValue="getBufferingHospital">{$hospitalFeatures}</To>
    </Copy>
</Assign>
```

Assign activity also supports XPath similar to BPEL for accessing value in XML DOM as below example.

```
<Assign name="assignUserHospitalDistance">
    <Copy>
         <From inputValue="wpsExecuteRequest">//Input[1]/LiteralValue</From>
      <To outputValue="getBufferingHospital">{$hospitalDistance}</To>
    </Copy>
</Assign>
```

Geospatial data such as wfs: Feature collection that response from one WPS service require spatial filter in which extracting demanded features for sending as input parameters to next process actor in a service chain. User can define <ogc: Filter> inside <Assign> activity to filter spatial features at engine site as follows example:

Advance assignment:

```
<Assign name="assignRequiredIntersectArea">
    <Copy>
        <From inputValue="responseIntersectPolygon">
           <ogc:Filter>
                <ogc: PropertyIsGreaterThan>
                    <ogc:Function name="Area">
                    <ogc:PropertyName>THE_GEOM</ogc:PropertyName>
                  </ogc:Function>
                  <ogc:Literal>2000</ogc:Literal>
              </ogc: PropertyIsGreaterThan>
        </ogc:Filter>
    </From>
    <To outputValue="suitableArea">{$suitableAreaFeature}</To>
  </Copy>
</Assign>
```

Exceptions handler: In complex geospatial services orchestration with involving the number of services, the demand for managing exceptions is required that not only for process debugging purpose but also the compensation mechanism for geospatial web services composition such as described in the research of (Yanfeng *et al*., 2005).

A number of ways to manage an exception, which is a response from geospatial service in case of that is, an invalid work. A fault response from an <Invoke> activity is one source of faults, where the fault name is based on exceptions that thrown from geospatial information services such as ows: Missing Parameter Value, ows: Invalid Parameter Value or soapenv: Server. User Exception. Each service shall respond to an Exception Report message that describes what request is invalid. Similar to typical programming language the exception gives the orchestration script easier to maintain. Using fault handlers is simple and that similar to catching exception in Java which user defines in Try-Catch block. Fault handlers can be declared globally across an entire process or directly on an <Invoke> activity which is defined locally.

In addition, the occurred exception normally has two categories, error in transportation and exception from service provider. The exception which is issued from service provider is usually an XML-based structure for examples, ows: Exception Report and soapenv: Fault. To access fault name in that exception document an Xpath must be specified in "path" attribute of <catch> element. This approach gives an engine to catch any exception structures from any standard service provider.

The fault handler used to declare a catching of invalid task is demonstrated as below example.

```
<ProcessDefinition …>
      <Variables>…</Variables>
      <Activities>
         <!--Global Fault Handlers-->
         <faultHandlers>
           <catch faultName=”500”>
              <HTTPHeaders><GetStatus/></HTTPHeaders>
            </catch>
            <catch faultName=”ns1:Client” path=”//faultcode”>
                <!—handler an activities such as invoke, reply, and so on -->
            </catch>
            <catch faultName=”InvalidParameterValue”
               path=”/ExceptionReport/Exception/@exceptionCode”>
          …</catch>
          <catchAll>…</catchAll>
       </faultHandlers>
       <Sequence>
         <Receive operation="wps:Execute" />
         <Invoke operation="wfs:GetFeature" >
           <!--Local Fault Handlers-->
           <faultHandlers>
             <catch faultName=”ows:MissingParameterValue”>…</catch>
            <catch faultName=”ows:InvalidParameterValue”>…</catch>
              <catch faultName=”soapenv:Server.userException”>…</catch>
         </faultHandlers>
      </Invoke>
         <Invoke operation="wfs:GetFeature" />
          …
         <Assign name="assignSchoolToBuffering” />
          …
        <Reply outputVariable="intersectedPolygon" />
      </Sequence>
   </Activities>
</ProcessDefinition>
```

IMPLEMENTATION

We design an engine component similar to typical interpretation of computer programs that requires a structure of variable declaration, a control flow and an exception handler. That structure divides engine into a number of components i.e., Variable Handler, Control Flow Handler, Coordination Handler and Exception Handler (Fig. 1). The Variable Handler is used to handling a related geospatial data (e.g., wfs: GetFeature, wfs: Feature Collection, wps: Execute, GeoTiff, and Shapfile). Control Flow Handler is used to orchestrate a coordination of external services as defined in <Receive>, <Invoke>, <Assign>, and <Reply> activities or structured activities such as <Sequence> and <Flow>. Engine Manager will execute the service activities in order to orchestrate a sequence that is expressed in the orchestration script. The faults handler, which is user, defined in an orchestration script will be caught by Exception Handler when it occurs during services orchestration.

