15th ICCRTS

"The Evolution of C2"

Semantic Web Services and Semantic Service-Oriented Architectures

Topic 9 – C2 Architectures and Technologies, Topic 3 – Information Sharing

Albert Frantz & Randall McIntyre

POC: Albert Frantz

Air Force Research Laboratory/Information Directorate

AFRL/RISE

525 Brooks Road

Rome, NY 13441-4505

(315) 330-1456

albert.frantz@rl.af.mil

randall.mcintyre@rl.af.mil

Semantic Web Services and Semantic Service-Oriented Architectures[1]

Albert Frantz & Randall McIntyre
Air Force Research Laboratory/Information Directorate
AFRL/RISE
525 Brooks Road
Rome, NY 13441-4505
(315) 330-1456
albert.frantz@rl.af.mil, randall.mcintyre@rl.af.mil

Abstract

The DoD is moving toward a Web services-based net-centric service-oriented architecture. A critical part of this effort will be the ability to discover, invoke and compose Web services in response to warfighter needs. Currently, discovery involves manual keyword search of the Universal Description, Discovery and Integration registry to find the appropriate Web service and its Web Services Description Language (WSDL) document. The WSDL specifies the syntactic description of how to invoke the Web service, but semantics (meaning) are needed to describe its function. The semantic Web is an evolution of the Web in which semantics of information and services are defined, allowing machines to better understand and satisfy user requests. An ontology can be defined as a set of classes, attributes (properties), and relationships among class members - with which to model a knowledge domain. Communities of interest are developing domain-specific ontologies. Semantic Web services can use these representations to describe the operations, inputs and outputs of Web services promising to pave the way for computers to discover, invoke and compose Web services with significantly less manual effort and programming. This report highlights the current approaches for developing semantic Web services.

Keywords: semantic Web services, semantic service oriented architecture, Semantic Annotations for WSDL and XML Schema (SAWSDL), Semantic Markup for Web Services (OWL-S),

[1] Distribution A: Approved for Public Release; distribution unlimited. (WBAFB PA # 88ABW-2010-1737)

Semantic Web Services Framework (SWSF), Web Service Modeling Ontology (WSMO)

Introduction

The *Joint Vision 2020*[2] (JV2020) is a guide to the continuing transformation of America's Armed Forces for the 21st century. JV2020 states, "The overarching focus of this vision is full spectrum dominance-achieved through the interdependent application of dominant maneuver, precision engagement, focused logistics, and full dimensional protection. The transformation of the joint force to reach full spectrum dominance rests upon *information superiority* as a key enabler." *Joint Publication 1-02, The Department of Defense Dictionary of Military and Associated Terms*[3] defines information superiority as — **"**The operational advantage derived from the ability to collect, process, and disseminate an uninterrupted flow of information while exploiting or denying an adversary's ability to do the same." "Service oriented computing is rapidly becoming the dominant computing paradigm."[4] According to a 2008 Gartner Inc. survey, "53 percent of large organizations were already using service oriented architectures (SOA) in some part of their organizations and another 25 percent were not using it, but had plans to do so in the next 12 months."[5] The military is moving toward a net-centric Service Oriented Architecture (SOA) based on Web services to achieve information superiority.

In 2007, the Department of Defense (DoD) released the *Net-Centric Services Strategy* (NCSS)[6] that defined the vision and strategy for a Net-Centric, Service Oriented DoD Enterprise. The NCSS document describes the DoD's vision for establishing a "Net-Centric Environment (NCE) that increasingly leverages shared services and SOA." The DoD Net-Centric Services Strategy reflects their recognition that a service oriented approach can result in an explosion of capabilities for our warfighters and decision makers, thereby increasing operational effectiveness. Furthermore, it can accelerate the DoD's ongoing effort to achieve net-centric operations by ensuring that our warfighters receive the right information, from trusted and accurate sources, when and where it is needed. The NCSS defines SOA as "a paradigm for defining, organizing, and utilizing distributed capabilities in the form of loosely coupled software

---

[2] http://www.dtic.mil/doctrine/jel/jfq_pubs/1225.pdf
[3] http://www.dtic.mil/doctrine/jel/new_pubs/jp1_02.pdf
[4] http://www.sti-innsbruck.at/fileadmin/documents/DERI-TR-2005-12-26.pdf
[5] http://www.gartner.com/it/page.jsp?id=790717
[6] http://www.defenselink.mil/cio-nii/docs/Services_Strategy.pdf

services that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with, and use capabilities to produce desired effects that are consistent with measurable preconditions and expectations."

A critical part of this effort will be the ability to discover, invoke and compose Web services to make information technology more responsive to the warfighters needs. "The semantic Web is an evolving development of the World Wide Web in which the meaning (semantics) of information and services on the web is defined, making it possible for the web to understand and satisfy the requests of people and machines to use the web content."[7] Semantics will make discovering, invoking and composing Web services easier. The current practice of finding, invoking and composing Web services primarily involves manual keyword search of the Universal Description, Discovery and Integration (UDDI) registry and Web Services Description Language (WSDL) document. The UDDI registry enables businesses to publish business information and service listings to discover each other.

