CODE: A Development Environment for OWL-S Web services

Naveen Srinivasan[1], Massimo Paolucci[1], and Katia Sycara[1]

[1] The Robotics Institute, Carnegie Mellon University,
Pittsburgh, PA 15213, USA
{naveen, paolucci, Katia}@cs.cmu.edu

Abstract. The generation of Semantic Web services is a complex and error prone process. CODE, the system that we intend to demo, is an Integrated Development Environment that supports the developer through the whole process from the Java generation, to the compilation of OWL-S descriptions to the deployment and registration with UDDI.

1 Introduction

To implement an OWL-S Web service, a Web Services developer engages in many different activities: first she implements the Web service; second, she provides the WSDL description, third she compiles the OWL-S description of the Web Service. The latter step requires the compilation of a Process Model that is faithful to the actual Web service implementation, a Profile for discovery and a Grounding that maps the Process Model to WSDL. This process is very time consuming and error prone, and the few tools that are available to support the developer do not form a consistent suite, therefore, they are very difficult to use on a consistent basis.

CODE (CMU's OWL-S Development Environment) addresses the problems of the developer by providing a uniform integrated development environment. CODE supports the developer from the Java development to the generation of the OWL-S descriptions, to the deployment and registration of the Web service with UDDI. Furthermore, through its editing facilities, CODE guarantees the syntactic correctness of the service description, and it allows the developer to use the SPIN model checker[11] to verify correctness claims about the control flow of the OWL-S Process Model. As a result CODE helps the developer to detect problems at development and compilation time, reducing the likelihood of execution time errors.

The guiding principle of the design of CODE is to integrate the tools that the developer needs during the implementation, description and deployment of Semantic Web services, in a single consistent and extensible environment. The consistency of the development tools allows the developer to move seamlessly between the different aspects of Semantic Web services development, while the extensibility of the environment allows other parties to provide additional contributions.

2 Walkthrough of the demo

During the demo we will show how CODE supports the developers in the complete development lifecycle of both real Web services such as the one provided by the Amazon Bookstore[1] as well as fictitious Web services such as the BravoAir example provided by the OWL-S coalition. Specifically, we will use CODE to generate the OWL-S descriptions of those Web services starting from a Java interface or the WSDL specification. In the process, we will use those specifications to deploy the service, advertise with an OWL-S based UDDI[2] as well as to generate clients that can interact with the Web service.

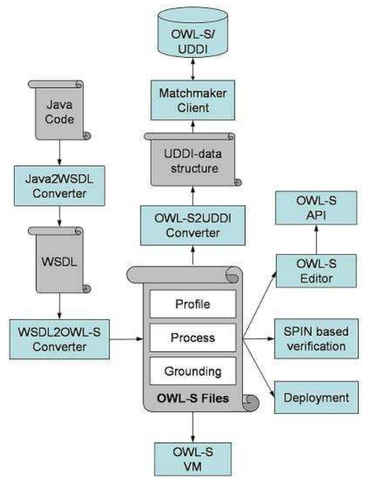


Figure 1: the Web services life cycle

Figure 1 shows the life cycle of Web service development. The boxes in the figure represent activities that the developer needs to perform and/or tools that are available to partially automate such activities; while the scrolls represent code or data that the developer needs to generate, possibly with the assistance of the tools.

2.1 Generation of WSDL and OWL-S

The first activity of the developer is the generation the Java[3] code that implements the Web service. This is supported by the *eclipse IDE*[4]. Once the code is generated, the developer generates the WSDL description of the Web service with the support of Apache's *Java2WSDL*[5] and the OWL-S description with the support of *WSDL2OWL-S*[6]. The results of this process are a complete WSDL, OWL-S Grounding descriptions, and a skeletal OWL-S Process Model and Profile. While WSDL2OWL-S greatly facilitates the developer activities, many aspects of the description are left to be complited. Specifically, the developer needs to add control flow and data flow information as well as to complete the Profile non-functional parameters.

---

[1] http://www.amazon.com/gp/aws/sdk/

[2] http://www.daml.ri.cmu.edu/matchmaker

[3] In principle the developer may use any other language, but we indicate Java here because eclipse, on which CODE is based, is mainly a Java IDE.

[4] www.eclipse.org

[5] http://ws.apache.org/axis/

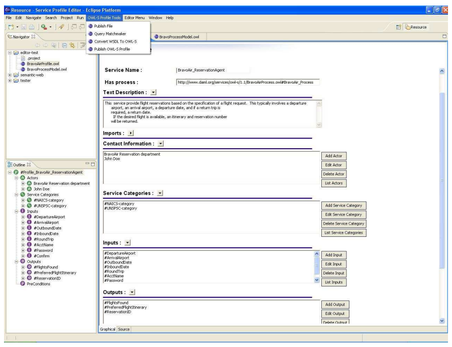
[6] http://www.daml.ri.cmu.edu/wsdl2owls

2.2 Editing of OWL-S

The schematic OWL-S generated by the WSDL2OWL-S tool can be completed using the OWL-S editor. CODE provides two forms of editing for OWL-S ontologies, the first one is form based, as shown in Figure 2 and Figure 3, the other is text based by allowing the developer to edit directly the OWL-S source.

The frame-based editor provides guidance to the developer on what information should be added at each stage of the compilation of the OWL-S description. For instance, in the compilation of a process, it requires the developer to enter the inputs, outputs, preconditions and effects. In turn, each one of them is a form that requires the developer to enter the appropriate information. If the information entered is not correct the developer is flagged an error that she can immediately fix.

