

Workflow Enactment of Grid-Enabled Geospatial Web Services

Gobe Hobona, David Fairbairn, Philip James
School of Civil Engineering and Geosciences
Newcastle University, NE1 7RU
email:g.e.hobona@newcastle.ac.uk

Abstract

The Open Geospatial Consortium (OGC) has developed a series of specifications for geospatial web services. In parallel, the grid community has developed the Open Grid Services Architecture (OGSA) to facilitate the development of web services on the Grid. The orchestration of OGC and OGSA web services in workflows is currently limited by a lack of interoperability between these two architectures. This paper discusses the differences in OGC and OGSA web services and proposes an approach for orchestrating these web services in workflows based on the Business Process Execution Language (BPEL).

1. Introduction

The role of geographic data in business and governance has been steadily increasing since the first Geographic Information System (GIS) was developed in the 1960s. A GIS is a computerised tool for solving geographic problems and portraying geographic information. It is estimated that 80 percent of data collected and used in governance contains a spatial reference. By definition, geographic data links place with other attributes. An example of such data includes medical records which may be geo-referenced through postal codes or tax records which may have geographic coordinates attached. Recent advances in satellite-based remote sensing and Global Navigation Satellite Systems (GNSS) have resulted in the rapid collection of high-resolution geographic data, resulting in very large databases. These databases are analysed using geographic operations—an integration of geometric, statistical and other mathematical functions. These operations distinguish GIS from other database-driven graphical software in that they are aware of geographic coordinate systems, a fundamental aspect of any GIS.

It is the role of the geospatial community to collect, manage and use geographic data. To facilitate the use and sharing of such data, the geospatial community implemented specifications for improving the interoperability of GIS. An organisation involving multi-national governmental agencies, corporations

and universities was formed and tasked with facilitating the development of these specifications; the organisation is called the Open Geospatial Consortium (OGC). Different working groups within the OGC focus on a variety of specialist areas such as geospatial sensors, semantics, coordinate systems, metadata discovery, defence and several other areas of interest. By agreeing on standards for encoding geographic data and interfaces for transporting such data between heterogeneous GIS, the OGC has helped grow the geospatial industry into a multi-billion dollar industry. Some of the specifications relate to web services — a software system designed to support interoperable machine-to-machine interaction over a network [1]. OGC web service specifications formalise some of the different parameters that are expected in most interactions between GIS, for example geographic coordinate systems. This paper refers to OGC web services as geospatial web services.

The Grid is a vision of a global network of hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to distributed high-end computational capabilities and data sources [2]. To facilitate the development of the Grid, the grid community has published a specification for an Open Grid Services Architecture (OGSA). This architecture is based on the web services embodiment of Service Oriented Architecture (SOA) and, therefore, inherits

SOAP and Web Service Description Language (WSDL). These specifications formalise the syntax for describing and binding web services. An additional concern within OGSA is the access and management of resources; where a resource is defined as an identifiable logical entity with zero or more properties expressible in eXtensible Markup Language (XML) [3]. Unfortunately, OGC and OGSA web services are not, at present, interoperable. A related study on the grid-enabling of OGC web services is presented by Di [4].

The OGC services architecture, identical to international standard ISO19119, offers a classification of abstract geospatial services. These geospatial services include human interaction services, model/information management services, workflow/task management services, processing services, communication and system management services. Specifications for information management services are arguably the most mature from the suite of OGC specifications. These include feature, map and coverage services. Although the ISO19119 recognises the potential benefit of geospatial workflows, specifications for standardising their implementation within geospatial information networks have not yet been developed. In contrast, the Organisation for the Advancement of Structured Information Standards (OASIS) — a consortium of computing companies from a variety of disciplines — is developing a specification, called the Business Process Execution Language (BPEL), for supporting the development of service-based workflows. This paper considers an approach for the development of workflows of grid-enabled OGC web services.