"The W3C (World Wide Web Consortium) recommendation of the Web Services Description Language (WSDL) specifies a standard way to describe the interfaces of a Web service at a syntactic level and how to invoke it. While the syntactic descriptions provide information about the structure of the input and output messages of an interface and about how to invoke the service, semantics are needed to describe what a Web service actually does. These semantics, when expressed in formal languages, disambiguate the description of Web services interfaces, paving the way for automatic discovery, composition and integration of software components. The WSDL does not explicitly provide mechanisms to specify the semantics of a Web service."[8] As semantic Web technologies mature, they hold the promise of making software machine-interoperable. Communities of interest (COIs) are developing formal ontologies describing the semantics (meaning) of the COI domains. Semantic Web services add semantic descriptions of the operations, inputs and outputs of web services, which hold the promise of allowing computers to automatically find, invoke and compose web services with minimal manual effort and programming.

[7] http://en.wikipedia.org/wiki/Semantic_web
[8] http://www.w3.org/TR/sawsdl-guide/

Understanding Web Services

"Web services provide interoperability solutions, making application integration and transacting business easier. … Web services are software applications that can be discovered, described, and accessed based on XML (eXtensible Markup Language) and standard Web protocols over intranets, extranets and the internet."[9] A Web Service is defined by the W3C as "a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages (Simple Object Access Protocol), typically conveyed using HTTP (Hypertext Transfer Protocol) with an XML serialization in conjunction with other Web-related standards."[10]

Web services have distinct advantages, including:

1. Promoting interoperability because they are based on open standards, such as HTTP and XML-based protocols, including the WSDL, SOAP messages and UDDI registry, thereby making Web services platform (machine), programming language and operating system independent;
2. Communicating with SOAP messages over HTTP, usually using port 80, which means they can communicate through many common firewall measures and can easily be distributed on the internet or an intranet;
3. Exchanging text messages based on XML making it easy for developers to understand;
4. Promoting reusability for different applications - even when the applications are between different organizations or companies;
5. Providing the ability to be combined to create services that are more complex.

Web services also promote "loose coupling" between it and the applications that use Web services. Coupling refers to the direct knowledge that the Web service requestor has about the Web service provider. Loose coupling for Web services can be described as the Web service

---

[9] Michael C. Daconta, Leo J. Orbst and Kevin T. Smith "The Semantic Web", 2003, Wiley Technology Publishing ISBN 0-471-43257-1

[10] http://www.w3.org/TR/ws-gloss/

requestor and provider knowing only about the interface between the services and not the underlying programming implementation. If the interface between the provider and requestor remains the same in the WSDL document, then the underlying client and requestor software implementation can change without breaking the Web service operation.

Web services can be used with legacy information systems. If a legacy system provides the functionality required for a business service, a Web service interface can be written to utilize the legacy software, as opposed to writing new software to perform the same function. The ability to use legacy systems is extremely significant due to the investment organizations have in legacy software and the cost to replace them - particularly in the DoD.

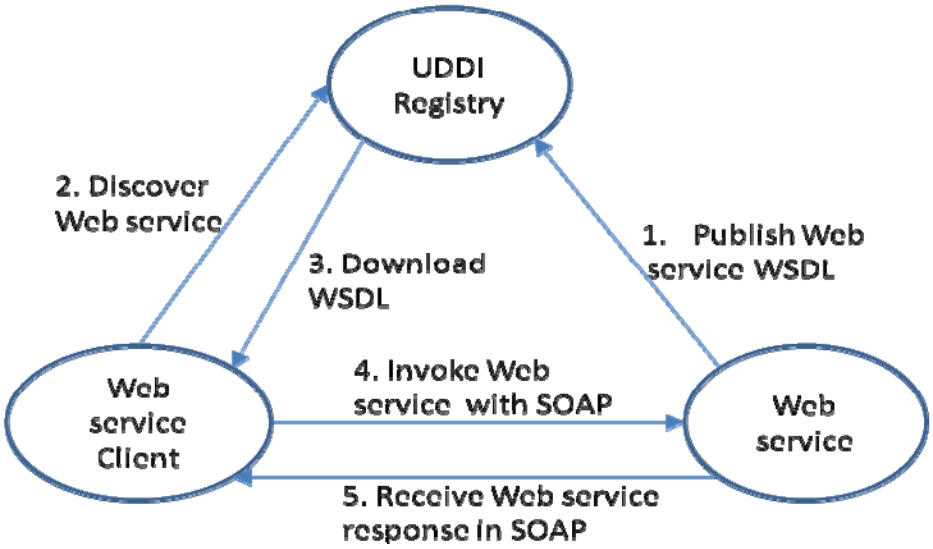

Figure 1 A common scenario for using a Web service without semantics

Figure 1 shows a typical scenario for a Web service, which can be described as follows:

1. The Web service is written and a WSDL document describing the Web service, in both concrete and abstract terms, is generated (usually automatically) and published to the UDDI registry.
2. A Web service client looks for and discovers the appropriate Web service in the UDDI registry. It is very important to note that discovery is typically a manual keyword search process requiring human oversight.
3. The client downloads the WSDL describing the Web service.