Additionally, we develop the Orchestration Script Repository which is used to store a script, that have to read, customize or insert new later. Engine Manager also provides script management through WPS-T deployed/undeployed process. The engine's components are drawn in Fig. 1.

TESTING

Scenario: The scenario for testing a service prototype is derived from common problem of GIS projects such as "site selection" algorithm (Stollberg and Zipf, 2008). The site selection is about finding the suitable area from various factors involving a difference specific location. For example, a family may look for a house near a school and hospital within a specific distance between the two places. The question for this algorithm could be "In which area do I have to rent a house from housing estate which is within distance of 0.5 km from a hospital, and within a distance of 0.4 km from a school. In practical the services, which are orchestrated by engine, require WFS service to provide locations of hospital, school and housing estate. WPS is also setup "buffer" and "intersect" operations for buffering those locations to polygon and then performs the intersection of hospital polygon, school polygon and housing estate polygon for making a solution which is the area that user may looking for renting a house.

The best way in which we propose to derive an orchestration script from any geo-processing workflows is following the steps in a sequence diagram. Many GI systems that have dynamic behavior can be expressed in terms of specific sequences of message between a small, fixed number of processes. We have to define a sequence

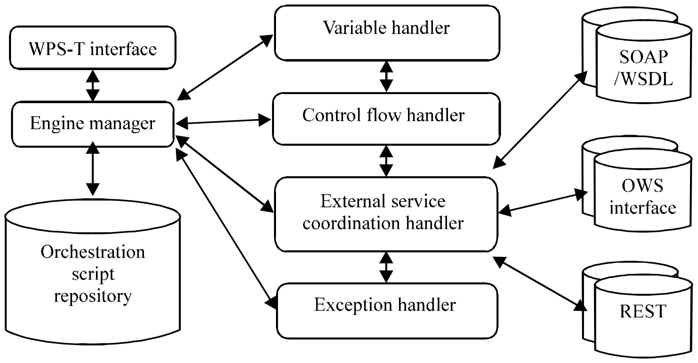


Fig. 1: Orchestration service components

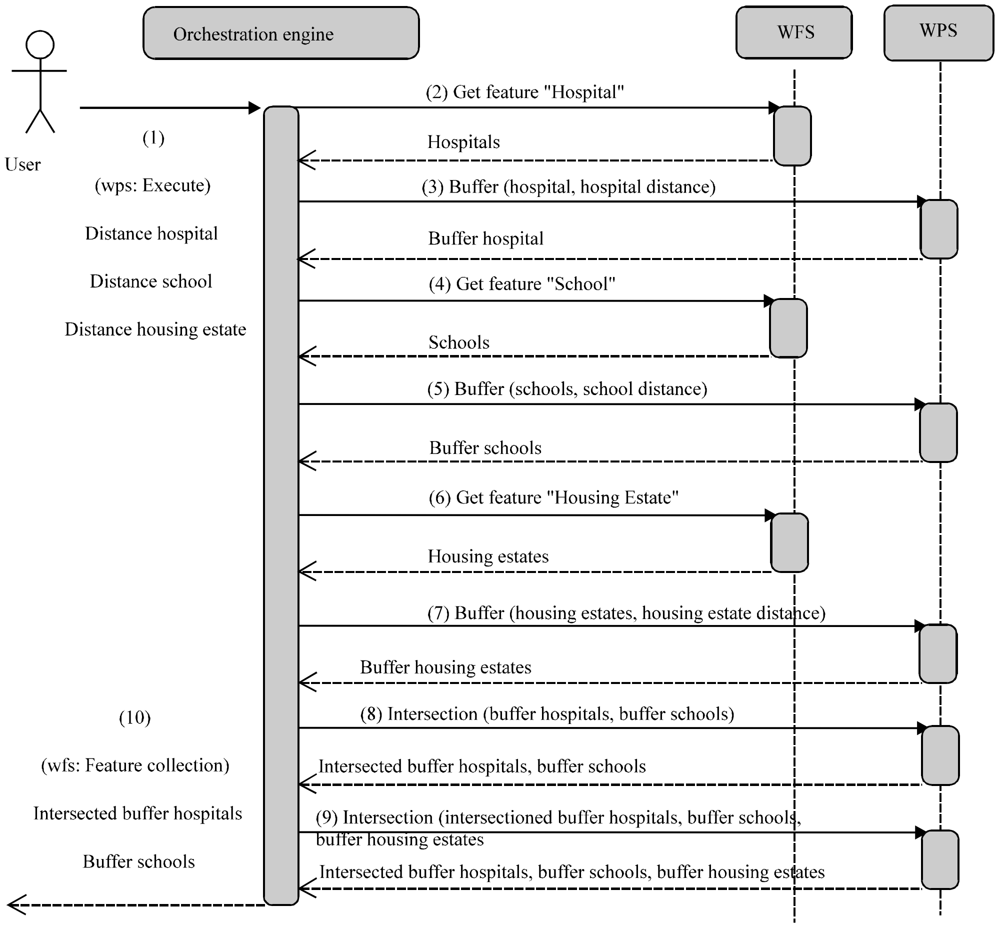

Orchestration engine
WFS
WPS
User
