Towards a Semantic Choreography of Web Services: from WSDL to DAML-S.

Massimo Paolucci[1], Naveen Srinivasan[1], Katia Sycara[1], Takuya Nishimura[1,2]

[1]The Robotics Institute, Carnegie Mellon University, USA

[2]Media Technology Development Division, SONY Corporation, Japan

{paolucci,naveen,katia,nishi}@cs.cmu.edu

***Abstract*— The relation between DAML-S, a language for the description of Web services grounded in the Semantic Web, and the growing Web services infrastructure based on WSDL is, by an large, still an open question. In this paper we describe a mapping from WSDL to DAML-S whose contribution is twofold: on the theoretical side it clearly shows what information is contributed by the DAML-S specification, on a more practical side it facilitates the compilation of DAML-S descriptions.**

***Index Terms*—Web services, WSDL, DAML-S.**

I. Introduction

Existing specifications of Web services describe the primitive units of interaction. In the real world, there is a need to describe the ordering of business activities and their interactions in terms of lower level services and compose their execution. Such descriptions of Web service linkages and interactions have been described in industry using terminology such as orchestration, collaboration, coordination, composition and choreography. In this paper, we follow the definition of the Web services Architecture Working Group of the World wide Web Consortium (W3C) [11] and the definition in the recently chartered W3C Choreography Working Group [12] and call these set of activities *Choreography*. The activities comprising choreography can be different steps within a particular web service or belong to different web services. Current industry description languages, such as WSDL [4] have proven very useful in describing the interface of Web services. WSDL has recently started to be used extensively by industry (e.g. Amazon.com, Google, Acrobat to mention just a few). However, currently, natural language descriptions (understandable only by human programmers) must accompany WSDL descriptions in order to outline how to use the service (e.g. operation sequencing, state management), the participants' obligations, compositionality of results etc. It is therefore desirable to replace these natural language imprecise instructions with formal semantically meaningful and program understandable descriptions. Such precise specification could reduce the cost of businesses to integrate their processes using Web services. The Darpa Agent Markup Language for Services (DAML-S) [7] provides Web service providers with a core set of markup language constructs for describing the concepts and capabilities of their Web services in unambiguous and computer interpretable terms. The current paper reports on our work in developing and implementing a method and tool for translating WSDL descriptions to DAML-S descriptions.

WSDL provides declarative information to map abstract messages[1] into concrete messages and it expresses the bindings to specify the port where to post a message or to read the message from. In this capacity WSDL provides the foundation for composition of Web services, by providing the information that supports information exchange between Web services. But WSDL is not rich enough to specify the semantics of the composition or of the interaction protocol needed for composition. In contrast to WSDL, DAML-S, rather than describing Web services in terms of their ports or the messages that they receive, it describes the capabilities of Web services in terms of the abstract function that they provide, their Process Model, (ie what the workflow of the service steps is) and the Grounding, which describes how services interact. WSDL and DAML-S are complementary to each other: DAML-S provides the abstract information about composition of operations and information exchange, while WSDL describes how such abstract information is mapped into actual messages and how these messages are transmitted. Because of its compementarity with DAMLS, WSDL has been incorporated in the specification of the DAML-S Grounding to provide binding information.

Given the above analysis, a top-down approach, which first

[1] WSDL supports two modes of interaction between Web services: Remote Procedure Call (RPC) and asynchronous messaging. The distinction between them is not relevant for this paper; with *message* we intend any information exchange between Web services which may or may not be implemented using RPC.

formalizes the abstract information exchanged in DAML-S, and then specifies WSDL to meet the needs of the DAMLS specification, seems the correct way to represent Web services. Yet many times the opposite path is more appropriate: given a WSDL specification, the problem becomes to enrich it and transform it in a DAML-S specification. This process has both theoretical and practical value. On the theoretical side, it pinpoints exactly the information that is added by DAML-S specifications, which cannot be derived from WSDL. On the practical side, the description of a mapping from WSDL to DAML-S simplifies the compilation of DAML-S documents. Indeed, WSDL specifications of Web services are widely available, providing an obvious starting point for their DAML-S formalization; furthermore WSDL specifications can be generated automatically using Java2WSDL [2].