4. The client then creates the client code to invoke the Web service with a SOAP message.
5. The Web service sends the response to the client in a SOAP message.

The WSDL is an XML-based language that provides a model for describing the syntax of Web services. WSDL 2.0 is a W3C recommendation as of 26 June 2007. "Web Services Description Language Version 2.0 (WSDL 2.0) provides a model and an XML format for describing Web services. WSDL 2.0 enables one to separate the description of the abstract functionality offered by a service from concrete details of a service description such as "how" and "where" that functionality is offered. This specification defines a language for describing the abstract functionality of a service as well as a framework for describing the concrete details of a service description. It also defines the conformance criteria for documents in this language."[11] WSDL 1.1 is still used extensively, but is not a W3C recommendation and the differences will not be covered in this paper. Figure 2, modified from the WSDL website on Wikipedia[12], shows the representation of the concepts of a WSDL 2.0 document.

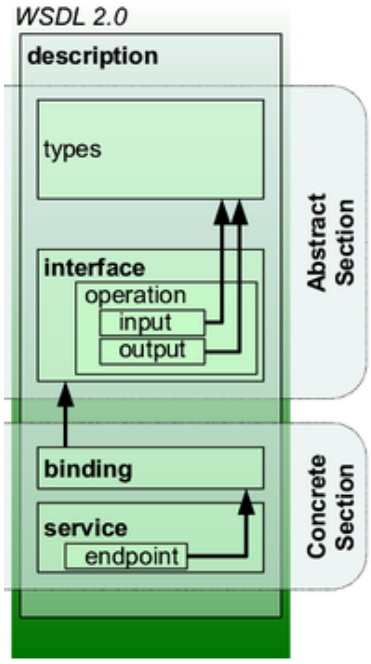

Figure 2 Representation of concepts defined by a WSDL 2.0 document

[11] http://www.w3.org/TR/wsdl20/
[12] http://en.wikipedia.org/wiki/Web_Services_Description_Language

In Figure 2 the *types* element describes the kinds of messages that the service will send and receive in XML Schema. The *interface* element describes what abstract functionality the Web service provides, including the operations and types of messages the service can send and receive as part of that operation. The *binding* element describes how to access the service, including the message format and transmission protocol details. The *service* element describes where to access the service.[13]

The UDDI registry is the equivalent of a phone book for Web services. The UDDI "enables businesses to publish service listings and discover each other and define how the Web services interact over the Internet."[14] The UDDI is an open industry initiative sponsored by the Organization for the Advancement of Structured Information Standards (OASIS). The UDDI is written in XML and contains three basic components:

1. White Pages – Business addresses, points of contact and identifiers;
2. Yellow Pages – Business industrial categorizations based on standard taxonomies;
3. Green Pages – Technical information about the Web services exposed by the Business including the WSDLs.

The original vision for the UDDI registry was a public place where Web service providers could publish information about their business and the Web services they provided. Anyone needing a Web service could browse the UDDI to find the appropriate Web service, the descriptions of how to use and where to invoke it, along with the Web service policies such as cost of use. Unfortunately, the large public UDDIs hosted by IBM, Microsoft, and SAP were shut down in January 2006, making it even harder to find Web services.[15] Domain or organization specific UDDIs of a smaller scale are still in use today.

Adding Semantics to Web Services

"The Semantic Web is an evolving development of the World Wide Web in which the meaning (semantics) of information and services on the web is defined, making it possible for the web to understand and satisfy the requests of people and machines to use the web content."[16] One of the key ways to define the semantics of information on the web is the ontology. "In

[13] http://www.w3.org/TR/2007/REC-wsdl20-primer-20070626/
[14] http://en.wikipedia.org/wiki/UDDI
[15] http://uddi.microsoft.com/about/FAQshutdown.htm
[16] http://en.wikipedia.org/wiki/Semantic_web

computer science and information science, an ontology is a formal representation of a set of concepts within a domain and the relationships between those concepts. … An ontology provides a shared vocabulary, which can be used to model a domain – that is, the types of objects and/or concepts that exist, and their properties and relationships. … The creation of domain ontologies is also fundamental to the definition and use of an enterprise architecture framework."[17] The W3C has published several XML-based recommendations for writing ontologies on the Web, namely the Web Ontology Language (OWL)[18], Resource Description Framework (RDF)[19] and RDF Schema (RDFS).[20] The ontologies in XML are machine-interpretable, as well as human-readable, although they are much easier for humans to read in ontology editing tools. Figure 3 shows a simple ontology in a graphical form. In OWL, all concepts (classes) are inherited from the *Thing* class. In this ontology, the *Thing* class has three subclasses; *Aircraft*, *Weapons* and *Airbase* and the *Aircraft* class has two subclasses; *Transport_Aircraft* and *Combat_Aircraft*.