2. Geospatial Web Services

Three of the most popular geospatial web services include web feature services (WFS), web map services (WMS) and web coverage services (WCS)[5]. These web services are primarily for the dissemination of vector and raster geographic information. The ISO19119 classifies them as being information management services. This is in contrast to web processing services (WPS), a recently approved specification, that offer processing and analytical functions. A processing service does not offer persistent storage or data transfer functionality[6]. To improve interoperability between web services, the Geography Markup Language (GML) was published as a standard

for serialising the payload of a vector geographic dataset in XML. The OGC has developed other specifications for describing the colour schemes of maps and for encoding filters for queries to geospatial web services. However, this paper is concerned with only the aforementioned information management services.

WFS offer a *getFeature* operation which can be annotated with spatial or comparison filters for defining constraints for a query. The response from a WFS operation is encoded in GML. Another operation offered by WFS is the *DescribeFeatureType* operation which presents the available attributes (fields) and their data types (for example, string, integer, double and so on) as an XML schema definition file (XSD). In contrast, WCS are intended for handling raster geospatial data. A *getCoverage* operation is used for retrieving a raster dataset from WCS. Unlike their vector counterparts though, there are a variety of data types that a response from WCS can be encoded as, for example, as a PNG, JPEG or TIFF image. Alternatively, the response from a WCS can be encoded as an ASCII text file. Unsurprisingly, support for each of these file types varies considerably with each WCS implementation.

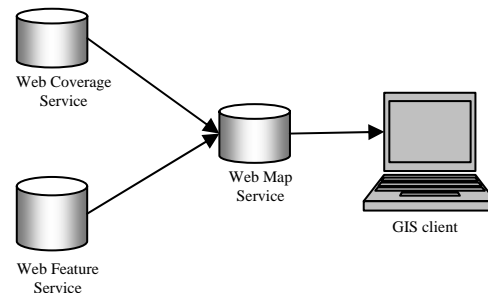


Figure 1 Geospatial Service Chaining (adapted from Alameh, 2003)

Whereas WFS and WCS are primarily for disseminating geospatial data, WMS offer portrayal services. Portrayal services offered by WMS allow a user to specify how to render various layers of geographic datasets onto the same map. Schemes for colours or symbols can be applied to features of specific attributes using a Style Layer Descriptor (SLD). A *getMap* operation is offered for retrieving a rendered map and a *getFeatureInfo* operation is offered for accessing the attributes of a feature at a particular location on the map. Although the response from a WMS is a raster image, it is important to highlight that it is primarily meant

for portrayal whereas a WCS is for disseminating raster data. All the geospatial web services discussed in this section offer a *getCapabilities* operation which lists the layers accessible through that web service and the coordinate systems supported by that web service. Figure 1, adopted from Alameh [7], illustrates how a variety of geospatial web services can be chained together into a geospatial process.

3. Grid-Enabling Geospatial Web Services

Geospatial web services are implemented through specifications different to OGSA services. One of the differences is the encoding of messages. SOAP is the primary message exchange format for OGSA services. In contrast, geospatial web services communicate through a variety of XML (WFS), binary (WMS and WCS) and ASCII text (WCS). OGC XML messages are not based on SOAP; however, the organisation is currently debating support for SOAP for its web service specifications. In addition, there are differences in how the OGC and OGSA offer service description and discovery. A comparison of the different approaches is presented in Table 1.

	OGSA	OGC
Discovery	UDDI, Monitoring and Discovery System (MDS)	Catalogue Services for the Web (CS-W)
Description	WSDL	getCapabilities, describeCoverage, describeFeatureType, describeProcess operations
Messaging	SOAP	Image binary, ASCII text, OGC XML encoding profiles e.g. GML,

Table 1 A comparison of OGSA and OGC service description, discovery and messaging approaches

Hence to grid-enable geospatial web services it is necessary to enable communication between grid-based applications and geospatial web services. Where a grid-based client is the original sender, the geospatial web service container becomes an ultimate receiver. To implement such architecture, an intermediary SOAP node is required to act as a proxy between the geospatial web services and other SOAP nodes, including the original sender. We refer to this intermediary as the proxy web service. In addition to acting as an adapter between grid and geospatial web services, the

proxy web service offers the possibility of incorporating grid-based security for encrypting messages or authenticating clients.

