DAML-S and Related Technologies

DAML-S is a DAML+OIL ontology for describing Web Services. The language DAML+OIL [11] represents part of the Semantic Web initiative to provide semantics to the Web, i.e. to make Web content unambiguously computer-interpretable [10]. DAML-S, specifically, aims to make Web services computer-interpretable—described with sufficient information to enable *automated* Web service discovery, invocation, composition and execution monitoring.

As a DAML+OIL ontology, DAML-S retains all the benefits of Web content described in DAML+OIL. Namely, it has a well-defined semantics, it enables the definition of a Web services vocabulary in terms of objects and the complex relationships between them, including class, subclass relations, cardinality restrictions, etc. [11]. It also includes all the XML typing information.

The DAML-S ontology comprises three parts:

1. `ServiceProfile`: This is similar to a yellow page entry for a service. It describes properties of a service necessary for automatic discovery, such as the functionality the service offers, and its inputs, outputs, preconditions and effects.

2. `ServiceModel`: This describes the service's process model (the control flow and data-flow involved in using the service). It is designed to enable automated composition and execution of services.

3. `ServiceGrounding`: This connects the process model description to communication-level protocols and message descriptions in WSDL.

Due to their use of the DAML+OIL ontology-definition language, these components are annotated with classes of well-defined types that make the service descriptions machine-readable and unambiguous. Additionally, the ontological structure of classes allows class definitions to draw properties from hierarchical inheritance and through relationships to other classes.

1 Related Work

Industry efforts to develop standards for electronic commerce, and in particular for the description of Web-based services, revolve around UDDI, WSDL, BPEL4WS and ebXML. There have also been company-specific initiatives to define architectures for e-commerce, most notably e-speak from Hewlett-Packard.

In the following sections, we look in greater detail at each of these technologies in turn and compare them to DAML-S.

1.1 UDDI

UDDI (Universal Description, Discovery and Integration) is an initiative begun by Microsoft, IBM and Ariba to develop a standard for an online registry, to enable the publishing and dynamic discovery of Web services offered by businesses [1]. UDDI allows programmers and other representatives of a business to locate potential business partners and form business relationships on the basis of the services they provide. It thus facilitates the creation of new business relationships.

The primary target of UDDI seems to be integration and at least semi-automation of business transactions in B2B e-commerce applications. It provides a registry for registering businesses and the services they offer. These are described according to an XML schema defined by the UDDI specification. A Web service provider registers its advertisements along with keywords for categorisation. A Web services user retrieves advertisements out of the registry based on keyword search. The UDDI search mechanism relies on predefined categorisation through keywords and does not refer to the semantic content of the advertisements. The registry is supposed to function in a fashion similar to white pages or yellow pages, where businesses can be looked up by name or by a standard service taxonomy as is already used within the industry. UDDI attempts to cover all kinds of services offered by businesses, including those that are offered by phone or e-mail and similar means; in principle, DAML-S could do this, but it has not been our focus.

Technically speaking, each business description in UDDI consists of a businessEntity element, akin to a White Pages element describing the contact information for a business. A businessEntity describes a business by name, a key value, categorisation, services offered (businessService elements) and contact information for the business. A businessService element describes a service using a name, key value, categorisation and multiple bindingTemplate elements. This can be considered to be analogous to a Yellow Pages element that categorises a business. A bindingTemplate element in turn describes the kind of access the service requires (phone, mailto, http, ftp, fax etc.), key values and tModelInstances. tModelInstances are used to describe the protocols, interchange formats that the service comprehends, that is, the technical information required to access the service. It is also used to describe the namespaces for the classifications used in categorisation. Many of the elements are optional, including most of the ones that would be required for matchmaking or service composition purposes.

UDDI aims to facilitate the discovery of potential business partners and the discovery of services and their groundings that are offered by known business partners. This may or may not be done automatically. When this discovery occurs, programmers affiliated with the business partners program their own systems to interact with the services discovered. This is also the model generally followed by ebXML. DAML-S enables more flexible discovery by allowing searches to take place on almost any attribute of the ServiceProfile. UDDI, in contrast, allows technical searches only on tModelKeys, references to tModelInstances, which represent full specifications of a kind of service.

UDDI, of itself, does not support semantic descriptions of services. Thus, depending on the functionality offered by the content language, although agents can search the UDDI registry and retrieve service descriptions, a human needs to be involved in the loop to make sense of the descriptions, and to program the access interface.