There are three relationships between the classes:

1. An instance of *Aircraft departs_From* an *Airbase;*
2. Its inverse relationship - An *Airbase has_Departing Aircraft* and;
3. A *Combat_Aircraft has_Weapon* from the *Weapon* class.

[17] http://en.wikipedia.org/wiki/Ontology_(information_science)
[18] http://www.w3.org/2004/OWL/
[19] http://www.w3.org/RDF/
[20] http://www.w3.org/TR/rdf-schema/

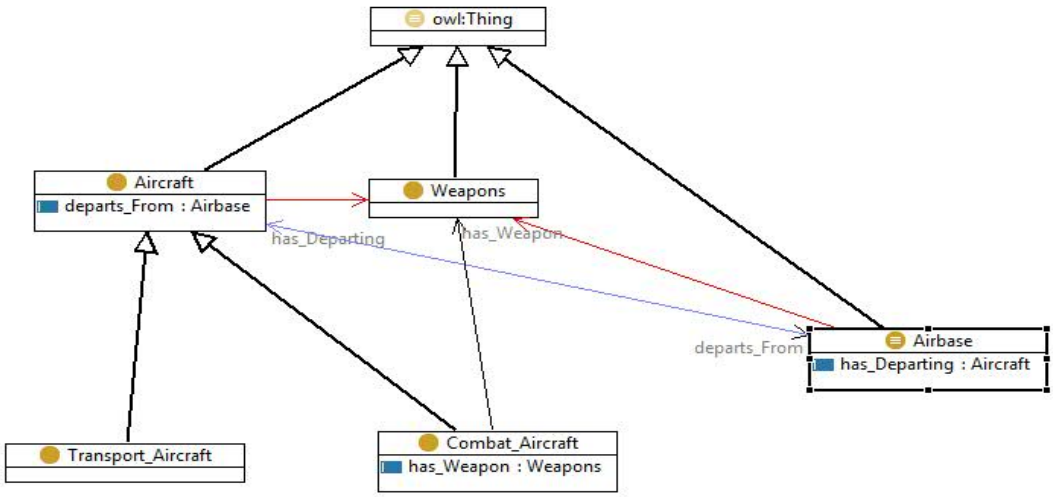

Figure 3 A Simple Ontology

DoD Communities of Interest (COIs) are writing ontologies to establish standard vocabularies for information sharing within the domain of their community. These ontologies provide an excellent source of semantics for Web services.

"As the set of available Web Services expands, it becomes increasingly important to have automated tools to help identify services that match a requester's requirements. Finding suitable Web services automatically depends on the facilities available for service providers to describe the capabilities of their services and for service requesters to describe their requirements in an unambiguous and, ideally, machine-interpretable form. Adding semantics to represent the requirements and capabilities of Web services is essential for achieving this clarity and machine-interpretability."[21]

Several approaches for semantic Web services have been proposed, but currently, only Semantic Annotations for WSDL and XML Schema (SAWSDL) has become a W3C recommendation.[22] SAWSDL enables the addition of semantic notations for Web services using and building on the WSDL and UDDI. The semantic annotations can be used for classifying, discovering, matching, monitoring, composing and invoking Web services. SAWSDL is based

[21] http://www.w3.org/TR/sawsdl-guide/
[22] http://www.w3.org/2002/ws/sawsdl/

on the W3C member submission WSDL-S[23] from IBM and the University of Georgia. It was agreed upon as the first step and evolutionary approach to semantic Web services at a W3C Workshop, which ultimately led to the SAWSDL Working Group.

To discover a Web service or its operations, SAWSDL provides a mechanism to semantically annotate the service and its operations with categorization information that can be published in a registry. This SAWSDL mechanism is called *modelReference* and points to concepts in an ontology. SAWSDL does not require the ontology to be written in semantic Web languages like RDF or OWL, but these will likely be the most popular choices since they are based on XML - the same as SAWSDL, WSDL and UDDI registry.