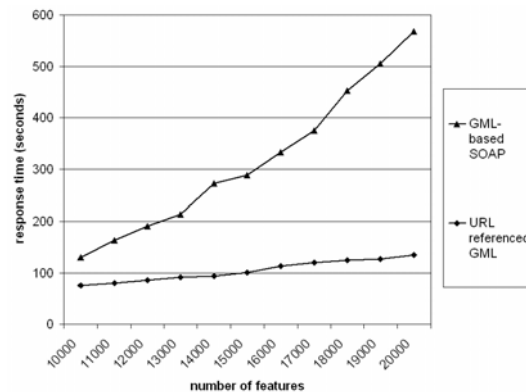


Figure 2 GML-based SOAP messaging versus URL referencing of GML resources

The WFS specification proposes example SOAP bindings for WFS operations. The example WSDL imports WFS schema definition files (XSD) such that a proxy web service can use the schema definitions to serialise outgoing messages (responses) or deserialise incoming messages (requests). By importing WFS schema into its own schema, our proposed proxy web service is then able to send valid requests to WFS. However, the WFS specification also suggests serialising feature collections as GML-based SOAP payloads. This results in a significant load on bandwidth due to the verbose messages being returned from the WFS. Further, the process of serialising and deserialising the SOAP messages imposes a significant load on the web service container. Our study conducted an experiment to determine the performance of exchanging GML-based SOAP messages between a web service container (Globus Toolkit) and a workflow engine (ActiveBPEL). The Globus Toolkit provides a grid service container and is offered by the Globus Alliance (<http://www.globus.org>: last visited 09/01/2007). ActiveBPEL is a workflow engine, developed by Active Endpoints, based on the BPEL specification (<http://www.activebpel.org>: last visited 01/02/2007). The results, presented in Figure 2, show the time taken for a feature collection to be received by a workflow engine from the moment a *GetFeature* request is sent. Based on the results illustrated, we propose persisting the feature collection in a web server and returning only a URL reference to the feature collection. Referencing a feature collection by URL, is consistent with suggestions made by the OGC OWS-4

Geoprocessing Working Group which also discovered performance limitations to the retrieval of feature collections as GML-based SOAP messages.

The proposed structure of requests sent from the service requestor to WFS, WCS and WPS is presented in Figure 3. As illustrated, a request sent from a Proxy Feature Service contains an OGC GetFeature object, which is the request for retrieving feature collections from WFS. Similarly, a Proxy Coverage Service includes a GetCoverage object for requesting a coverage from a WCS. Last, a Proxy Processing Service contains an Execute object for invoking a WPS process. In addition to these objects, we propose the inclusion of a stateless property called 'host' which references the intended geospatial web service. By making the host property stateless, the proxy service is able to dynamically reference any geospatial web service at runtime.

Handling raster datasets offers a different challenge as they are not serialised in XML. Further, there are a variety of different file formats that are considered de-facto standards for encoding raster data. In designing a proxy coverage service, we considered two approaches for returning the output from a WCS, i) where the raster dataset is embedded within a SOAP message as binary, ii) where the SOAP response offers a URL reference to a raster dataset. Despite the available memory (heap size) being set to more than 1GB, raster images larger than 300MB caused the Java runtime environment to run out of available memory. It is possible that a native-based application could have offered better performance. Whereas, a non-Java platform may have offered better performance, an approach that includes URL references to datasets would offer a more platform independent solution. Hence similar to the approach adopted for returning vector datasets from WFS, we adopted SOAP responses with URL references for WCS. For processing services, however, datasets are referenced through URLs; therefore, the responses from WPS can be handled as are without any modification.