(1)
(wps: Execute)
Distance hospital
Distance school
Distance housing estate
(2) Get feature "Hospital"
Hospitals
(3) Buffer (hospital, hospital distance)
Buffer hospital
(4) Get feature "School"
Schools
(5) Buffer (schools, school distance)
Buffer schools
(6) Get feature "Housing Estate"
Housing estates
(7) Buffer (housing estates, housing estate distance)
Buffer housing estates
(8) Intersection (buffer hospitals, buffer schools)
Intersected buffer hospitals, buffer schools
(9) Intersection (intersectioned buffer hospitals, buffer schools, buffer housing estates
Intersected buffer hospitals, buffer schools, buffer housing estates
(10)
(wfs: Feature collection)
Intersected buffer hospitals
Buffer schools

Fig. 2: Site selection sequence diagram

diagram for describing site selection workflow that shows how processes operate with one another and in what order. A number of process actors which is OWSs involving site selection process. WFS web service is used to publish the location and information of hospital, school and housing estate at Bangkok, Thailand. WPS processing services are used to provide fundamental GIS functions i.e., Buffer and Intersect for decision making. The orchestration engine itself is also provided through WPS interface.

The sequence diagram is demonstrating the processes in the order from top to bottom (Fig. 2). The engine receives the three buffering distances of hospital, school and housing estate from client request.

Then the engine starts to retrieve features that represent the location of hospital, school and housing estate from WFS service. Each feature group has to buffer at WPS service with user specific distance. After finishing the buffering three location features, the three polygon groups will be intersected at WPS in the last step.