Listing 1 is an example WSDL from the "Semantic Annotations for WSDL and XML Schema - Usage Guide"[24] (SAWDL User Guide). Like all WSDLs, it is written in XML and not particularly easy to read. The interface section of the WSDL defines a service named the *CheckAvailabilityRequestService* (in red font - added for readability) that has one operation (also in red) called the *CheckAvailabilityRequestOperation*. The operation has an input request and an output response (in red.) The types section of the WSDL defines the input requirements for the request *CheckAvailabilityRequestServiceRequest.* This request has three elements (in blue) called the *itemCode*, *date* and *qty*, along with one boolean response for the *CheckAvailabilityRequestServiceResponse* - an *itemConfirmation* (in blue) whose value depends upon if the item (*itemCode*) and quantity (*qty*) are available on the required *date*. This WSDL example does not include the concrete section that contains the binding and service endpoint.

```
<wsdl:description
  targetNamespace="http://org1.example.com/wsdl/CheckAvailabilityRequestService/"
  xmlns="http://org1.example.com/wsdl/CheckAvailabilityRequestService/"
  xmlns:wsdl="http://www.w3.org/ns/wsdl"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <wsdl:types>
   <xsd:schema targetNamespace="http://org1.example.com/wsdl/CheckAvailabilityRequestService">
    <xsd:element name="CheckAvailabilityRequestServiceRequest">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="itemCode" type="xsd:string"/>
          <xsd:element name="date" type="xsd:string"/>
          <xsd:element name="qty" type="xsd:float"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
```

[23] http://www.w3.org/Submission/WSDL-S/
[24] http://www.w3.org/TR/sawsdl-guide/

```
    <xsd:element name="CheckAvailabilityRequestServiceResponse" type="itemConfirmation"/>
    <xsd:simpleType name="itemConfirmation">
      <xsd:restriction base="xsd:boolean"/>
    </xsd:simpleType>
    </xsd:schema>
  </wsdl:types>

  <wsdl:interface name="CheckAvailabilityRequestService">
   <wsdl:operation name="CheckAvailabilityRequestOperation"
pattern="http://www.w3.org/ns/wsdl/in-out">
     <wsdl:input element="CheckAvailabilityRequestServiceRequest"/>
     <wsdl:output element="CheckAvailabilityRequestServiceResponse"/>
   </wsdl:operation>
 </wsdl:interface>
</wsdl:description>
```

Listing 1 A WSDL sample for a CheckAvailabilityRequestService

In Listing 2 a SAWSDL *modelReference* annotation is included for the Web service pointing to an ontology shown (in bold.) This statement equates the *CheckItemAvailabiltyRequestService* Web service to the *ItemAvailabilityCheck* concept in the *POServiceClassification* (PO meaning Purchase Order) ontology.

```
...
<wsdl:interface name="CheckItemAvailabilityRequestService"

  sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/spec/examples/taxonomy/POServiceClassif
ication#ItemAvailabilityCheck">
    ...
</wsdl:interface>
    ...
```

Listing 2 A SAWSDL annotation added to the CheckAvailabilityRequestService

Figure 4 demonstrates the basic concept of this SAWSDL *modelReference* semantic categorization annotation. It shows the *modelReference* SAWSDL annotation in the WSDL interface of the *CheckItemAvailabilityRequestService* Web service pointing to the ontology class *ItemAvailabilityCheck.* Since the ontology is machine-interpretable, a computer search engine can automatically find the Web service, even though the service's name is different from the ontology class name being searched for because the SAWSDL annotation defines them as equivalent.

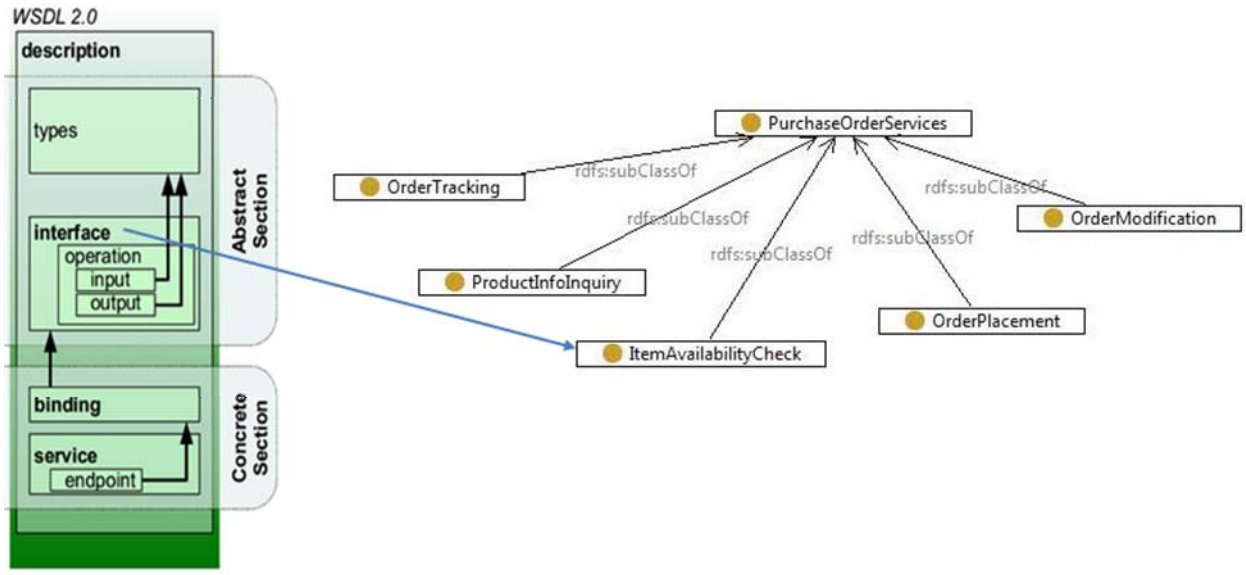

Figure 4 Semantic Annotations to Publish a Web Service

Categorization annotations using *modelReference* can also be used in the WSDL and UDDI to point to industry standard classifications taxonomies, such as North American Industry Classification System (NAICS)[25] - the SAWSDL User Guide gives an example of this.