UDDI does not provide or specify content languages for advertisement so far. Although WSDL is most closely associated with UDDI as a content language, the specification refers to ebXML and XML/edi also as potential candidates. Content languages could be a possible bridge between UDDI and DAML-S. DAML-S is also a suitable candidate for a content language and in this sense, DAML-S and UDDI are complementary. A higher-level service or standard defined on top of UDDI could take advantage of the additional richness of content DAML-S has to offer within the UDDI registries. This has been discussed further in [**?**]

1.2 WSDL

WSDL (Web Services Description Language) is an XML format, closely associated with UDDI as the language for describing interfaces to business services registered with a UDDI database. It is thus closer to DAML-S in terms of functionality than UDDI. Like DAML-S, it attempts to separate services, defined in abstract terms, from the concrete data formats and protocols used for implementation, and defines bindings between the abstract description and its specific realization [3]. However, the abstraction of services is at a lower level than in DAML-S.

Services are defined as endpoints, which are essentially sets of ports, that is, network addresses associated with certain protocols and data format specifications. The abstract nature of a service arises from the abstract nature of the messages and operations mapped to a port and define its port type. Port types are reusable and can be bound to multiple ports [4]. Operations are of four basic kinds in WSDL: a one-way, a (two-way) request-response, a (two-way) solicit-response and a (one-way) notification message. A message itself is defined abstractly as a request, a response or even a parameter of a request or response and its type, as defined in a type system like XSD. They can be broken into parts to define the logical break-down of a message.

Like UDDI, WSDL does not support semantic description of services. WSDL focuses on the grounding of services and although it has a concept of input and output types as defined by XSD, it does not support the definition of logical constraints between its input and output parameters. Thus its support for discovery and invocation of services is less versatile than that of DAML-S.

The ability of WSDL to specify the translation from abstract messages that are expressed in terms of the information to be transmitted, and concrete messages that specify the format of the message exchanged has been exploited by the DAML-S specification of the grounding. Within DAML-S the abstract message is specified by the information used in the input and output of processes in the Process Model, this information is mapped through DAML-S Grounding to WSDL abstrct messages which can then tranformed into concrete messages used in the interaction between web services.

1.3 BPEL4WS

BPEL4WS is essentially a process modeling language. It relates most closely to the ServiceModel or Process Model component of DAML-S. It has been designed to enable a would-be service composer to aggregate one or more Web services into a (possibly non-deterministic) execution of one or more Web services.

BPEL4WS distinguishes between abstract and executable processes. Abstract process may cloak internal behavior (e.g. decision processes) as non-deterministic junctions, while executable processes model the actual behavior of the process. Abstract processes are useful for describing business protocols, while executable processes may be compiled into invokable services.

Aggregated services are modeled as directed graphs where the nodes are services and the edges represent a dependency link from one service to another. The runtime semantics of the links may be specified in the BPEL4WS document. For example, the user may simulate Petri-Net behavior by stipulating that a service may execute only after all its parents execute successfully. Canonical programmatic constructs like SWITCH, WHILE and PICK allow properties of inter-service messages to direct an execution's path through the graph.

For descriptions of what services do and how they work, BPEL4WS references port types contained in WSDL documents. Transitively, then, the expressiveness of service behavior and inputs/outputs is constrained by XML and XML schema. A BPEL4WS document uses these descriptions to define "roles" within a composition that are filled by "partners". A service that meets the restrictions set by a partner definition may fill that role in a composition. The port-specific information about a partner may be set at run time, allowing partner roles to be filled dynamically.

BPEL4WS was released along with two others specs: WS-Coordination and WS-Transaction. WS-Coordination [15] describes how services can make use of pre-defined coordination contexts to subscribe to a particular role in a collaborative activity. WS-Transaction [16] provides a framework for incorporating transactional semantics into coordinated activities. In essence, WS-Transaction uses WS-Coordination to extend BPEL4WS to provide a context for transactional agreements between services. Different agreements may be described in an attempt to achieve consistent, desirable behavior while respecting service autonomy.

Clearly DAML-S and BPEL4WS have broad and somewhat complementary objectives. The DAML-S ServiceProfile complements and extends ideas in UDDI. The DAML-S ServiceGrounding connects the application level content description of a service to communication level descriptions in WSDL. It is the ServiceModel (aka ProcessModel) in DAML-S that relates most closely to the business process model in BPEL4WS.