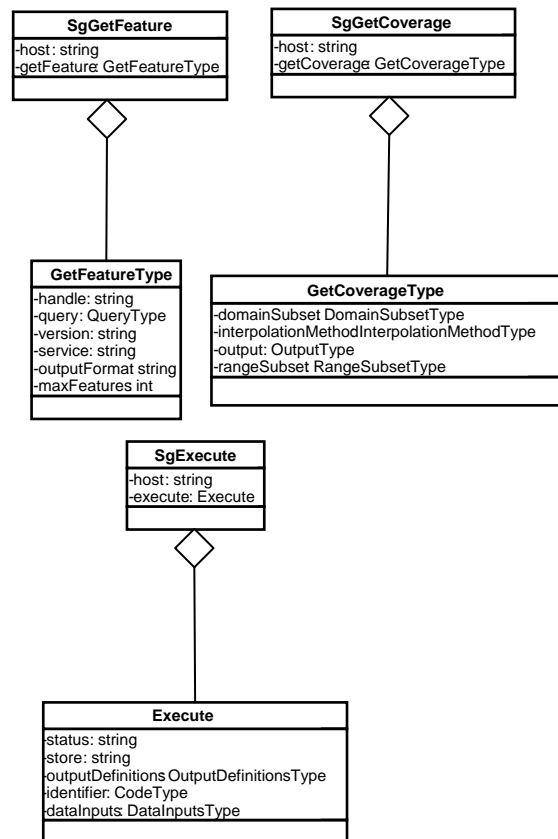


Figure 3 Class diagram of the proposed requests (Sg suffixed) and their OGC counterparts

A feature of grid service architectures is that the availability of a resource can be queried. Such a resource could be available memory, processor speed, connections to a database or any other computational resource. OGC web service specifications offer a *getCapabilities* operation for querying the availability of geospatial datasets, which can be considered resources based on the OASIS definition of a resource. Although the OGC services architecture recognises “system and management services”, the suite of OGC specifications does not cover the management of non-geospatial resources such as system memory. In contrast, OGSA offers Resource Management Services (RMS) for the allocation, reservation, monitoring and controlling of resources. An example of such a service is the Grid Resource Allocation Management (GRAM) module offered by the Globus Toolkit.

4. Workflow Enactment

Activities are the fundamental components of a BPEL workflow. They are responsible for

input, output, service invocation, fault handling, assignment, conditional flow control (i.e. *if-else-while* constructs) and several other functions in a workflow. The OASIS specification classifies them into two main groups 'basic' and 'structured'. Basic activities perform the elemental functions within a workflow. Whereas, structured activities manage control-flow by nesting other basic or structured activities. In addition to activities, a BPEL workflow contains variables. Each invocation activity returns a response message that is bound to a variable that adopts the schema of the response message. The values of the response message are assigned to a corresponding variable. Other variables, though not associated with a web service response, can also be declared. The state of variables persists throughout the lifecycle of the workflow. Therefore, parameters of the variables can be queried or modified at any time during process enactment. Not only are variables used for creating requests and responses, they can also be used for controlling flow. For example, a conditional activity (equivalent to an *if-else* statement) could be based on the value of a variable.

Our study considers the scenario presented in Figure 4. Within this scenario, a client sends a request for vector data to WFS and the WFS returns a feature collection (analogous to a relational table). The feature collection is then forwarded to a WPS, based at Newcastle University, for generalisation (a cartographic process of reducing the number of coordinates in geometry whilst maintaining the general appearance of the shape). The result from the first WPS is then forwarded to a second WPS, based at George Mason University, which clips the dataset based on a specified bounding rectangle. The result from the second WPS is then returned to the original sender, the client. All these activities are controlled by a workflow engine. This scenario is an extension of the OGC OWS-4 experiment on Geoprocessing Workflows. Our study extends the scenario by retrieving a coverage from a WCS and merging the feature and coverage output into a single Keyhole Markup Language (KML) file for visualisation on Google Earth. The KML exporter WPS also renders the coverage from an ASCII text file, illustrated in Figure 7, to a conventional JPEG image.