The research contribution of this paper is the description and implementation of WSDL2DAML, a tool for the translation of WSDL into DAML-S specifications. WSDL2DAMLS takes as input a WSDL specification, and it returns as output a partial DAML-S description of the Web service. The translation is based on the assumption that there is a 1:1 correspondence between DAML-S atomic processes and WSDL operations, which allows a partial specification of DAML-S Process Models. In addition, WSDL2DAMLS generates a complete specification of the Grounding which is used to map DAML-S Atomic Processes to WSDL, a primitive DAML-S Profile, and a DAML ontology of concepts based on the data types adopted by WSDL. Such ontology can be used to complete the specification of the DAML-S Process Model and Profile as well as for mapping the new DAML-S specification into existing DAML ontologies. Such a completion requires information that is not contained in the WSDL document and it should be done through human intervention.

In the rest of the paper, we first give a brief overview of DAML-S; we then describe the algorithms behind WSDL2DAMLS and the assumptions behind such a tool. We will then describe how this tool has been used to map the WSDL of the amazon.com Web service into a DAML-S Web service. The result of this experiment is a DAML-S specification that when used by a DAML-S processor automatically generates a client for the Web service. Finally, we conclude with an analysis of the tool and future directions of this project.

II. DAML-S

DAML-S is emerging as a Web Services description language that enriches Web Services descriptions based on WSDL with semantic information from DAML [6] ontologies and the Semantic Web [3]. DAML-S is organized in three modules: a Profile that describes capabilities of Web Services as well as additional features that help to describe the service; a Process Model that provides a description of the activity of the Web Service provider from which the Web Service requester can derive the interaction; Grounding that is a description of how abstract information exchanges described in the Process Model is mapped onto actual messages that the provider and the requester exchange.

The DAML-S Profile describes the Web Service capabilities, as well as additional features of Web Services such as provenance and quality of cost specifications of the Web service. The role of the DAML-S Profile is to support different modalities of discovery [10] and to support the requester decision of whether to use a given Web service. DAML-S describes capabilities of Web Services by the transformation that they produce. This transformation is described at two levels: at the information level a set of inputs are transformed in a set of outputs; at the domain level a set of conditions become true, while others become false. For example, if we consider a travel booking Web service, at the information level it may require departure and arrival information and it provides a flight schedule and a confirmation number; while at the domain level it books a flight, generate a ticket, and charges a credit.

In addition to capabilities, DAML-S Profiles provides *provenance* information that describes the entity (person or company) that deployed the service; and *non-functional parameters* that describe features of the services such as quality rating for the service.

The second module of DAML-S is the Process Model; the Process Model fulfills two tasks: the first one is to specify the interaction protocol in the sense that it allows the requester to know what information to send to the provider and what information will be sent by the provider at a given time during the transaction. In addition, to the extent that the provider makes public its own processes, it allows the client to know what the provider does with the information.

A Process Model is defined as an ordered collection of processes, where each process produces a state transformation or a data exchange with the Web service clients. The DAML-S Process Model distinguishes between two types of processes: composite processes and atomic processes. Atomic processes correspond to operations that the provider can perform directly. Composite processes are used to describe collections of processes (either atomic, or composite) organized on the basis of some control flow structure. For example, a sequence of processes is defined as a composite process of type sequence. Similarly, a conditional statement (or *choice* as defined in DAML-S) is also a composite process. The DAML-S process model allows any type of control flow structure including loops, sequences, conditionals, non-deterministic choice and concurrency.

The last module of DAML-S is the Grounding that describes how atomic processes; which provide abstract descriptions of the information exchanges with the requesters, are transformed into concrete messages that can be exchanged on the net, or through procedure call. Specifically, the DAML-S Grounding is defined as a one to one mapping from atomic processes to WSDL specifications of messages. From WSDL it inherits the definition of abstract message and binding, while the information that is used to compose the messages is extracted by the execution of the process model.

III. WSDL2DAMLS

The goal of WSDL2DAMLS is to provide a translation between WSDL and DAML-S. The results of this translation are a complete specification of the Grounding and an incomplete specification of the Process Model and Profile. The incompleteness of the specification is due to differences in information contained in DAML-S and WSDL. Specifically WSDL does not provide any process composition information, therefore the result of the translation will also lack process composition information; furthermore, WSDL does not provide a service capability description, therefore the DAML-S Profile generated from WSDL is also necessarily sketchy and must be manually completed. Nevertheless the outputs of WSDL2DAMLS provide the basic structure of a DAML-S description of Web services and saves a great deal of manpower[2].

The mapping produced by WSDL2DAMLS is roughly based on the following two observations.