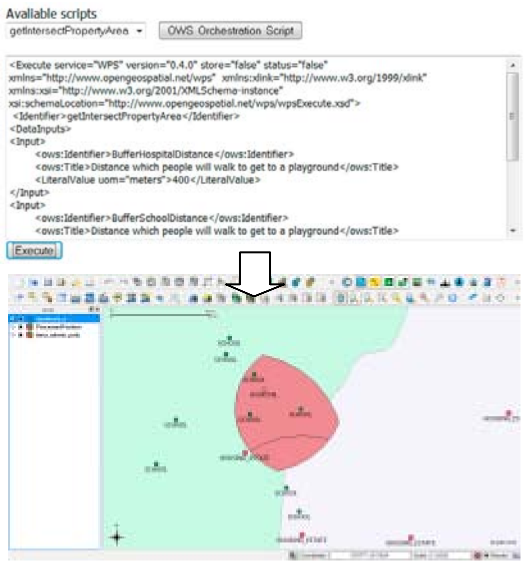
The additional processing service such as GML to Shape File conversion is appended to a script. That “gml2shp” operation is provided through Web Service interface via SOAP transport protocol. The “gml2shp” operation which is the public GRASS We Service from GeoBrain (Yue *et al*., 2009) is attached in order to demonstrate the cross-interface orchestration ability.

The final shorthand script of “site selection” process is written as follows:

```
cuwps:ProcessDefinition name=”SiteSelectionInShapeFile”>
<Variables>
…
<!— OGC WFS request document -->
<Variable name="getHospital" portType="wfs:GetFeatureType">
   <wfs:GetFeature service="WFS" version="1.1.0" outputFormat="GML2">
      <wfs:Query typeName="cuwps:landmark">
         <ogc:Filter>
            <ogc:PropertyIsEqualTo>
               <ogc:PropertyName>LOC_TYPE</ogc:PropertyName>
               <ogc:Literal>HOSPITAL</ogc:Literal>
            </ogc:PropertyIsEqualTo>
         </ogc:Filter>
      </wfs:Query>
   </wfs:GetFeature>
</Variable>
<!— W3C SOAP request document -->
<Variable name=”soapGetShapeFile” portType=”soap:Envelope”/>
   <soap:Envelope>
      <soap:Body>
         <geob:GML2SHPElement
            xmlns:geob="http://Vector_GML2SHP.grass.ws.laits.gmu.edu">
            <geob:sourceURL>{$urlToGml}</geob:sourceURL>
         </geob:GML2SHPElement>
      </soap:Body>
   </soap:Envelope>
 </Variable>
</Variables>
<Activities>
<!-- Catch Service Exception -->
<faultHandlers>
 <catch faultName=”ows:InvalidParameterValue”>…</catch>
 <catch faultName=”soapenv:Server.userException”>…</catch>
 <catchAll>…</catchAll>
</faultHandlers>
<Sequence>
    <Receive operation="wps:Execute" outputVariable="wpsExecuteRequest" />
    <!—Assign User Specific Distance into WPS:Execute Document -->
    <Assign name=”assignUserHospitalDistance”>…</Assign>
    <Assign name=”assignUserSchoolDistance”>…</Assign>
    <Assign name=”assignUserHousingEstateDistance”>…</Assign>
    <!— Invoke services through OGC’s interface standard -->
    <!-- GetFeature and Buffer -->
    <Invoke operation="wfs:GetFeature" inputVariable="getHospital"
    outputVariable="responseHospital" port="http://161.200.86.131/geoserver/wfs" />
    <Assign name=”assignHospitalToBuffering”>…</Assign>
    <Invoke  operation="wps:Execute" inputVariable="getBufferingHospital"
         outputVariable="bufferedHospital" port="http://161.200.86.131/geoserver/wps"/>
    <Invoke  operation="wfs:GetFeature" inputVariable="getSchool"
         outputVariable="responseSchool" port="http://161.200.86.131/geoserver/wfs"/>
<Assign name="assignSchoolToBuffering">…</Assign>
<Invoke  operation="wps:Execute" inputVariable="getBufferingSchool"
         outputVariable="bufferedSchool" port="http://161.200.86.131/geoserver/wps"/>
<Invoke  operation="wfs:GetFeature" inputVariable="getHousingEstate"
```

Continued:

```
          outputVariable="responseHousingEstate" port="http://161.200.86.131/geoserver/wfs"/>
      <Assign name="assignHousingEstateToBuffering">…</Assign>
      <Invoke  operation="wps:Execute" inputVariable="getBufferingHousingEstate"
          outputVariable="bufferedHousingEstate" port="http://161.200.86.131/geoserver/wps"/>
      <!-- Intersect Features -->
      <Assign name="assignHospitalToIntersect">…</Assign>