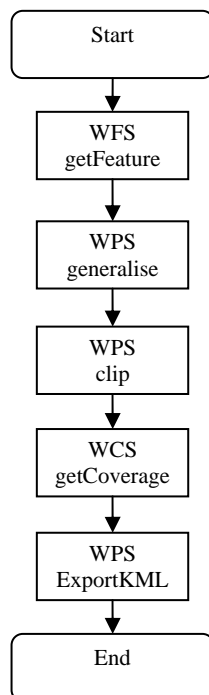


Figure 4 Example scenario

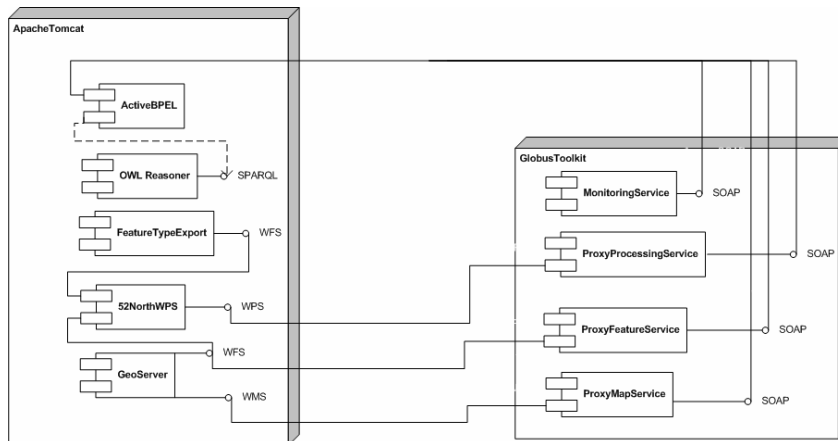


Figure 5 Deployment diagram of the proposed system

5. Implementation

A prototype was implemented using the Globus Toolkit (version 4) and ActiveBPEL (version 3). Although other technologies could have been adopted, these were selected because they are open source software with significant support within the scientific community. Geospatial components within the prototype architecture include Geoserver (<http://docs.codehaus.org/display/GEOS> : last visited 11/04/2007) and the 52 North WPS (<http://52north.org> : last visited 11/04/2007). Geoserver is a geospatial web service container that allows for the creation of WFS, WMS and WCS. The 52 North WPS offers facilities for buffering and generalisation. It also offers a pluggable architecture that allows for additional geoprocesses to be incorporated into the WPS. An illustration of the proposed architecture is presented in Figure 5.

As already observed, returning URLs to persisted feature collections offers better performance than serialising the complete feature collections in a GML-base SOAP message. A question was then how to persist the feature collections, either as local files or in a spatially-enabled Relational Database Management Systems (RDBMS). An example of such an RDBMS is PostgreSQL; through its spatial extension PostGIS [8] it offers an object-relational schema for storing geographic features. Results from a PostGIS query are returned in the Well Known Text (WKT) format [8] which can then be parsed and transformed into GML-based SOAP objects by the proxy feature service. PostGIS offers several geospatial functions that can be invoked on geospatial features through the calling of an SQL statement, for example, calculation of bounding boxes, spatial intersections, buffering

and others. Therefore persisting the feature collection in an RDBMS offered the workflow engine the possibility of incorporating further operations. Once the feature collection has been persisted in a PostGIS database, it then becomes necessary to enable its export as GML such that it can be accessed by WFS. A servlet, named FeatureTypeExport on Figure 5, was implemented to allow for the exporting of feature collections as GML. The feature collection therefore becomes a resource that can be retrieved at anytime during the execution of a workflow.

The example workflow illustrated in Figure 3 was constructed and deployed in an instance of ActiveBPEL. The sample vector data from the WFS is retrieved as GML and the raster data from the WCS is retrieved as an ASCII text file of space-separated pixel values and a single spatially referenced anchor point. Subsets of the input files are presented in Figures 6 and 7. The results of the workflow are presented in Figure 8 as a Google Earth visualisation. A fundamental concern in handling geospatial data is the spatial referencing system of a dataset. The sample vector data adopts the British National Grid (BNG) while the raster data adopts the World Geodetic System 1984 (WGS84). As the resultant dataset was meant for visualisation within Google Earth, it was then necessary to project the vector data from BNG to WGS84. This functionality was included within the processing on the KML exporter WPS. Further, the source raster, needed to be rendered such that it could also be included in the visualisation. A function was added to the KML exporter for determining the maximum and minimum pixel values of the raster dataset and scaling the shade of pixels over a 255 unit greyscale. Hence, the path of a

river is shaded the darkest in Figure 8 and the highest elevations have the lightest shade.