The SAWSDL *modelReference* mechanism can also be used to semantically annotate Web services so that these semantics can be used to match and compose Web services. If Web services match, then they are functionally equivalent. This might occur when checking the availability of products from different vendors that have different Web services to do so. The two Web services, their operations, inputs and outputs may have different names, but if the individual parts have *modelReferences* pointing to the same classes in an ontology, they will be functionally equivalent and match as shown in the example in the SAWSDL User Guide. The first Web service, *CheckItemAvailabilityRequestService* has an operation called *CheckAvailabilityRequestOperation* with inputs of *itemCode*, *date* and *qty* and an output *itemConfirmation*. A second Web service called *CheckInventoyService* might have an operation called *checkInventoryOperation* with inputs of *SKU*, *deliveryDate* and *numBundles* and an output of *conf*. In this case, if both Web services inputs and outputs SAWSDL *modelReferences* point to the same classes of the ontology, then they would match. The ontology classes might be called *SKU*, *DueDate*, *Quantity* and *AvailabilityConfirmation*. The Web services may point to

[25] http://www.census.gov/eos/www/naics/

different ontologies, but if the classes in the ontologies can be determined to be equivalent or similar through class equivalence relationships or reasoning they may still match. "A semantic reasoner is a piece of software able to infer logical consequences from a set of inferred facts."[26] In the example, *SKU* is determined to be a subclass of *itemCode*, making the Web services still match.

To compose Web services, some of the outputs of the first Web service must match the inputs of the second Web service. For instance, a Web service may require an input of an aircraft designation like *F-15E*, but what exists is the aircraft name *Strike_Eagle*. If a Web service is available for converting aircraft names to aircraft designations, it could be used to convert the name to the proper designation for the second Web service to use as an input. A machine agent may use SAWSDL annotations and ontologies to automatically compose multiple Web services. The SAWSDL *modelReference* may also point to semantic rules for conditions on inputs and outputs and use reasoners to verify these rules. Section 3.7 of the SAWSDL User Guide has some fairly complex examples of these, which we do not have space to cover in this paper.

SAWSDL also provides a mechanism, called *Schema mapping*, to facilitate data transformations for Web services. "A *Schema mapping* allows the specification of transformation functions on the WSDL elements to map instance data defined by that XML schema document to the semantic data of the concepts in a semantic model. It also allows the specification of transformation functions that map the semantic data of ontological concepts to the instance data values that adhere to the XML schema document that is being annotated. In the former case, the transformation functions are referred to using the extensibility attribute *liftingSchemaMapping* and in the latter case it is called *loweringSchemaMapping*. These kinds of mappings are useful in general when the structure of the XML instance data does not correspond directly to the organization of the semantic data. Also, these types of mappings can be used to generate mediation code to support invocation of a Web service."[27] SAWSDL does not require a specific mapping language, but Extensible Stylesheet Language (XSL) Transformations (XSLT) are popular because, like WSDLs and SAWSDL, they are XML compliant.

---

[26] http://en.wikipedia.org/wiki/Reasoning_engine
[27] http://www.w3.org/TR/sawsdl-guide/

A *liftingSchemaMapping* takes as input XML data (that adheres to a given XML Schema) and produces semantic data that adheres to a semantic model. The XSLT can convert a document from one type to another. The *liftingSchemaMapping* in a WSDL points to an XSLT document that transforms the XML data into a semantic model. The SAWSDL User Guide gives an example of a *lifitingSchemaMapping* that takes a purchase order request in the WSDL in XML and converts it to a semantic model in RDF (also XML compliant). The XML *OrderRequest* has *firstName*, *lastName* and *item* where *item* is a complex type including *itemCode*, *quantity*, *dueDate* and *billingInfo*. The XSLT document converts it to an RDF *OrderRequest* ontology with classes *LineItem*, *Part*, *PartNum*, *Date*, *Customer*, *Name* and properties of *hasLineItems*, *hasBillingInformation*, *hasPart*, *hasPartCode*, *hasDueDate*, *hasQuantity*, *hasCustomer*, and *hasCustomerName*. Figure 5 shows the resulting classes (orange circles) starting with the *OrderRequest* class, data properties (green rectangles) and object properties (blue rectangles) in the ontology. Object properties point from one class to another, while data properties point from an object to data.