      <Assign name="assignSchoolToIntersect">…</Assign>
      <Invoke  operation="wps:Execute" inputVariable="getIntersectHospitalSchool"
          outputVariable="intersectedSchoolHospitalPolygon"
          port="http://161.200.86.131/cuwps/wps_intersect.jsp"/>
      <Assign name="assignHospitalSchoolToIntersect">…</Assign>
      <Assign name="assignHousingEstateToIntersect">…</Assign>
      <Invoke  operation="wps:Execute" inputVariable="getIntersectHospitalSchoolHousingEstate"
          outputVariable="intersectedSchoolHospitalHousingEstatePolygon"
          port="http://161.200.86.131/cuwps/wps_intersect.jsp"/>
      <!— Assign input parameter to SOAP -->
      <Assign name="assignGmlToSoap">
          <Copy>
              <From inputValue=" intersectedSchoolHospitalHousingEstatePolygon"><GetRefereceURL/></From>
              <To outputVairable="soapGetShapeFile">{$urlToGml}</To>
          </Copy>
      </Assign>
      <!—Invoke service through W3C's interface standard -->
      <Invoke  name="InvokeSoapWebService"  operation="gml2shp" inputVariable="soapGetShapeFile"
          outputVariable="soapResponse"
          method="POST"
          port="http://geobrain.laits.gmu.edu/axis/services/Vector_GML2SHP">
      <!— Assign SOAPAction -->
      <HTTPHeaders>
            <SetHeader>
                 <Name>SOAPAction</Name><Literal>gml2shp</Literal>
            </SetHeader>
         </HTTPHeaders>
       </Invoke>
       <Reply outputVariable="soapResponse" />
     <Sequence>
   </Activities>
</cuwps:ProcessDefinition>
```



```
<Execute service="WPS" version="0.4.0" store="false" status="false"
xmlns="http://www.opengeospatial.net/wps"  xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengeospatial.net/wps/wpsExecute.xsd">
 <Identifier>getIntersectPropertyArea</Identifier>
<DataInputs>
<Input>
      <ows:Identifier>BufferHospitalDistance</ows:Identifier>
      <ows:Title>Distance which people will walk to get to a playground</ows:Title>
      <LiteralValue uom="meters">400</LiteralValue>
</Input>
<Input>
      <ows:Identifier>BufferSchoolDistance</ows:Identifier>
      <ows:Title>Distance which people will walk to get to a playground</ows:Title>
```

Fig. 3: WPS request for executing "site selection" process with three distance parameters (http://161.200.86.131/cuwps/)

Execution: Standard OGC WPS is implemented to demonstrate the cross-interface orchestration approach. Client can execute pre-defined orchestration script by wps:Execute operation. The script will be executed as a single process activity in the client point of view. The process chain is hidden behind the client visibility.

In addition, user can load a pre-defined script for reading and understanding the process sequence through GetScript function with specified process identifier. The execution and result of the executed "site selection" script is demonstrated in Fig. 3.

CONCLUSION

A geospatial process workflow which is defined as orchestration script, "site selection", is executed through OGC WPS service. The cross-organizational geospatial services are chained to produce a single spatial solution. The designed language is verified that it can be used to orchestrate a cross-interface geospatial services over HTTP transport protocol.