Both provide a mechanism for describing a business process model. With so many candidate formalisms for describing a business process (e.g., XLANG, WSFL, BPMI, BPML, now BPEL4WS, etc.) DAML-S was designed to be agnostic with respect to a process model formalism. Rather, it aimed to provide the vocabulary and agreed upon (necessary) properties for a process model. In so doing, we hoped to remain compatible with what we anticipated would eventually be an agreed-upon standard for process modeling. If such a standard did not come to pass, DAML-S would provide a way of talking about different process models, in keeping with the approach and spirit of NIST's PSL [12]. Here are some of the features that distinguish/differentiate DAML-S from BPEL4WS.

Expressiveness:

- preconditions and effects: DAML-S is augmented with preconditions and effects. This enables encoding of side-effects of services. This is important for Web service composition because it enables higher-level reasoning about how services may be aggregated to achieve a particular goal while effecting particular changes on the world.
- hierarchies, taxonomy information: DAML-S classes may draw properties from inheritance and other relationships to other DAML-S classes, thus providing for a richer representation of an individual service and the relationships between services.
- rich "typing" of Web concepts: DAML+OIL enables the definition of classes in terms of their property ranges, and their relationships to other classes. E.g., we can define a class called US-FAA-flight codes as a subclass of FAA-flight codes where the location of the airport designated by the code is restricted to the USA. In so doing, we can type content in terms of these classes and reason and search over them. DAML-S also includes the full suite of XML data types.
- BPEL4WS as well as DAML-S use WSDL port type information for service descriptions. WSDL does not describe side-effects or preconditions of services, and the expressiveness of service behavior and inputs/outputs is restricted to the interaction specification.

Semantics:

- The intended interpretation of the DAML-S process model can be defined in three ways:
  1. By a translation to (axiomatization in) first-order logic
  2. By a translation to an operational semantics using Petri Nets [13], and

3. By a comparable translation to subtype polymorphism [14], where 2. and 3. are very similar. (See the discussion in [14].) Note that the semantics is defined by a translation because the semantics of DAML+OIL (the language in which the DAML-S ontology is described) is not sufficiently expressive to capture the intended interpretation of a rich process model.

- Although BPEL4WS represents the merging of XLANG and WSFL–rooted in Pi-calculus and Petri Nets, respectively–there is currently no evidence that BPEL4WS is based on a formal semantics.

Automated discovery, composition, and execution:

- The DAML-S ServiceProfile and ServiceModel provide sufficient information to enable automated discovery, composition, and execution based on well-defined descriptions of a service's inputs, outputs, preconditions, effects, and process model.

- BPEL4WS does not provide a well-defined semantics. Partners are restricted by structured XML content contained in WSDL port type definitions.

Fault handling, execution monitoring, and transactions:

- BPEL4WS defines a mechanism for catching and handling faults similar to common programming languages like Java. One may also define a compensation handler to enable compensatory activities in the event of actions that cannot be explicitly undone. DAML-S currently does not define recovery protocols but the formalised translations of DAML-S descriptions (as in [14] and [13]) may be extended to support them.

- Neither BPEL4WS or DAML-S directly support query mechanisms to expose the state of executing processes. BPEL4WS lists this item as a 'Future Direction'. Once again, the formalised translations of DAML-S descriptions may be extended to support execution monitoring.

- BPEL4WS may be extended with WS-Coordination [15] and WS-Transaction [16] to provide a context for pre-defined transactional semantics.

This is a preliminary sketch of the distinctions between BPEL4WS and DAML-S. However, we are examining the differences between the two specifications in detail and will update our findings in [17].

1.4 E-speak

E-speak is one of the earlier service architectures, developed by Hewlett-Packard. E-speak and UDDI have similar goals in that they both facilitate the advertisement and discovery of services. E-speak is also comparable to WSDL in that it supports the description of service and data types [5]. It has a matching service to compare service request and service descriptions, which it does primarily on the basis of input-output and service types matching.

E-speak describes services (Resources in the e-speak world) as a set of attributes within several Vocabularies. Vocabularies are sets of attributes common to a logical group of services. E-speak matches lookup requests against service descriptions with respect to these attributes. Attributes take common value types such as String, Int, Boolean and Double. There is a base vocabulary which defines basic attributes such as Name, Type (of value String only), Description, Keywords and Version. Currently, there is no semantic meaning attached to any of the attributes. Any matching which takes place is done over the service description attributes which does not distinguish between any further subtypes. DAML-S had a much richer set of attributes, in DAML-S terminology, the input/output parameters, effects and additional functional attributes. In addition, dependencies between attributes and logical constraints on them are not expressible within E-speak.

