

Removing Syntactic Barriers for Semantic Geospatial Web Services

Xuan Shi

West Virginia University

Abstract

Web services are technologies designed mainly to eliminate the technical problems of interoperability among diverse applications and heterogeneous development platforms. Web services enable GIS software to be decomposed into functional components that can be accessed and integrated through the standard Web Services Description Language (WSDL) interface. However, Web service infrastructure has been criticized as providing descriptions only at the syntactic level, making the interpretation of meaning or semantics difficult. Particularly, such syntactic barriers are a major impediment to the building of semantic geospatial Web services. This paper will address issues on how the syntactic architecture of WSDL Web services prohibits computers from automatically deriving service descriptions and semantics and how ontology-based “semantic request and response” approach can simplify and standardize WSDL Web services for geospatial data processing and service integration. Additionally, the paper demonstrates how the implementation of intelligent and semantic Web services can be enabled. Defining service description and semantics, however, remains a challenge for further discussion.

Introduction

Web services promise a new level of interoperability between applications that will eliminate mainly the technical problems. The fundamental goal of interoperability in Web services is to blur the lines between the various development environments used to implement services so that developers using those services do not have to think about which programming language or operating system the services are hosted on (Snell, 2002). The mechanism for the interoperability via web service is that each service encapsulates specific business functionality that can be invoked through SOAP as a common protocol over HTTP. SOAP uses serializer and deserializer objects to translate from the native language of a software application to the SOAP protocols that move the request over the wire. The most useful functionality offered by SOAP implementations is the ability to convert data between a platform-specific format and the XML format used in SOAP messages. It is this functionality that allows an application running on one platform to convey information to an application running on another platform (Deem, 2002).

Semantic Web services can be defined as “the augmentation of Web Service descriptions through Semantic Web annotations, to facilitate the higher automation of service discovery, composition, invocation, and monitoring in an open, unregulated, and often chaotic environment” (Payne, et al., 2004). Web service infrastructure (WSDL, UDDI and SOAP) has been criticized in almost all papers on the topic of “Semantic Web

Services”, typically as “existing technologies for Web services only provide descriptions at the syntactic level, making it difficult for requesters and providers to interpret or represent non-trivial statements such as the meaning of inputs and outputs or applicable constraints” (Cabral, et al., 2004). Adding rich semantics into Web Services has been a main research initiative in the IT community. Current approaches for Semantic Web Service infrastructures are developed by a variety of research groups, among those efforts, OWL-S (Martin, D., et al., 2004), WSMO (Roman, et al., 2004), etc. were the most recognized and outstanding achievement to date in this field. However, all such efforts tried to create supplementary mechanisms, which focused on describing the semantics and relationship of the names of service, function, input and output which formulate the syntactic architecture and hierarchy of Web services. All such extra mechanisms have to be correlated with the WSDL interface which formulates the computing semantics.

Web service application in the GIS domain has been changing the scenario of semantic Web service research and development. New problems and questions have been arising in this process since geospatial data structure and geoprocessing functions are more complex than those simple business models and transactions. While WSDL has been the source for other approaches to derive service semantics, it has been demonstrated (Shi, 2004) that WSDL cannot be used as the source to derive the semantics in both existing and prospective geospatial Web services. If the goal of semantic Web services is to be feasible, implementable and extensible to every different service domains, the building mechanism has to be applicable universally.

This paper will first discuss the syntactic barriers within WSDL file that prevented the creation of semantic geospatial Web services. It demonstrates that the semantic request and response approach not only can transform the composite Web services into atomic ones but also set up the true semantic service contract between service providers and requesters. By separating service contract from the WSDL file, we can then focus on the semantics inside such contract. On the one hand, adding service domain ontology and functional category into the semantic template of service description will enable the developers to understand which kind of service and functions are provided. On the other hand, adding formal ontology onto data description will set up the correlation of different domain ontology to define the service semantics in a more standardized form. In this way, the syntactic barriers can be removed to build the intelligent and semantic Web services in general, and geospatial Web services in particular.