The process sequence of the script can be read and understood by another person. The "site selection" process can be extended in a number of ways such as defining more process (service), or included in other workflow as a single service actor in other domain.

Other requirements that we found from the investigation and the experiment of public geo-processing workflow, are that the geo-processing requires more complex components to be included in orchestration service such as process status, process monitoring, asynchronous orchestration, notification model and so on. These components are required to be addressed from various geosciences researches and geospatial services implementations.

ACKNOWLEDGMENT

This research was supported by the 90th Anniversary of Chulalongkorn University Fund (Ratchadaphiseksomphot Endowment Fund).

REFERENCES

Alameh, N., 2003. Chaining Geographic Information Web Services, in Internet Computing. IEEE Computer Society, New Jurcy, USA.

Amirian, P. and A.A. Alesheikh, 2008. A hybrid architecture for implementing efficient geospatial web services: Integrating NET Remoting and web services technologies. J. Applied Sci., 8: 730-742.

Benatallah, B., F. Casati, D. Grigori, H.R.M. Nezhad and F. Toumani, 2005. Developing adapters for web services integration. Proceedings of the International Conference on Advanced Information Systems Engineering, (CAISE'05), Porto, Portugal, pp: 1-15.

DeRemer, F. and H. Kron, 1975. Programming-in-the large versus programming-in-the-small. Proceedings of the International Conference on Reliable Software, (CRS'75), New York, USA., pp: 114-121.

Di, L., 2004a. Distributed geospatial information services-architectures, standards and research issues. Proceedings of the International Archives of Photogrammetry, Remote Sensing, and Spatial Information Sciences, (APRSSIS'04), New York, USA., pp: 1-7.

Di, L., 2004b. GeoBrain-A web services based geospatial knowledge building system. Proceedings of the NASA Earth Science Technology Conference, Sept. 14, George Mason University, Washington, DC., USA., pp: 22-24.

Einspanier, U., M. Lutz, K. Senkler, I. Simonis and A. Sliwinski, 2003. Toward a Process Model for GI Service Composition. GIDays, Muenster, Germany, pp: 1-16.

Fleuren, T. and P. Muller, 2008. BPEL workflows combining standard OGC web services and grid-enabled OGC web services. Proceedings of the 34th Euromicro Conference on Software Engineering and Advanced Applications, (CSEAA'08), Parma, Italy, pp: 337-344.

Florescu, D., A. Grunhagen and D. Kossmann, 2003. XL: An XML programming language for web service specification and composition. Comput. Networks, 42: 641-660.

Gone, M. and S. Schade, 2008. Towards semantic composition of geospatial web services-using WSMO in comparison to BPEL. Int. J. Spatial Data Infrastructures Res., 3: 192-214.

Granell, C., L. Diaz and M. Gould, 2010. Service-oriented applications for environmental models: Reusable geospatial services. Environ. Modelling Software, 25: 182-198.

ISO19119, 2001. Geographic information-services. NASA/GST, Inc., http://www.digitalearth-isde.org/cms/upload/2007-07-27/DE_A_275.PDF.

Ioup, E., B. Lin, J. Sample and K. Shaw, 2008. Chapter 4: Geospatial Web Services: Bridging the Gap Between OGC and Web Services. In: Geospatial Services and Applications for the Internet, Sample, J.T., K. Shaw, S. Tu and M. Abdelguerfi (Eds.). Springer, Science+Business Media, New York.

Kadry, S. and K. Smaili, 2007. A solutions for authentication of web service users. Inform. Technol. J., 6: 987-995.

Kiehle, C., 2006. Business logic for geoprocessing of distributed geodata. Comput. Geosci., 32: 1746-1757.

Lee, C. and G. Percivall, 2008. Standards-based computing capabilities for distributed geospatial applications. IEEE Comput. Soc., 41: 50-57.