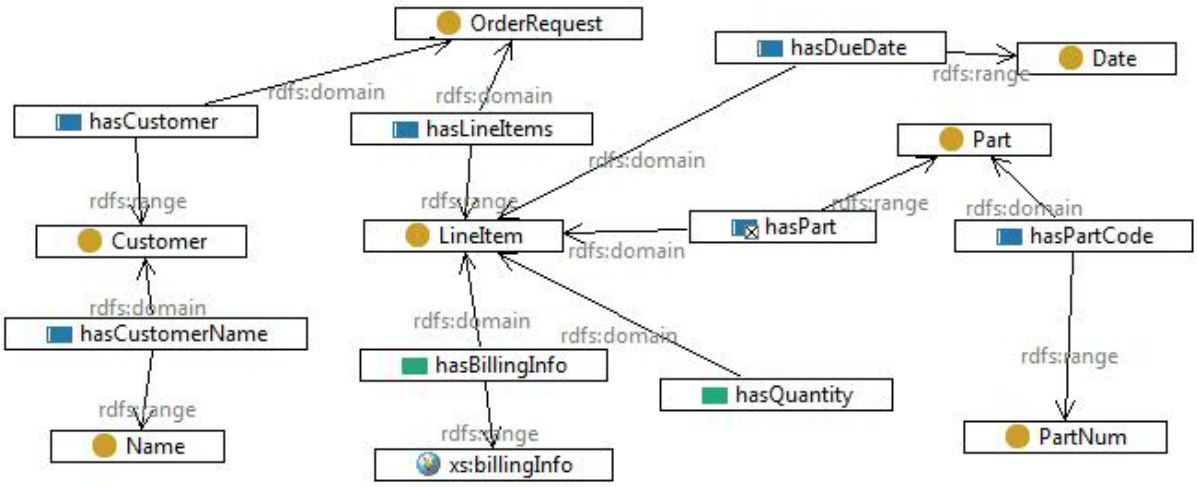

Figure 5 OrderRequest Ontology Classes and Properties

As one might expect, a *loweringSchemaMapping* performs the opposite mapping from RDF schema data to the XML data. The *loweringSchemaMapping* may use the W3C recommendation SPARQL Query Language for RDF (SPARQL)[28] on the semantic data to

[28] http://www.w3.org/TR/rdf-sparql-query/

obtain specific semantic data to transform with an XSLT. An example is given in the SAWSDL User Guide.

Tools available for SAWSDL include:

- SAWSDL4J[29] from Wright State University and the University of Georgia;
- Radiant[30] from the University of Georgia;
- Semantic Tools for Web Services[31] from IBM alphaWorks and;
- WSMO Studio[32] from the Digital Enterprise Research Institute (DERI) that includes a SAWSDL editor.

Another approach, OWL-S, the Semantic Markup for Web Services, was a member submission to the W3C in 2004.[33] OWL-S, like SAWSDL, is designed to support automatic Web service discovery, invocation, composition and interoperation. OWL-S is an ontology built of top of the Web Ontology Language (OWL) for describing semantic Web services. The OWL-S ontology has three main parts (sub-ontologies): the service profile, the process model and the service grounding. Figure 6 from the OWL-S submission illustrates these relationships.

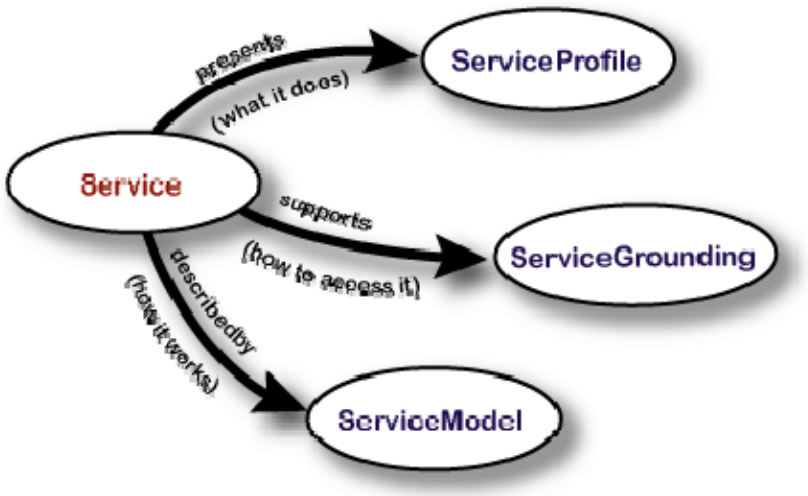

Figure 6 OWL-S Upper Ontologies

[29] http://knoesis.wright.edu/opensource/sawsdl4j/
[30] http://lsdis.cs.uga.edu/projects/meteor-s/downloads/index.php?page=1
[31] http://www.alphaworks.ibm.com/tech/wssem
[32] http://www.wsmostudio.org/
[33] http://www.w3.org/Submission/OWL-S/

The service profile describes what the service does, in a way suitable for automatic service discovery. The service profile includes the service name, provider information, the inputs and outputs of the service, the preconditions required by the service, a description of what the service accomplishes, and the expected effects that result from execution of the service. The profile may also provide service classification information, as well as information about the quality of service it provides.