1. *A WSDL operation is equivalent to a DAML-S atomic process:* in other words we can guess the atomic processes of the DAML-S Process Model from the operations of the WSDL description.
2. *XSD types are realized as DAML concepts:* DAML-S descriptions make use of DAML concepts to specify the content of inputs and outputs, while WSDL makes use of XSD types to specify inputs and outputs. Since the DAML-S Grounding that specifies the mapping between DAML-S Process Models and WSDL does not provide any mapping from concepts to types we are forced to assume a 1:1 correspondence between them.

The first observation provides the basic mapping between WSDL and DAML-S. It is used for both the generation of the basic Process Model and for the generation of the Grounding. The second rule generates the basic data used by DAML-S. The two rules correspond to the two main modules of translation as shown in figure 1.

Upon loading a WSDL file, WSDL2DAMLS first uses the *XSD→DAML Converter* to translate XSD types into the corresponding DAML concepts; then it uses the constructed mapping in the *Operation Converter* to translate WSDL operations into DAML-S atomic processes, generating the Grounding as well as a rough Profile. In the next two sections we describe the two modules in details.

[2] The translation of a complex WSDL document such as the specification of the Amazon Web service takes about a week of man time, where most of the time is spent dealing with the syntactic transformation from WSDL to DAML-S and only a few hours to construct the composition of processes in the Process Model and compiling the description of the Profile. Using WSDL2DAMLS the syntactic translation takes less than a minute and the programmer can finally concentrate on the Process Model composition and Profile description. We observed that the tool significantly reduced the time and effort to create the DAML-S augmented web service. We measured that the tool generated 100% of the DAML-S Grounding, 90% of the needed Process Model information.

A. *XSD→DAML Converter*

The task of the *XSD→DAML Converter* is to translate the XSD types defined in the WSDL specification into corresponding DAML ontologies. There are two design alternatives. The first is to generate DAML-S specifications that make use of XSD types and no use of DAML ontologies. This solution would prevent any type of reasoning about the concepts used in the DAML-S specification [9]. The second alternative is to force a translation to generate concepts that could be totally unrelated to ontologies available in the Semantic Web and therefore effectively useless from the automatic reasoning point of view. We chose the second solution because it allows programmers or automatic ontology-mapping programs to map the generated ontologies to existing ontologies on the Semantic Web.

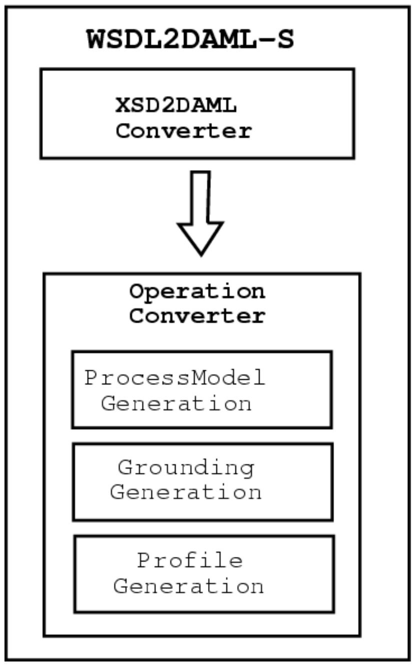

Figure 1. WSDL2DAMLS Architecture

XSD2DAML converter parses the WSDL file and extracts the XSD definitions defined between the WSDL type tags. The extracted XSD definitions are converted into DAML classes.

The conversion process is defined as follows:

- Primitive XSD types like string and integer are not converted to DAML definitions, rather they are defined directly as inputs or outputs of atomic process in the process model file.
- Complex XSD types are translated into DAML concepts whose properties correspond to the elements in the translated type.

This translation generates correct DAML ontologies, which, as described above, need to be mapped onto existing ontologies in the Semantic Web to become useful for automatic process composition.

B. Operation Converter

The conversion of the WSDL operations into DAML-S processes is based on rule 1. mentioned above where the basic idea is that WSDL operations map into DAML-S atomic processes, with the result that the WSDL **`portType`** description defines a primitive Process Model. The complete mapping is described in figure 2.