6. Discussion

Although OGSA and OGC web services adopt different XML profiles, their interoperability is significantly improved by the adoption of XML schemas. The schemas made it possible to construct requests on a SOAP-based client and to send the request to a geospatial web service with minimal modifications. However, there were minor variations in the interpretation of the schemas by each SOAP-node in our architecture. For example, some nodes ignored the absence of specific elements within messages, whilst others enforced the addition of other elements.

```
<wfs:FeatureCollection xsi:schemaLocation="http://www.openplans.org/topp
http://localhost/DescribeFeatureType?type=topp:buildings
http://www.opengis.net/wfs http://localhost/geoserver/schemas/wfs/1.0.0/WFS-
basic.xsd">
  <gml:featureMember>
    <topp:buildings fid="buildings.3">
      <topp:the_geom>
        <gml:MultiPolygon srsName="EPSG:27700">
          <gml:polygonMember>
            <gml:Polygon>
              <gml:outerBoundaryIs>
                <gml:LinearRing>
                  <gml:coordinates decimal="." cs="," ts=" " >
423946.36203139,565512.78804144 423943.55166311,565514.76922564
...
423946.60190341,565512.61805742 423946.36203139,565512.78804144
          </gml:coordinates>
        </gml:LinearRing>
      </gml:outerBoundaryIs>
    </gml:Polygon>
  </gml:polygonMember>
</gml:MultiPolygon>
</topp:the_geom>
<topp:TOID>1000030049139</topp:TOID>
<topp:FEATCODE>10021.0</topp:FEATCODE>
<topp:VERSION>2.0</topp:VERSION>
<topp:VERDATE>2001-11-05</topp:VERDATE>
<topp:THEME>Buildings</topp:THEME>
<topp:CHANGE>1988-07-28 New</topp:CHANGE><topp:PHYSPRES>Low
Density Private</topp:PHYSPRES>
<topp:BROKEN>0.0</topp:BROKEN>
<topp:OID_>64201.0</topp:OID_>
</topp:buildings>
</gml:featureMember>
</gml:featureMember>
.....
</gml:featureMember>
</wfs:FeatureCollection>
```

Figure 6 Subset of the input GML document

We attribute the variation in interpretation of schemas to the different XSD compilers adopted by the Globus Toolkit, ActiveBPEL and Geoserver. A key concern, therefore, was ensuring that the proxy service did not introduce errors into messages in-transit. Consequently, the evaluation of the prototype has included the use of TCPMon, a message monitoring program distributed with the Globus Toolkit.

A feature of BPEL workflows is their ability to handle faults through a 'catch' activity. A message type can be declared as being a fault such that it can be handled by a catch activity. OGC schemas define message types for exceptions. A proxy web service, therefore, needs to import an OGC exception schema into

a SOAP-based message. This is achieved similar to other OGC request and response messages, as illustrated in Figure 3. The SOAP-based message is then added as a variable within a BPEL workflow. Further, at any point on the workflow a service can be unavailable. This means that the response from the geospatial web service may not be an XML-based message but a conventional HTTP response code. Such faults would require the proxy web service to identify the fault as an HTTP fault rather than an OGC exception.

```
NCOLS 522
NROWS 300
XLLCENTER -1.6878262291489
YLLCENTER 54.93369175561999
CELLSIZE 3.013533976228086E-4
NODATA_VALUE -9999.0
54 54 53.59999847
53.79999924 53.79999924 53.5 53.5
53.79999924 53.79999924 53.5 53.5
53.79999924 53.79999924 53.5 54
54 54 53.70000076
54 56 54 53.79999924
54 54 54
54.09999847 54.09999847 54 54
54.40000153 54.40000153 54.20000076 52
54.5 53 54.5 54.40000153
54.70000076 54.70000076 54.59999847 53
53.79999924 53.79999924 53.5 53.5
```

Figure 7 Subset of the input raster as text

An additional issue encountered was the variation of GML-based schemas from the included WFS and WPS. Although the output GML was valid, one of the included WPS could not support the output of another. Such a situation requires a shim service [9]. A BPEL workflow engine can address such situations by either offering custom shim functions from within the workflow engine or by forwarding the output to a shim service before continuing a workflow. The former approach limits the workflow to be non-transferable as the custom functions are not available to other workflow engines. The latter approach offers a more application independent solution as the shim services are accessed similar to other web services.

7. Conclusions

This paper has presented a study to develop workflows of grid-enabled geospatial web services. An architecture is proposed and a prototype developed. The prototype includes the Globus toolkit, Geoserver and ActiveBPEL. These technologies were integrated for hosting web services and workflows. An example workflow was constructed to test the prototype; this workflow included feature, coverage and processing services. The results of the workflow are exported in KML for visualisation in Google Earth. We conclude that BPEL-based

workflows can support grid-enabled geospatial web services.

