The OWL-S Editor – A Development Tool for Semantic Web Services

Daniel Elenius, Grit Denker, David Martin, Fred Gilham, John Khouri, Shahin Sadaati, and Rukman Senanayake [⋆]

SRI International, Menlo Park, California, USA
firstname.lastname@sri.com

Abstract. Semantic Web Services (SWSs) promise to provide solutions to the challenges associated with automated discovery, dynamic composition, enactment, and other tasks associated with managing and using service-based systems. One of the barriers to a wider adoption of SWS technology is the lack of tools for creating SWS specifications. OWL-S is one of the major SWS description languages. This paper presents an OWL-S Editor, whose objective is to allow easy, intuitive OWL-S service development and to provide a variety of special-purpose capabilities to facilitate SWS design. The editor is implemented as a plugin to the Protégé OWL ontology editor, and is being developed as open-source software.

1 INTRODUCTION

Semantic Web Services (SWSs) [1] introduce ontologies to describe, on the one hand, the concepts in the services' domains (e.g., flights and hotels, tourism, e-business), and on the other hand, characteristics of the services themselves (e.g., control flow, data flow) and their relationships to the domain ontologies (via inputs and outputs, preconditions and effects, and so on). These semantically rich descriptions enable automated machine reasoning over service and domain descriptions, thus supporting automation of service discovery, composition, and execution, and reducing manual configuration and programming efforts. The three most prominent SWS specification approaches currently under development are OWL-S [2], WSMO[1], and SWSL[2].

The field of SWSs is still in an early stage, and adoption has been slow. A limiting factor has been the lack of tool support. The objective has been to enable machines to manipulate services, yet so far arduous human work has been necessary to create the semantic service descriptions. While tools to create and edit Semantic Web ontologies in general do exist [3], modeling SWSs requires additional functionality and developer support in order to be practically feasible.

⋆ Supported by the Defense Advanced Research Projects Agency through the Air Force Research Laboratory under Contract F30602-00-C-0168 to SRI, and in part by Vinnova (grant no. 2002-00907) and The Swedish Research Council (grant no. 621-2003-2991).

1 `http://www.wsmo.org`

2 `http://www.daml.org/services/swsl/`

Tools that make the SWS technology accessible to a broad audience with diverse needs are a crucial factor in the success of SWS technology. Tools are needed to facilitate tasks such as service definition and annotation, execution and monitoring, and service registration and discovery. The OWL-S Editor is aimed at providing a flexible, yet powerful editor for OWL-S service definitions. This paper describes the design and current functionality of the OWL-S Editor as well as its future directions.

2 DESIGN PHILOSOPHY

There are two main tasks in the development of OWL-S services. The first task is to define the service's domain ontologies in terms of OWL classes, properties, and instances. The second task is to create an OWL-S description of the service, relating this description to the domain ontologies. An OWL-S service description consists of *instances* of OWL-S classes such as `Service`, `Process`, `Input`, and `Output`. In some cases, the OWL-S ontology is also *extended* to handle specific modeling situations.

In order to best facilitate these tasks, we built the OWL-S Editor on top of the Protégé OWL Ontology Editor [3]. Protégé allows editing of domain ontologies out-of-the-box. However, efficient development of services requires additional features. Our strategy has been to leverage the existing functionality of Protégé, and to utilize Protégé's pluggable architecture to extend it where we judged it would be helpful for the SWS developer. The result is a SWS development environment where the domain ontologies are well integrated with the service descriptions.

In addition, building our tool on top of Protégé means that users can take advantage of the many other existing Protégé plugins, e.g. for querying and visualizing the Knowledge Base (KB), and to export the KB to different formats. These different plugins coexist gracefully, all working on the same KB.