The mapping of WSDL Operations into DAML-S Atomic Processes is realized in the following way:

- *The name of the operation becomes the name of the corresponding atomic process*
- *The inputs messages of the operation become the inputs of the atomic process*
- *The outputs and faults messages of the operation become the outputs of the atomic process*

Once the atomic processes are generated WSDL2DAMLS proceeds with the specification of the grounding. This specification is quite straightforward since all the pieces are in place and we make explicit the link between the operations and their parts with atomic processes and their parts. In contrast to other DAML-S modules, the Grounding is completely specified during the translation and it should not require any modification.

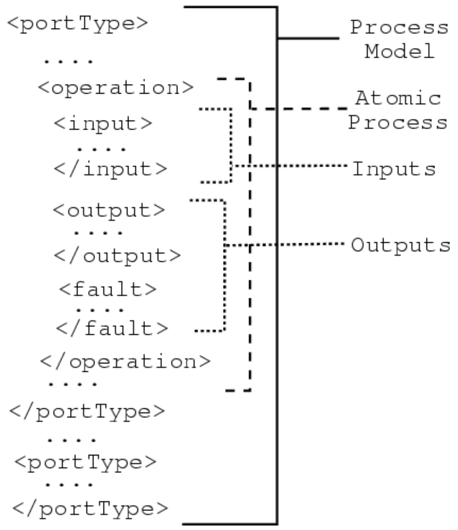

Figure 2: Operation Translation

One issue with the mapping described here is that DAML-S Atomic Processes may include inputs and outputs that are local to the process, and not reflected in the WSDL document. These inputs and outputs should be added by a programmer when she refines the DAML-S description. Furthermore the list of atomic processes may not be complete since DAML-S allows the addition of atomic processes that do not generate any message, but provide (partial) visibility on the internal processes of the Web service.

C. Service Profile Generator

The last translation performed by WSDL2DAMLS is the generation of the Service Profile; the result of this transformation is a skeleton Profile. DAML-S Profiles consist of three sections: the first one is *provenance* information that describes the entity (person or company) that deployed the service; the second consists of *non-functional parameters* that describe features of the services such as quality rating for the service and finally it provides a description of the capability/functionality of the service in terms of the inputs it receives, the outputs it generates, the furthermore, the preconditions that should be satisfied for the Web service to execute and the effects that will result as consequence of such execution. Since WSDL provides only input and output information, the rest of the DAML-S profile must be completed manually.

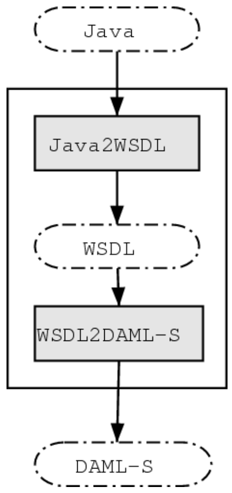

Figure3: From Java to DAML-S

One important contribution of the work described here is that it provides the basis for an automatic generation of DAML-S specifications starting from the Java implementation of the Web service. The complete process is described in figure 3. The first step is to use JAVA2WSDL [2] to generate the WSDL specifications directly from the Java code, and then compile the WSDL specification into the corresponding DAML-S specifications using the tool described here.

IV. Case Study: Amazon Web Service

Amazon.com has exposed a Web service [1] that allows to browse Amazon's catalogue and to fill a shopping cart with books that a customer would like to buy. Specifically, Amazon's Web Service allows 16 different varieties of searches (e.g. author search, actor search, ISBN search etc.) to browse and find items in their inventory, and five shopping cart operations including getting and clearing the shopping cart, add, removing and modifying items from the shopping cart.

The WSDL description of Amazon's Web service contains the 21 operations described above, 66 types, and 42 input/output messages (there is no use of fault messages). Using WSDL2DAMLS we were able to compile the WSDL specification in the corresponding DAML-S specification, which resulted of course in the 4 files describing the different aspects of DAML-S: Profile, Process, Grounding and a file that represents the XSD types into DAML concepts. After this translation a programmer is left with only three tasks to complete the DAML-S specification: the first one is to analyze the interaction flow between Amazon's Web service and its clients to complete the Process Model; the second one is to map the concepts derived from the XSD types into concepts in existing ontologies; finally the third task is to complete the specification of the DAML-S Profile adding provenance information and the non-functional parameters.