The syntactic barriers for Semantic Web services

According to W3C (2004a), WSDL defines service description about the message formats, data types, transport protocols, and transport serialization formats as a machine-processable specification of the Web service's interface. The semantics of a Web service is the shared expectation about the behavior of the service, and, is the “contract” between the service requester and the service provider regarding the purpose and consequences of the interaction. “While the service description represents a contract governing the

mechanics of interacting with a particular service, the semantics represents a contract governing the meaning and purpose of that interaction”.

While “a service description may include a description of the service's semantics”, “a service semantics should be identified in a service description” (W3C, 2004a). In practice, however, “knowing the type of a data structure is not enough to understand the intent and meaning behind its use” (W3C, 2004a). Further, WSDL can only expose such syntactic architecture of the service but prevents the service requester from understanding the service semantics. ESRI’s ArcWeb Services are good examples as demonstrated (Shi, 2004) that the content of WSDL file of AddressFinder Web services is full of puzzle and tricks. This means the service semantics may not be identified in the service description, but on the contrary, a well-defined service description can still hinder the derivation of service semantics.

Consider the following example of a Web service to perform a geospatial data processing function that create a new buffer feature around a given polygon feature at a user specified distance. The functional interface can be described as the follows:

Function ***bufferPolygonFeature*** (String: authenticationCode, String: polygon, double: distance, String: unit) String: bufferPolygon

Although the function statement looks like an atomic Web service with meaningful service description about the function and input/output variables, the service semantics may still not be identified in the service description file. Possible problems can be enumerated as follows:

1. Embedded or composite Web services: in the business domain, it is a common feature that one Web service function or object data type can be shared by the other services generated by the same service provider, such as ESRI’s ArcWeb Service. For example, the user may need to include an authentication code to invoke a paid service, while this authentication code is dynamically generated by the authentication Web services through a functional interface:

Function ***getAuthentication*** (String: username, String: password) String: authenticationCode

In the above example, the polygon feature as an input variable can either be uploaded from the requester’s local machine, or it can be obtained by invoking another Web service that provides sources for these data through functional interface:

Function ***getPolygon*** (String: featureName) String: polygon

However, the computer itself does not understand such a service behavior. What computer can do, if the preconditions are satisfied, is to invoke the function and return the outcome. If the preconditions cannot be satisfied, the computer then

cannot understand what to do and how to do it. Even for human beings, it is difficult to find such embedded Web services just by reading such service description files. We have to find some external information and reference resources to understand how to invoke the *bufferPolygonFeature* Web services.

2. The semantics of the polygon: what is a polygon? Is it in the form of a simple geometry or a georeferenced one? For geospatial features, we need to be concerned what kind of coordinate system is used to define each vertex of the polygon. It may be geographic coordinate such as longitude and latitude, or it may be in a projected system such as UTM, or state plane, together with the measurement datum. However, the service description cannot describe such service semantics.
3. The semantics of units: what kind of options can be used to measure the distance, e.g. millimeter, centimeter, meter, kilometer, inch, feet, mile, etc? Such question is also based on the common sense supposition that the user is able to understand that the unit is used for measuring distance and not for some other measurement. However, in a varied application domain, the units may be used for other purposes, such as measuring temperature, precipitation, pressure, weight, etc. The user cannot differentiate the meaning of the unit by its name.
4. Not all functions and input/output variables can be defined explicitly and meaningfully. For example, a geospatial function that performs a query based on spatial selection: select features in polygon 1 that are contained in polygon 2. How can we name the function for a semantic service description? Can we name it as “selectionPolygon” or “polygonSelection” or whatever? Maybe there is no perfect name to enable service requesters to understand clearly what this function does exactly. The names of the input/output variables maybe meaningless also since the polygon itself is not defined comprehensively.
5. How is the relationship of the input variables defined? That is to say, how can we describe the service so that the user can understand such relationship between the two input polygons as described in such sample function that selects features in polygon 1 that are contained in polygon 2? If the requester changes the order of the input variables (polygon 2 first and then polygon 1), the result will be wrong.
6. Derivable meaning of the input variables: GIS data have certain implied information. In the example of case 2, the GIS may use a projected coordinate system (e.g. UTM Zone 17 North) with the NAD83 (North American Datum 1983) datum. This means the measurement units are meters. The same data, however, can be described in the geographic coordinate system, which means the measurement units are longitude and latitude (degree/minute/second). In another coordinate system such as the West Virginia State Plane with NAD83 datum, the measurement units are U.S. feet. If we want to process these different data source, the service provider has to provide semantic and meaningful descriptions about the requirements of the input polygon so that the data in different spatial reference

systems can be converted into equivalent coordinate systems with the same datum before processing. Otherwise, the results will be incorrect.