3 OVERVIEW OF FEATURES

The OWL-S Editor presents the user with a tab inside Protégé as the main point of interaction (see Figure 1). The OWL-S tab provides a more direct view of the OWL-S classes and instances than what Protégé provides by default. The OWL-S tab is separated into two parts. The left-hand side provides *instance panes* which allow users to easily navigate their service descriptions. The instance panes list all instances of a service, divided into service, profile, process, and grounding instances. The right-hand side of the OWL-S tab is an *editing pane* that changes depending on the selection in the instance panes, to show a specialized editing mode for the chosen type of OWL-S instance. For example, if the user selects a profile instance (used for service discovery), then the right pane will show all properties of the profile, allowing the user to create fine-tuned service advertisements. If a composite process is selected, the editing pane changes to a graphical process editor (see below).

Additional functionality provided by the OWL-S Editor includes:

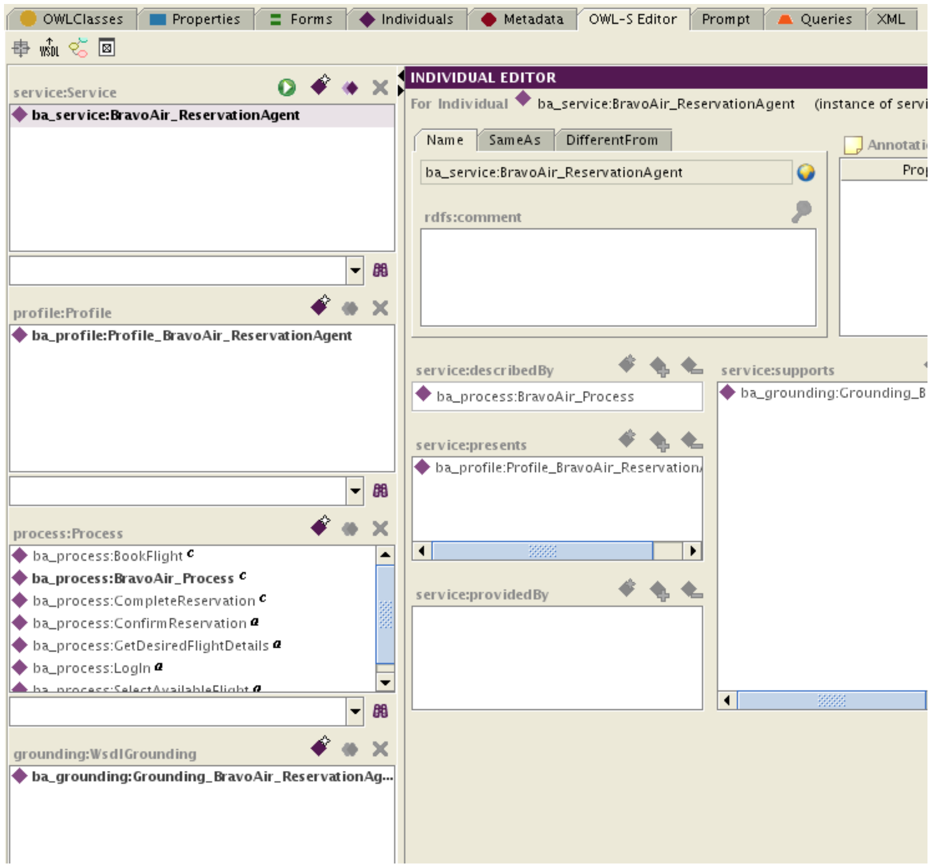

OWLClasses
Properties
Forms
Individuals
Metadata
OWL-S Editor
Prompt
Queries
XML
service:Service
ba_service:BravoAir_ReservationAgent
profile:Profile
ba_profile:Profile_BravoAir_ReservationAgent
process:Process
ba_process:BookFlight
ba_process:BravoAir_Process
ba_process:CompleteReservation
ba_process:ConfirmReservation
ba_process:GetDesiredFlightDetails
ba_process:LogIn
grounding:WsdlGrounding
ba_grounding:Grounding_BravoAir_ReservationAg...
INDIVIDUAL EDITOR
For Individual ba_service:BravoAir_ReservationAgent (instance of servi
Name
SameAs
DifferentFrom
Annotati
Prop
rdfs:comment
service:describedBy
ba_process:BravoAir_Process
service:supports
ba_grounding:Grounding_B
service:presents
ba_profile:Profile_BravoAir_Reservation
service:providedBy