The service grounding specifies the details of how to access a service. Typically, the grounding will specify a communication protocol, message formats, and other service-specific details, such as port numbers used in contacting the service. The OWL-S submission describes the complementary relationship between the service grounding and the WSDL.

The service model describes how a service works, including how to invoke, compose and monitor the service. This description includes the sets of inputs, outputs, pre-conditions and results of the service execution. The service model can be viewed as a process - a specification of the ways a client may interact with a service.

The latest release of OWL-S, version 1.2, including all the OWL-S ontologies, can be found at the DARPA Agent Markup Language (DAML) Services Site.[34] A hyperlink to an extensive list of OWL-S tools can also be found there. Overall, OWL-S has more capabilities than SAWSDL, but critics state since OWL-S is based on OWL, it is not sufficiently expressive for all aspects of a semantic Web service and requires more expressive languages like SWRL, the Semantic Web Rule Language, to be used to gain the required expressiveness.[35]

Another approach to semantic Web services is the Semantic Web Services Framework (SWSF) that was submitted to the W3C as a member submission in 2005.[36] SWSF consists of two parts - the Semantic Web Services Language (SWSL) and the Semantic Web Services Ontology (SWSO) conceptual model. SWSL is used to specify formal characterizations of Web service concepts and descriptions of individual Web services. SWSL includes two sublanguages, (1) SWSL-FOL, based on first-order logic to express the ontology of Web service concepts, and; (2) SWSL-Rules, based on logic-programming used to support the in service ontology reasoning and execution environments. The SWSO, influenced by OWL-S, presents a conceptual model

[34] http://www.daml.org/services/owl-s/
[35] http://www.sti-innsbruck.at/fileadmin/documents/DERI-TR-2005-12-26.pdf
[36] http://www.w3.org/Submission/SWSF/

by which Web services can be described. SWSL is a general-purpose language with features to support the basic languages and infrastructure of the Web, including XML. Since SWSF is based on first-order logic and rules, it is more powerful and expressive than OWL-S, but has not been as popular. SWSF is likely to have a steeper learning curve than SAWSDL and OWL-S for the average programmer.

The other member submission to the W3C for semantic Web services was the Web Service Modeling Ontology (WSMO) in 2005.[37] The WSMO is part of the overall W<Triple> technology - an approach to semantically enrich service oriented architectures; it is described in detail in the "Semantically Enabled Service-Oriented Architectures: A Manifesto and Paradigm Shift in Computer Science".[38] The W<Triple> technology combines four major building blocks:

- The Web Services Modeling Ontology (WSMO) is a conceptual model for structuring semantic annotation of services. It is Web compliant and has four main components: Goals, Ontologies, Mediators and Web Services;
- The Web Service Modeling Language (WSML) is a family of languages that provides formal semantics for WSMO models. WSML is based on Description Logics, Logical Programming and Web standards. It also includes an extension of SAWSDL;
- The Web Service Execution Environment (WSMX) is an execution environment for the dynamic discovery, selection, mediation, invocation and inter-operation of the semantically described Services. The WSMX specification has been relabeled and is being developed through OASIS as the Semantic Execution Environment.
- Triple Space is a new paradigm for the enabling of communication and cooperation of semantic Web services based on knowledge technologies.

This submission represents an entire vision to achieve semantic Web services. WSMO tools can be found at http://www.wsmo.org/. Parts of WSMO are continuing in development and have evolved into part of the European Commission Seventh Framework Programme Service Web 3.0[39], SOA4All effort[40] and the Networked European Software & Services Initiative.[41] Europe

[37] http://www.w3.org/Submission/2005/06/
[38] http://www.sti-innsbruck.at/fileadmin/documents/DERI-TR-2005-12-26.pdf

[39] http://en.wikipedia.org/wiki/Service_Web_3.0
[40] http://en.wikipedia.org/wiki/SOA4All

is currently investing significantly in semantic Web services and Semantically Enable Service-Oriented Architectures (SESA).

Conclusions

As the DoD moves toward a net-centric service oriented architecture, there will likely be tens of thousands of Web services. Manually finding the right Web services and composing them to perform the desired tasks will be extremely challenging. Semantic Web services are an evolution of current Web service technology that offers the potential to automate much of this work. Semantic Web services are still evolving, but any semantics that can be added to Web services will be a significant addition to automating Web service discovery, invocation, composition and monitoring. With the acceptance of SAWSDL as a W3C recommendation, it offers an excellent starting point for adding semantics to Web services until more complex semantic Web technologies evolve.

[41] http://www.nessi-europe.eu/