7. Industrial or domain-specific standards: pre-defined EPSG (European Petroleum Survey Group) projections have been used as a *de facto* standard to describe the spatial reference systems used for GIS data, e.g. EPSG:4326 refers to WGS84 (World Geodesic Datum), EPSG:26917 refers to NAD83 / UTM zone 17 North, and EPSG:26717 refers to NAD27 / UTM zone 17 North, etc. these standards are commonly understood in the GIS community, however, other professionals may be confused by such domain-specific codes or standards.
8. Document based input/output variables: *particularly*, as a common feature of GIS applications, vector GIS data is described in an XML document, such as GML (Geographic Markup Language). By reading service description file, the user can only know the name of the input variable as a “polygon” or “polyline”, supposing both service provider and service requester are dealing with geospatial services. However, the information inside WSDL cannot describe the content of the XML documents.

In conclusion, WSDL is **NOT** the appropriate source from which to derive service semantics. If there is any semantics inside the service description file, (i.e. WSDL), such semantics is the name of service, functions and input/output variables. Moreover, such a naming system is based on common sense. Since the service providers want to make the service meaningful to the service requesters, they intentionally name the functions and variables. Such common sense behavior, however, is unreliable since the name can be assigned or changed without any rules and specifications. For example, we can name the function *bufferPolygonFeature* or *CreateBufferPolygon* and it will perform the same function. We can name the polygon *bufferPolygon* or *buffPolygon* or *bufferPoly*. This is a common feature for all existing Web services, that is to say, we can randomly change the name of functions and variables and all such Web services will work normally. Because the name has no semantic implications, such a service description cannot be used to generate semantic Web services.

On the other hand, WSDL is full of redundant and irrelevant information for service requesters. It has been demonstrated (Shi, 2004) that WSDL itself could not explicitly describe the content and meaning of a Web service. It is difficult for service requesters to trace the object diagrams or the supplementary descriptions to figure out the implementation details of services, hidden services, and shared common object data types used in multiple services. Whether by modeling the complexTypes (the collection of elements in WSDL schema) into class-subclass hierarchy with properties, or by adding semantics onto WSDL file, such approaches have to describe the service provider’s design and implantation details which are not necessary to the service requester who “just wants to know the x, y values rather than the implementation details on the server side” (Shi, 2004). However, before the service requesters can get the x, y coordinates, they have to figure out how to derive x and y values from varied object data types, such as LocationInfo, ArrayOfLocation, Location, Point, Envelope, etc.

Transforming composite Web services into atomic ones

Adding rich semantic descriptions into Web Services is expected to “support greater automation of service selection and invocation, automated translation of message content between heterogeneous interoperating services, automated or semi-automated approaches to service composition, and more comprehensive approaches to service monitoring and recovery from failure” (Martin, D., et al., 2004). In the OWL-S (2004) process ontology, Web services can be modeled as three kinds of processes: atomic processes, composite processes and simple processes. Atomic processes are single web services, which are executed by passing them particular input values. Composite processes are executed compound Web Services that can be de-composed into atomic Web Services, or further composite Web Services. While composite Web services have created difficulties for implementing semantic Web services, the semantic request and response approach (Shi, 2004) enables the transformation of composite Web services into atomic ones.