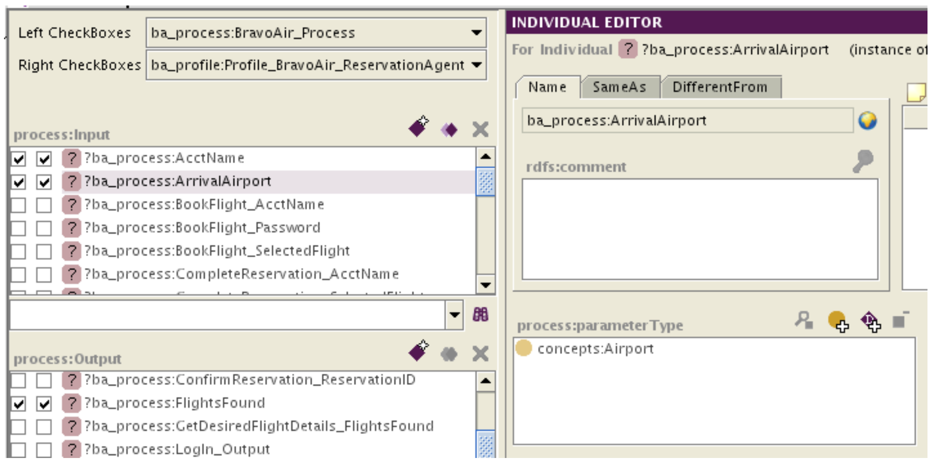
Fig. 1. The OWL-S Editor, a tab-widget plugin for Protégé, is shown here next to the standard Protégé-OWL tabs to the left, and, to the right, other tab-widget plugins for ontology management, queries, and XML management.

WSDL Support In many cases, it will be desirable to create a "skeletal" OWL-S description based on a preexisting WSDL file. Parts of the OWL-S description can be generated automatically based on the inputs and outputs defined in the WSDL file. To this end, we have integrated the WSDL2OWLS code that is part of the OWL-S API from Mindswap[3] into the OWL-S Editor. We also provide support for managing the mappings between the XSD datatypes of the inputs and outputs in the WSDL file, and the OWL classes in the OWL-S ontologies.

Input/Output/Precondition/Result Management OWL-S services are characterized by their inputs, outputs, preconditions, and results (IOPRs). A specialized window called the *IOPR Manager* (see Figure 2) allows users to a)

[3] `http://www.mindswap.org/2004/owl-s/api/`

get an overview of, and edit the properties of all IOPRs in the knowledge base (for preconditions and results, a semi-graphical SWRL expression builder will be used) and b) manage the sometimes complex relationships between how different processes and profiles utilize the IOPRs.



Left CheckBoxes ba_process:BravoAir_Process

Right CheckBoxes ba_profile:Profile_BravoAir_ReservationAgent

process:Input

✓	✓	?ba_process:AcctName
✓	✓	?ba_process:ArrivalAirport
☐	☐	?ba_process:BookFlight_AcctName
☐	☐	?ba_process:BookFlight_Password
☐	☐	?ba_process:BookFlight_SelectedFlight
☐	☐	?ba_process:CompleteReservation_AcctName

process:Output

☐	☐	?ba_process:ConfirmReservation_ReservationID
✓	✓	?ba_process:FlightsFound
☐	☐	?ba_process:GetDesiredFlightDetails_FlightsFound
☐	☐	?ba_process:LogIn_Output

INDIVIDUAL EDITOR

For Individual ?ba_process:ArrivalAirport (instance of

ba_process:ArrivalAirport

rdfs:comment

process:parameterType

concepts:Airport

Fig. 2. IOPR Manager

Graphical Overview A special window shows the relationships of all top-level OWL-S instances graphically. For example, this makes it easy to see all profiles for a given service, all groundings (descriptions of concrete implementation details) of a given process, and so on.

Execution We also provide an integrated *execution* environment for the OWL-S services being developed. Developers are thus able to verify that their specifications reflect their intentions, and to try out different possibilities before deploying their services. The execution environment takes the form of a window where users provide values for service inputs from the Protégé KB, and are provided with the outputs of the services.