The text based editor is an extension of SWeDE[7], an eclipse-based OWL editor that can be used by more experienced developers to generate their OWL-S code more expeditiously as well as to generate ontologies that are used to describe concepts that are specific of the Web service.



Service Name : BravoAir_ReservationAgent

Has process : http://www.daml.org/services/owl-s/1.1/BravoAirProcess.owl#BravoAir_Process

Text Description :

This service provide flight reservations based on the specification of a flight request. This typically involves a departure airport, an arrival airport, a departure date, and if a return trip is required, a return date. If the desired flight is available, an itinerary and reservation number will be returned.

Imports :

Contact Information :

BravoAir Reservation department
John Doe

Service Categories :

#NAICS-category
#UNSPSC-category

Inputs :

#DepartureAirport
#ArrivalAirport
#OutboundDate
#InboundDate
#RoundTrip
#AcctName
#Password

Outputs :

#FlightsFound
#PreferredFlightItinerary
#ReservationID

Figure 2: Profile Editor

The editing functions are based on the CMU OWL-S API that parses OWL-S definitions into Java objects extending the Jena OWL API[8].

2.2.1 Profile Editor

The Profile editor, shown in Figure 2, supports the developer in the following two tasks: the first one is the editing of the Service Profile of the Web service; the second one is the registration with an UDDI server.

The main pane of the window shown in Figure 2 displays the form-based editor that is used to compile the OWL-S Profile. Each field in the form corresponds to one attribute of the OWL-S Profile such as *contact information*, *service category*, *input* and *output*, etc. The buttons on the side of each field provide controls for adding, deleting and modifying various attributes of the OWL-S Profile.

The two smaller panes on the left side provide easy navigation functions that facilitate the operations of the developer. The top pane supports the file system navigation, while the bottom pane provides quick navigation within the OWL-S Profile by giving direct access to each input, output, category or the different aspects of the contact information.

---

[7] http://owl-eclipse.projects.semwebcentral.org/

[8] http://www.hpl.hp.com/semweb/jena.htm

2.2.2 Registration with UDDI

The Profile Editor also supports the developer through the discovery process by providing a direct registration and query to an UDDI server as well as the publication of the Profile on a public Web site. This process, that is supported by the open menu in Figure 2, provides functions such as *Publish Profile*. Before sending the Profile, CODE translates it in a UDDI compatible form using *OWL-S2UDDI*[9] and then automatically registers it with an UDDI server, such as the CMU's OWL-S/UDDI Matchmaker[10].

2.2.3 Process Editor

The Process Editor supports the developer in the generation of the Process Model using the same approach of the Profile editor. It provides a frame-based editor to define processes and their control and data flow, as well as a quick navigation panels.

In addition, the editor provides verification and execution functionalities. The developed Process Model can be verified using the Spin model checker[11], to eliminate any inconsistencies in the workflow. Likewise, the Process Model can be executed using OWL-S VM to eliminate any error during actual invocation of the Web service.

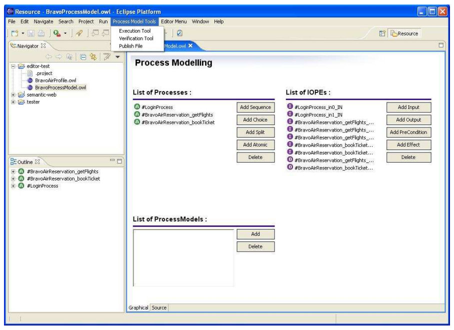

Resource - BravoProcessModel.owl - Eclipse Platform
File Edit Navigate Search Project Run Process Model Tools Editor Menu Window Help
Execution Tool
Verification Tool
Publish File
Resource
Navigator
editor-test
.project
BravoAirProfile.owl
BravoProcessModel.owl
semantic-web
tester
Outline
#BravoAirReservation_getFlights
#BravoAirReservation_bookTicket
#LoginProcess
Model.owl
Process Modelling
List of Processes :
#LoginProcess
#BravoAirReservation_getFlights
#BravoAirReservation_bookTicket
Add Sequence
Add Choice
Add Split
Add Atomic
Delete
List of IOPEs :
#LoginProcess_in0_IN
#LoginProcess_in1_IN
#BravoAirReservation_getFlights_...
#BravoAirReservation_getFlights_...
#BravoAirReservation_getFlights_...
#BravoAirReservation_bookTicket...
#BravoAirReservation_bookTicket...
#BravoAirReservation_getFlights_...
#BravoAirReservation_bookTicket...
Add Input
Add Output
Add PreCondition
Add Effect
Delete
List of ProcessModels :
Add
Delete
Graphical
Source

Figure 3: The Process Model Editor

3. Conclusion

The description and deployment of a Web service are very complex and time consuming processes that are likely to result in errors. CODE, the tool that we hope to demo, supports the developer during Web service generation, description and deployment. In the demo, we will show how CODE helps the developer by guiding the Web service development and preventing or detecting syntactic and semantic errors.

CODE is currently under testing, but it will be available for beta testing by the time of the conference and it will be distributed through the SemWebCentral[12] repository.

[9] http://www.daml.ri.cmu.edu/owls2uddi
[10] http://www.daml.ri.cmu.edu/matchmaker
[11] http://spinroot.com/spin/whatispin.html
[12] http://www.semwebcentral.org