Considering the composite Web service *bufferPolygonFeature*, the semantic request and response approach will transform all Web service transactions that are processed by the same service provider into an atomic Web services. Traditionally, service providers do not pay much attention at the development stage to how the service requester can understand and use the service. The design may look reasonable from the viewpoint of object-oriented approach and the provider’s business logic, but the result may be a puzzle full of tricks to the service requester. Since the service description file does not contain all of the necessary information for requesting services, the service requesters must do at least one of the following to understand how to invoke the Web service (Shi, 2004):

- Know the implementation details about how the Web service is constructed.
- Manipulate the WSDL files.
- Find the necessary information beyond the contract.
- Trace the object Diagram of the service.
- Learn from the samples.

Figure 1 shows such an example as from WSDL 3: the requester understands that a string named as “authenticationCode” is required but the requester does not understand where this “authenticationCode” comes from.

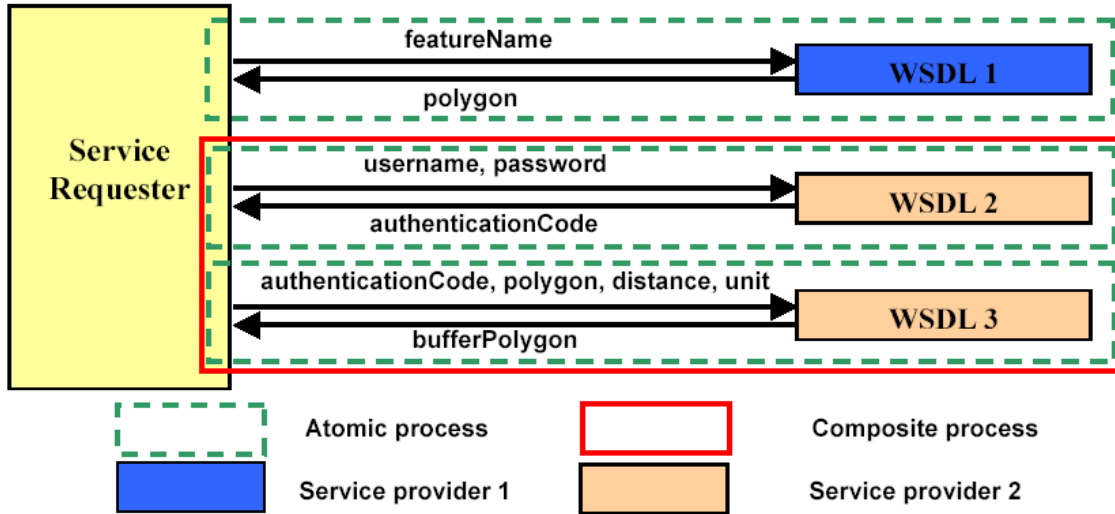


Figure 1. Atomic vs. composite Web service invocation

In order to build intelligent, semantic Web services, the service provider cannot suppose that the service requester understands how to use the service. Traditionally it is the service requester’s responsibility to chain and invoke different services in order to get the final result. Such a sequence is described in figure 1. The service requester sends a request to service provider 2 to invoke WSDL 2, receives the result, which is then used to invoke WSDL 3. In such composite Web services, at least one of the service descriptions cannot be explicitly defined and described.

The semantic request and response approach requires that the service provider chain and invoke necessary Web services. In this way, Web service can be simplified and standardized into one functional interface - Function `getService(String request): String response` - with one input (semantic request) and one output (semantic response). Both request and response will be XML documents that explicitly describe the required input variables and expected outcomes. By publishing the service description template, the service provider and requester can set up a real semantic and meaningful “contract” (Shi, 2004). In this way, WSDL is only a gateway or tunnel to transfer the XML documents as semantic request and semantic response. When all composite Web services can be transformed into atomic ones, we can then focus on the semantics inside the messages exchanged in the Web services without any relation with WSDL file.