Unlike UDDI, which was intended to be an open standard from the beginning, e-speak scores relatively low on interoperability. It requires that an e-speak engine be run on all participating client machines. Furthermore, although e-speak is designed to be a full platform for Web services and could potentially expose a execution monitoring interface, service processes remain a black-box for the e-speak platform and consequently no execution monitoring can be done.

1.5 ebXML

ebXML, being developed primarily by OASIS and the United Nations, approaches the problem from a workflow perspective. ebXML uses two views to describe business interactions, a Business Operational View (BOV) and a Functional Service View (FSV) [6] [7]. The BOV deals with the semantics of business data transactions, which include operational conventions, agreements, mutual obligations and the like between businesses. The FSV deals with the supporting services: their capabilities, interfaces and protocols. Although ebXML does not concentrate on only Web services, the focus of this view is essentially the same as that of the current DAML-S effort.

It has the concept of a Collaboration Protocol Profile (CPP) which allows a Trading Partner to express their supported Business Processes and Business Service Interface requirements [such that they are understood] by other ebXML compliant Trading Partners, in effect a specification of the services offered by the Trading Partner. A Business Process is a set of business document exchanges between the Trading Partners. CPPs contain industry classification, contact information, supported Business Processes, interface requirements etc. They are registered within an ebXML registry, in which there is discovery of other Trading Partners and the Business Processes they support. In this respect, UDDI has some similarities with ebXML. However, ebXML s scope does not extend to the manner in which the business documents are specified. This is left to the Trading Partners to agree upon a priori by the creation of a Collaboration Protocol Agreement.

In conclusion, the kind of functionality, interoperability and dynamic matchmaking capabilities provided by DAML-S is only partially supported, as the standards are currently positioned, by WSDL and UDDI. UDDI may become more sophisticated as it incorporates e-speak-like functionalities, but it will not allow automatic service interoperability until it incorporates the information provided by DAML-S.

References

[1] The UDDI Technical White Paper
`http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf`

[2] Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, Katia Sycara; "Importing the Semantic Web in UDDI" Forthcoming in Proceedings of Web Services, E-business and Semantic Web Workshop
`http://www-2.cs.cmu.edu/~softagents/publications.html`

[3] Web Services Description Language (WSDL) 1.1
`http://www.w3.org/TR/wsdl`

[4] Uche Ogbuji, Using WSDL in SOAP applications: An introduction to WSDL for SOAP programmers:
`http://www-106.ibm.com/developerworks/library/ws-soap/?dwzone=ws`

[5] E-Speak Architectural Specification Release A.0
`http://www.e-speak.hp.com/media/a0/architecturea0.pdf`

[6] The ebXML website. http://www.ebxml.org/

[7] David Webber and Anthony Dutton, Understanding ebXML, UDDI and XML/edi.

http://www.xml.org/feature_articles/2000_1107_miller.shtml

[8] DAML Services

http://www.daml.org/services/

[9] Business Process Execution Language for Web Services, Version 1.0

http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/

[10] Tim Berners-Lee, James Hendler and Ora Lassila. The Semantic Web

http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21

[11] The DAML+OIL Language (March 2001)

http://www.daml.org/2001/03/daml+oil-index

[12] The Process Specification Language

http://ats.nist.gov/psl/

[13] Narayanan, S. and McIlraith, S. "Simulation, Verification and Automated Composition of Web Services". To appear in the Proceedings of the Eleventh International World Wide Web Conference (WWW-11), May, 2002.

http://www.daml.org/services/nar-mci-www11.ps

[14] Anupriya Ankolekar, Frank Huch, and Katia Sycara. "Concurrent Execution Semantics of DAML-S with Subtypes". LNCS 2342, p. 318 ff.

http://www.daml.org/services/ISWC2002-ExSem.pdf

[15] Web Services Coordination (WS-Coordination) 9 August 2002

http://www-106.ibm.com/developerworks/webservices/library/ws-coor/

[16] Web Services Transaction (WS-Transaction) 9 August 2002

http://www-106.ibm.com/developerworks/webservices/library/ws-transpec/

[17] Comparison of DAML-S and BPEL4WS (initial draft)

http://www.ksl.stanford.edu/projects/DAML/Webservices/DAMLS-BPEL.html