Nabi, M.S.A., M.L.M. Kiah, B.B. Zaidan, A.A. Zaidan and G.M. Alam, 2010. Suitability of using SOAP protocol to secure electronic medical record database transmission. Int. J. Pharmacol., 6: 959-964.

OASIS, 2007. Web services business process execution language version 2.0. OASIS Standard. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html.

OGC, 2003. OWS 1.2 SOAP experiment report. Open Geospatial Consortium Inc. http://portal.opengeospatial.org/files/?artifact_id=1337

OGC, 2005. OWS 2 common architecture: WSDL SOAP UDDI. Open Geospatial Consortium Inc. http://portal.opengeospatial.org/files/index.php?artifact_id=8348.

OGC, 2008a. OGC reference model (OGC 08-062r4). Open Geospatial Consortium Inc. http://grupo.unavirtual.una.ac.cr/mahara/artefact/file/download.php?file=4921&view=961.

OGC, 2008b. Open GIS wrapping OGC HTTP-GET and-POST services with SOAP-discustion paper (OGC 07-158). Open Geospatial Consortium Inc.

OGC, 2009. OGC OWS-6 geoprocessing workflow architecture engineering report (OGC 09-053r5). Open Geospatial Consortium Inc.

Peltz, C., 2003. Web services orchestration: A review of emerging technologies, tools and standards. Technical Report, Hewlett Packard. http://itee.uq.edu.au/~infs3204/interesting_websites/WSOrchestration.pdf.

Sancho-Jimenez, G., R. Bejar, M. Latre and P. Muro-Medrano, 2008. A method to derivate SOAP interfaces and WSDL metadata from the OGC web processing service mandatory interfaces. Adv. Conceptual Model. Challenges Opportunities, 5232: 375-384.

Schaeffer, B., 2008. Towards a Transactional Web Processing Service (WPS-T). GIDays, Muenster, Germany, pp: 1-27.

Stollberg, B. and A. Zipf, 2008. Geoprocessing services for spatial decision support in the domain of housing market analyses experiences from applying the OGC web processing service interface in practice. Proceedings of the 11th AGILE International Conference on Geographic Information Science, (CGIS'08), University of Girona, Spain, pp: 1-10.

Tsalgatidou, A. and T. Pilioura, 2002. An overview of standards and related technology in web services. Distrib. Parallel Databases, 12: 135-162.

Wang, W., Q. Zeng, X. Hao, X. Tang, N. Xie and X. Yang, 2010. Development of an agricultural spatial information sharing platform for supporting user personalization. Inform. Technol. J., 9: 927-934.

Weiser, A. and A. Zipf, 2007. Web Service Orchestration of OGC Web Services for Disaster Management. In: Geomatics Solutions for Disaster Management, Li, J., S. Zlatanova and A.G. Fabbri (Eds.). Springer Berlin Heidelberg, New York, USA., pp: 239-254.

Yan-Ping, C. and L. Zeng-Zhi, 2007. E-WsFrame: A framework support QoS driven web services composition. Inform. Technol. J., 6: 390-395.

Yanfeng, S., M. Xiujun, X. Kunqing, C. Guanhua, L. Chen, L. Chenyu and C. Zhuo, 2005. A compensation mechanism in GIS Web service composition. Proceedings of the Geoscience and Remote Sensing Symposium, (IGARSS'05), Beijing, Chaina, pp: 4-4.

Yue, P., J. Gong, L. Di, J. Yuan, L. Sun and W. Wang, 2009. GeoPW: Towards the geospatial processing web. Web Wireless Geog. Inf. Syst., 5886: 25-38.

Zhao, P., G. Yu and L. Di, 2007. Geospatial Web Services. In: Emerging Spatial Information Systems and Applications, Hilton, B.N. (Ed.). Idea Group Publishing, Hershey, PA, USA., pp: 1-35.