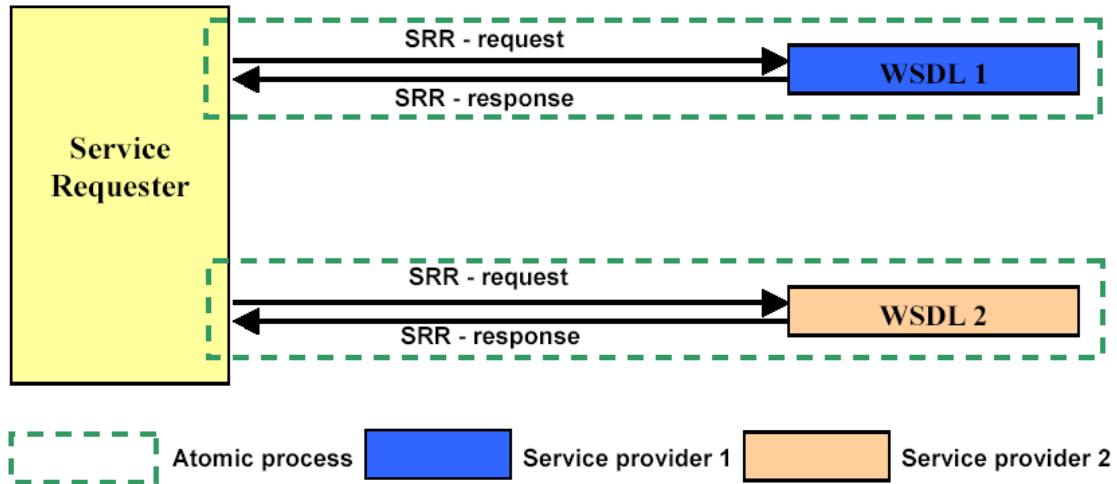


Figure 2. Web service transformation via semantic request and response

Adding formal ontology onto data and service semantics

Transforming composite Web services into atomic ones is the first step towards building semantic Web services. With the rapid development of Web service applications, more and more Web services become available. However, by reading the names of the service, functions and object data types, it is difficult to understand what the purpose of the provided services since such service “semantics” is not well defined based on strict specifications and agreements, but rather, is based on common sense interpretation. Consider for example, *getAuthentication* Web service and function, there may be thousands of such services in the existing Web services. It is difficult for service requesters to understand which *authenticationCode* can be used to invoke which other Web services. Given another example, *getTicket* Web service and function, we cannot understand whether this Web service can be used to get airline ticket, or a subway ticket, concert ticket, movie ticket, or Sea World ticket, etc.

On the other hand, the service description file contains redundant information that is irrelevant to the service requesters. Such redundant and irrelevant information is an unnecessary burden in describing service semantics. By adding ontology specification onto data and service semantics, self-describing service description will simulate human behavior as defined by human semantics rather than the computing semantics focusing on the hierarchy and relationship of object classes and functional operations. In this way, semantic Web service can be well defined.

First, a service domain ontology can be used to define service semantics that is domain/sub-domain independent and can be distinguished by functional category/subcategory. A temporary service domain ontology is described in Figure 3:

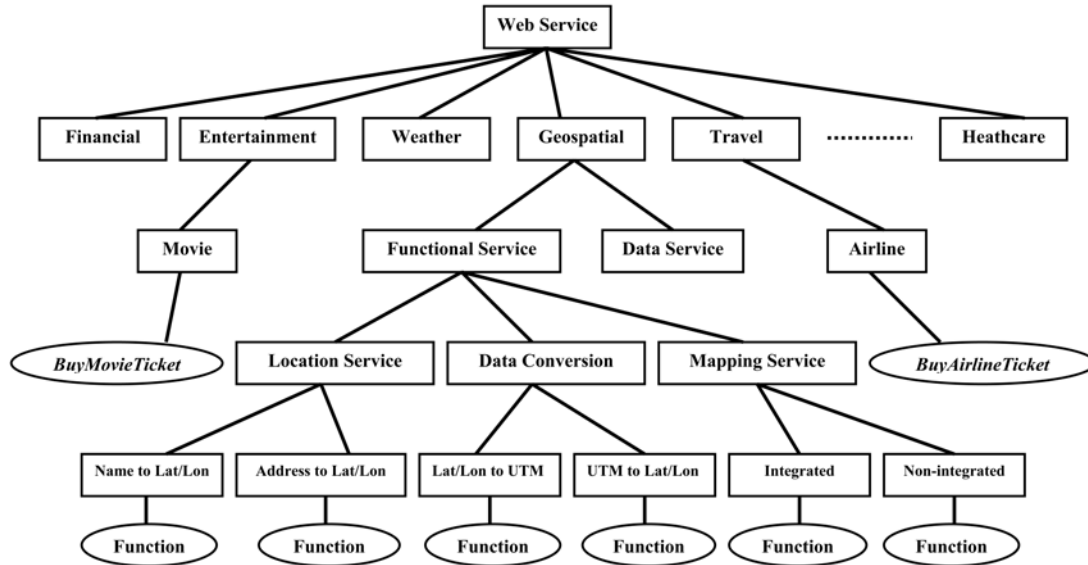


Figure 3. Hierarchy of service description

Given the example of the service template for the semantic request and response for ESRI’s Address Finder Web Service, which can be found at: <http://www-106.ibm.com/developerworks/webservices/library/ws-semantic/code14.html> , ontology-based service domain and functional category can be added as the follows:

```

<?xml version="1.0">
<WebService>
  <Service>
    <Domain>geospatial</Domain>
    <Category>functional service</Category>
    <Provider>ESRI</Provider>
    <Name>AddressFinder</Name>
    <Description>This service provides function to find the location (longitude and latitude) of a given address. </Description>
    <Function>
      <Name>findAddress</Name>
      <Category>Location service</Category>
      <Description>This function will find the location (longitude and latitude) of a given address.</Description>
      <Namespace> AddressFinder.v2.services.arcweb.esri.com</Namespace>
    </Function>
  </Service>
  .....
</WebService>
    
```

In this service description file, it is easy for requesters to understand that this service provides geospatial functional service that will find the location (longitude and latitude) of a given address. By adding service domain information into a service description file in which all elements are defined explicitly, it is easy for requesters to understand the nature of

the provided service, even if the service has certain functions that have the same name as those of the other services, such as *getTicket*.

Adding formal ontology specification onto data description will set up the correlation of different elements within different ontology domains. Figure 4 describes a Web service for geospatial data processing that creates a new buffer feature of an input polygon feature layer at a specified distance. In this example, semantic Web service is well defined since every element can be rooted to a specific ontology. Thus “Buffer” is a functional name of a Web service in the geospatial domain. “Polygon” is not a simple geometry but rather, a georeferenced polygon with a spatial reference system as NAD83 / UTM zone 17N. “Distance” is a double in the formal ontology of number while “Unit” refers to a kilometer length measurement. In contrast, the syntactic description of such a service can only tell the service requester the name of the service as well as the name of the function, input and output, as showed in such function *bufferPolygonFeature*.

By using OWL/RDF, the following Web service function:

Function *bufferPolygons* (String: *inputPolygon*, Double: *distance*, String: *Unit*)
String: *bufferPolygon*

can be described as below:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  .....>

<owl:Class rdf:ID="Geometry"/>
<owl:Class rdf:ID="DataType"/>
<owl:Class rdf:ID="SpatialReferenceSystem"/>
<owl:Class rdf:ID="Measurement"/>

.....

<owl:Class rdf:ID="2Dshape">
<rdfs:subClassOf rdf:resource="#Geometry"/>
</owl:Class>

<owl:Class rdf:ID="Polygon">
<rdfs:subClassOf rdf:resource="#2Dshape"/>
</owl:Class>

<owl:Class rdf:ID="Length">
<rdfs:subClassOf rdf:resource="#Measurement"/>
</owl:Class>

<owl:Class rdf:ID="Projected">
```

```
<rdfs:subClassOf rdf:resource="#SpatialReferenceSystem"/>
</owl:Class>

.....