In conducting this study, a number of issues were encountered. First, whether to serialise the response from feature services in GML or to offer a URL reference to a temporarily persisted GML document. The latter option was adopted as it offered a more efficient solution through smaller SOAP payloads. Similarly, responses from coverage services were returned as URL references to temporarily persisted data. Second, through the adoption of OGC schema, the web services could submit and receive valid geospatial messages. However, due to slight variations in the interpretation of XML schema by the different containers within our example workflow, it was necessary to make minor modifications to the WSDL to allow the service to transmit valid messages between the workflow engine and the geospatial web service.

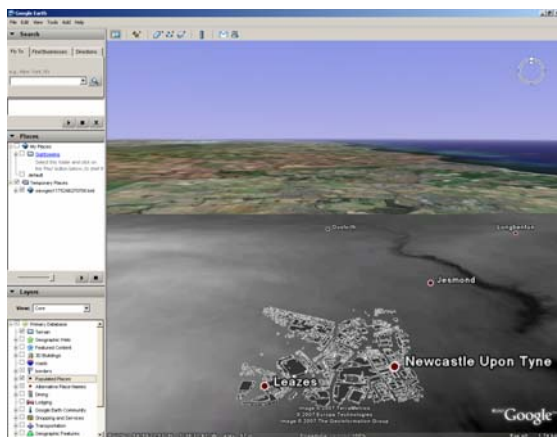


Figure 8 Results of workflow presented in a Google Earth visualisation.
Vector Data - Ordnance Survey © Crown Copyright

Future studies should investigate the possibility of automating workflow construction through semantic technologies. This would ensure that messages passed between SOAP nodes in a geospatial workflow are both syntactically and semantically correct. Related studies have achieved semi-automation [10] and further research is needed to enable applications to completely construct geospatial workflows without human assistance.

8. Acknowledgements

This paper is produced from the SAW-GEO project, a collaboration between the School of Civil Engineering and Geosciences at

Newcastle University and the North East Regional e-Science Centre (NEReSC). SAW-GEO is funded by the United Kingdom's Joint Information Systems Committee (JISC) through the Grid/OGC Collision Programme.

References

- [1] W3C, "World Wide Web Consortium Specifications," 1999-2006.
- [2] I. Foster and C. Kesselman, "Computational Grids," in *The Grid: Blueprint for a new computing infrastructure*, I. Foster, Ed. San Francisco: Morgan Kaufmann, 1999.
- [3] OASIS, "Web Services Resource Framework (WSRF)," Organization for the Advancement of Structured Information Standards (OASIS), 2006.
- [4] L. Di, "Geospatial Grid," in *Frontiers of Geographic Information Technology*, S. Rana and J. Sharma, Eds. Berlin: Springer Berlin Heidelberg, 2006, pp. 121-137.
- [5] OGC, "OpenGIS Service Architecture-ISO19119 (Open Geospatial Consortium)," ISO, Ed., 2001a.
- [6] Z.-R. Peng and M.-H. Tsou, *Internet GIS: Distributed Geographic Information Services for the Internet and Wireless Network* New Jersey: John Wiley & Sons, Inc, 2003.
- [7] N. Alameh, "Chaining Geographic Information Web Services," *IEEE Internet Computing*, vol. 7, pp. 22-29, 2003.
- [8] PostgreSQL Global Development Group, "PostgreSQL Documentation." vol. 2006: <http://www.postgresql.org/>, 2005.
- [9] D. Hull, R. Stevens, P. Lord, C. Wroe, and C. Goble, "Treating semantic web syndrome with ontologies," in *First Advanced Knowledge Technologies workshop on Semantic Web Services (AKT-SWS04)* vol., KMi, The Open University, Milton Keynes, UK, 2004.
- [10] R. Lemmens, A. Wytzisk, R. d. By, C. Granell, M. Gould, and P. van Oosterom, "Integrating Semantic and Syntactic Descriptions to Chain Geographic Services," *IEEE Internet Computing*, vol. 10, pp. 42-52, 2006.