The first task proves to be quite simple since the interaction flow is easily extracted from the control flow described in the documentation of the Web service, and it resulted in the addition of 5 composite processes. The other two tasks are more challenging because the mapping between the types used in the WSDL file are quite arbitrary and do not match with any ontology for bibliographic information such as the Dublin Core [8]. For example, one type is called *AuthorArray*, which has properties specific to both author and Amazon web service, and does not have any obvious semantic status. The Profile is also difficult to generate because the concepts used in the Profile are also arbitrary as a consequence of the ontologies used. We believe that the arbitrariness of the concepts used is a consequence of the general attitude in the construction of the Web service, which is targeted to human users rather than to automatic interaction with other Web services which is what DAML-S attempts to facilitate; furthermore, it is a consequence of the expectation that any activity involving the Web service will be mediated by programmers that will hardcode the connection between their Web services and Amazon's. Once DAML-S or other semantically oriented technologies become widespread, more principled use of ontologies can be expected.

V. Conclusions

The contribution of this paper is twofold: on the practical side it describes a tool for generating DAML-S descriptions of Web services starting from its WSDL description; on the theoretical side it highlights the contribution provided by DAML-S to the description of Web services.

Upon deciding to extend a WSDL specification into a DAML-S specification, a programmer can use WSDL2DAMLS described in this paper to generate the DAML and DAML-S code. The result of the translation is a first approximation of the final DAML-S description; which contains all the DAML-S modules, but still requires work to be completed.

The first task of the programmer is to map the ontologies generated into existing DAML ontologies. Our example shows that such mapping may prove very challenging since the XSD types may not have any ontological status.

The second task is to construct the composite processes that complete the Process Model. These composite processes specify the order of execution of the WSDL operations to implement the expected interaction protocol.

The Process Model may contain some atomic processes and information flow that are private of the Web service and not shared with its clients. Since this information is not reflected in the WSDL specification, the third task is to add it to the DAML-S description under construction.

Once the Process Model is completed, the programmer should complete the DAML-S Profile. Specifically, she needs to select which input and output information better specifies the service provided. This process may require the removal of some inputs and outputs and the generalization of others. Furthermore, the programmer needs to add other two pieces of information; the first one is provenance information that specifies the company or institution that provides the service; the second one is information about non functional parameters and service classification.

The work described in this paper may be generalized to include Web services information contained in UDDI [13] as well as BPEL4WS [5]. While this mapping would be a natural extension of the work presented here, some predictions on its results can already be drawn. The addition of this information may add some automatism to the translation: UDDI provides provenance information possibly some functional parameters, while BPEL4WS provides processes composition. Yet, neither of the two provides semantic information so the XSD$\rightarrow$DAML mapping described above would still be needed with the relative mapping to existing ontologies; furthermore, neither of the two provides capability description; which would still require the programmer intervention.

VI. Bibliography

[1] Amazon.com: *Web Services V.2.0*: http://associates.amazon.com/exec/panama/associates/ntg/browse/-/1067662/ref=gw_hp_ls_1_3/

[2] Apache Software Foundation: *WSIF*: http://ws.apache.org/wsif/

[3] T. Berners-Lee, J. Hendler, and O. Lassila.: *The semantic web*.: Scientific American, 284(5):34--43, 2001.

[4] E. Christensen, F. Curbera, G. Meredith, and S.Weerawarana.: *Web Services Description Language (WSDL)*: http://www.w3.org/TR/2001/NOTE-wsdl-20010315 2001.

[5] F. Curbera, Y. Goland, J. Klein, Microsoft, F. Leymann, D. Roller, S. Thatte, and S. Weerawarana: *Business Process Execution Language for Web Services, Version 1.0*: http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/

[6] DAML Joint Committee.: *Daml+oil (march 2001) language*.: http://www.daml.org/2001/03/daml+oil-index.html 2001

[7] DAML-S Coalition.: *Daml-s: Web service description for the semantic web*: In ISWC2002.

[8] Dan King: *The Dublin Core Element Set Ontology*: http://www.daml.org/ontologies/201

[9] M. Klein, D. Fensel, F. van Harmelen, and I. Horrocks. *The relation between ontologies and xml schemas*. In Electronic Trans. on Artificial Intelligence, 2001.

[10] M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara.: *Semantic matching of web services capabilities*. In ISWC2002, 2002

[11] W3C Web Services Architecture Working Group: *Web services Glossary*: http://www.w3.org/TR/2002/WD-ws-gloss-20021114/

[12] W3C Web services Choreography Working Group: *Charter*: http://www.w3.org/2003/01/wscwg-charter

[13] UDDI: The UDDI Technical White Paper.: http://www.uddi.org/ 2000.