<owl:DatatypeProperty rdf:ID="Distance">
<rdfs:domain rdf:resource="#DataType" />
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#double" />
</owl:DatatypeProperty>

<owl:ObjectProperty rdf:ID="Unit">
<rdfs:domain rdf:resource="#Measurement" />
<rdfs:range rdf:resource="#Length"/>
<owl:hasValue>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
<rdf:value="Kilometer"/>
</owl:hasValue>
</owl:ObjectProperty>

<owl:Class rdf:ID="inputPolygon">
<rdfs:subClassOf rdf:resource="#Polygon" />
<owl:ObjectProperty rdf:ID="SRS">
<rdfs:domain rdf:resource="#SpatialReferenceSystem" />
<rdfs:range rdf:resource="#Projected"/>
<owl:hasValue>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
<rdf:value="EPSG:26917"/>
</owl:hasValue>
</owl:ObjectProperty>
</owl:Class>

<owl:Class rdf:ID="bufferPolygon">
<rdfs:subClassOf rdf:resource="#Polygon" />
<owl:ObjectProperty rdf:ID="SRS">
<rdfs:domain rdf:resource="#SpatialReferenceSystem" />
<rdfs:range rdf:resource="#Projected"/>
<owl:hasValue>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
<rdf:value="EPSG:26917"/>
</owl:hasValue>
</owl:ObjectProperty>
</owl:ObjectProperty>

.....

</rdf:RDF>
```

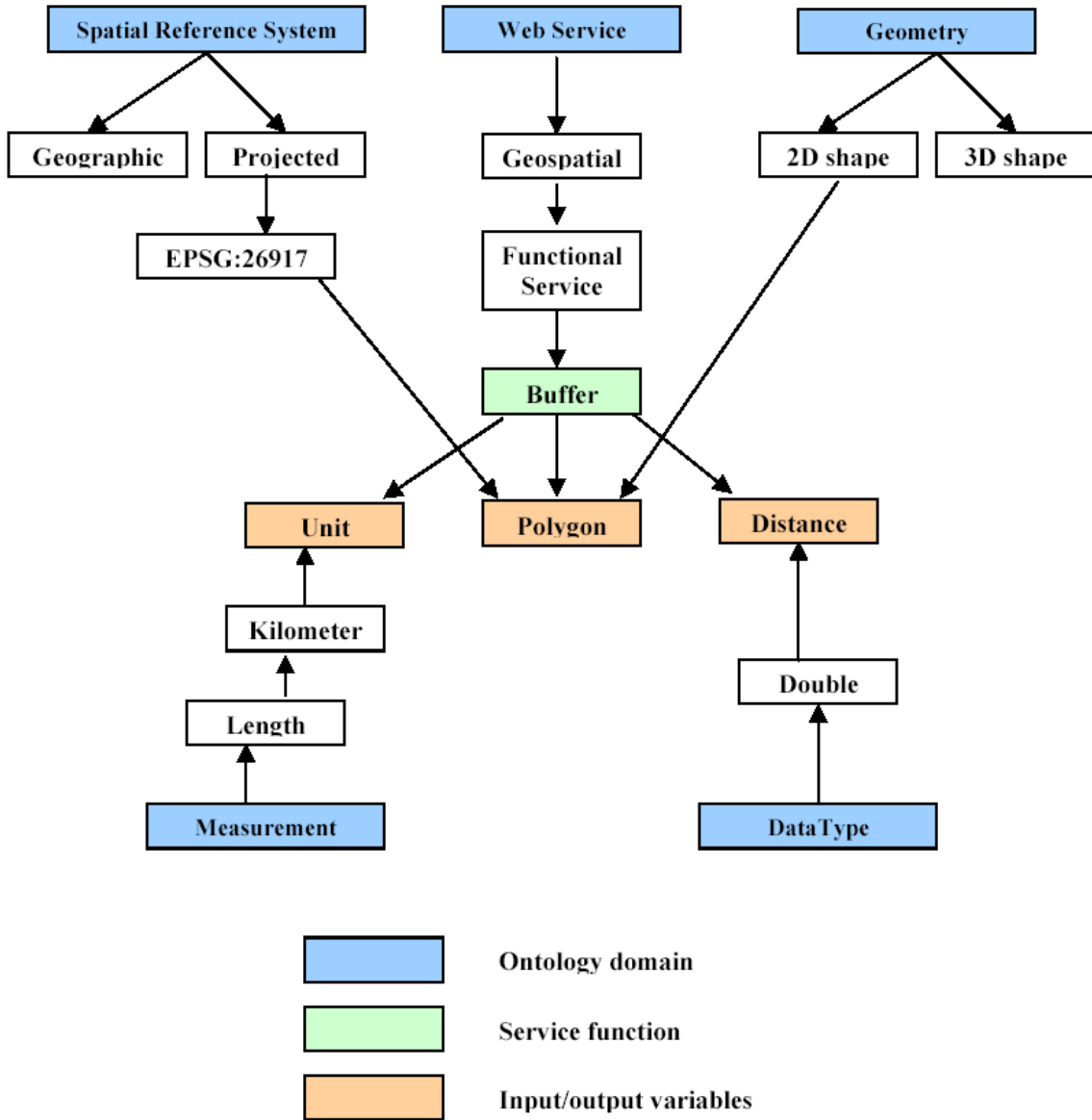


Figure 4. Adding ontology specifications onto data and service semantics

Future research and challenges

WSDL provides a standard interface for the exchange of objects and functions in distributed computing systems. As a description language, WSDL factually describes how the service architecture (the hierarchy and relation of data type, class, object, operation, etc.) is created but does not have the capability to define the content and meaning of the message exchanged in the communication. When developing Web service, semantics are imposed into programming languages to give names to different objects, classes, data types, operations, etc. Such a naming system is based on the common sense. For example, we can define a function name as *GetAirlineTicket* or *BuyAirlineTicket* or any other names to perform the same function. Under such situation, semantic Web service description tightly coupled with WSDL cannot be dependable for

service requesters to find what they really want and need. This paper has demonstrated that WSDL is *not* an appropriate source to derive service semantics by those syntactic problems on the one hand, while on the other, WSDL description may be full of redundant and irrelevant information that is useless for requesters in understanding the service semantics.

The semantic request and response approach (Shi, 2004) will help to remove syntactic barriers by transforming the composite Web services into atomic ones with explicit and meaningful description about the requirements of the service request and the expected outcome of the service response. Adding the ontology domain onto the data and service semantics described in semantic request and response will provide a way to correlate different domains into one service description that enable the service provider to describe the service semantics more accurately in a standardized form.

Future research and development will face the challenge of how to define data and service semantics for geospatial data processing and spatial analysis. More and more standards are expected to be formulated as the way to build semantic and intelligent Web services in general, and for geospatial applications in particular.

Reference:

- Cabral, L., Domingue, J., Motta, E., Payne, T., and F. Hakimpour. 2004. Approaches to Semantic Web Services: An Overview and Comparisons. <http://eprints.aktors.org/326/> and <http://eprints.aktors.org/326/01/cabralESWS04.pdf>
- Deem, M. 2002. Microsoft SOAP Toolkit Type Mappers. <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnservice/html/service06042002.asp>
- Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., Parsia, B., Payne, T., Sabou, M., Solanki, M., Srinivasan, N., and K. Sycara. 2004. Bringing Semantics to Web Services: The OWL-S Approach. In: Proceedings of First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004), July 6-9, 2004, San Diego, California, USA.
- Payne, T. and O. Lassila. 2004. Semantic Web Services. IEEE Intelligent Systems. July/August 2004
- Roman, D., Keller, U. and H. Lausen (eds.). 2004. Web Service Modeling Ontology - Standard (WSMO - Standard), version 0.1 <http://www.wsmo.org/2004/d2/v01/>
- Shi, X. 2004. Semantic request and response for standardized Web services. <http://www-106.ibm.com/developerworks/webservices/library/ws-semantic/>
- Snell, J and T. Glover. 2003. Portability and interoperability <http://www-106.ibm.com/developerworks/webservices/library/ws-port/>
- OWL-S (2004) Semantic Markup for Web Services <http://www.daml.org/services/owl-s/1.1/overview/>
- W3C. 2004a. Web Services Architecture. <http://www.w3.org/TR/ws-arch/>
- W3C. 2004b. OWL Web Ontology Language Overview. <http://www.w3.org/TR/2004/REC-owl-features-20040210/>

Acknowledgement

The author gratefully acknowledges Dr. Greg Elmes' kind direction, review and advice to this paper.