Process Modeling A powerful feature of OWL-S is the ability to model composite processes. A composite process is constructed from subprocesses that can in turn be composite, atomic, or simple. The control flow of a composite process is defined using control constructs, such as If-Then-Else, Sequence, and Repeat-Until. These constructs can be nested to an arbitrary depth.

These control flows are particularly difficult to generate by hand or in a plain ontology editor not designed for this task. The OWL-S editor visualizes these control flows graphically, in a style similar to UML Activity Diagrams, using boxes for subprocess invocation (called Performs in OWL-S), diamonds for conditional nodes (e.g., for If-Then-Else constructs), and arrows showing the flow of execution. OWL-S control flows have more structure than arbitrary flow charts or UML activity diagrams, however. Therefore, we do not allow users to directly “draw” the work flow. Instead, we take advantage of the fact that all OWL-S control flows are *trees*. We let the user model the control flow in a GUI tree component, with drag-and-drop support, whereas the corresponding work flow graph is updated to reflect any changes to this tree (see Figure 3).

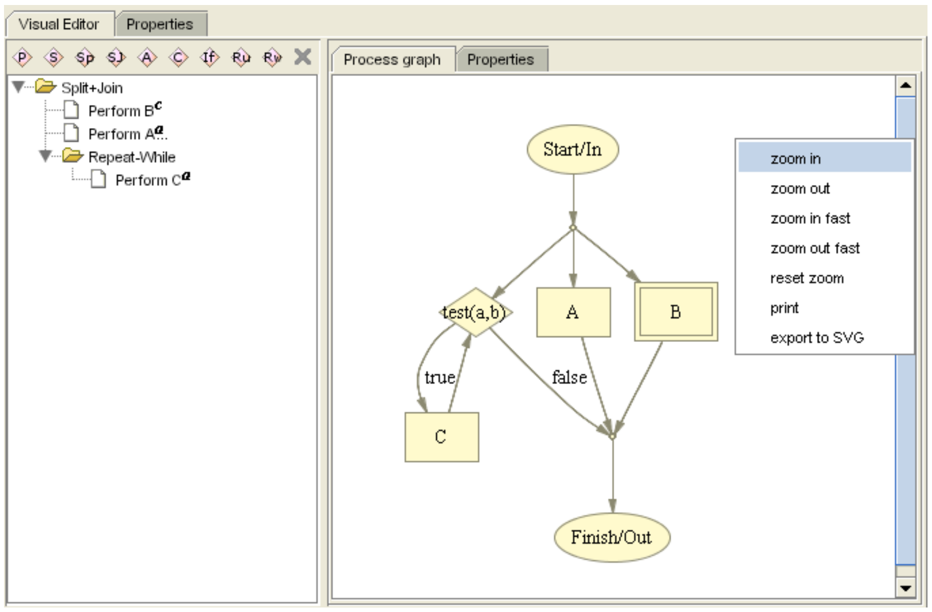


Fig. 3. A composite process, its tree structure shown to the left, and its graph representation to the right.

We can also specify the *data flow* of composite processes. For example, we can state that a certain input of Process B should be taken from a certain output of Process A. This is also supported by our graphical editor (not shown here).

4 CONCLUDING REMARKS

Our strategy has been to leverage the existing functionality of Protégé, and to utilize Protégé’s pluggable architecture to extend it where we judged it would be helpful for the SWS developer. The result is a SWS development environment where the domain ontologies are well integrated with the service descriptions.

Looking forward, there are several developments that would further our aims of easy and powerful Semantic Web Service development:

A highly desirable feature which we have not yet implemented is online search for services. Such a search facility could be used to find services or service components to be included as parts of a composite process that the user is working on in the OWL-S Editor. Ideally, the user should be able to give detailed search criteria, and find a service that matches her current needs (e.g. to find a service with inputs matching the outputs of previous processes in a composite process model). In conjunction with the built-in service execution, this would provide an extremely powerful development environment.

In the long term, we would also like to see a tighter connection between the semantic service markup, and the actual development of the service implementation (e.g. using Java). The OWL-S IDE project[4] is to some extent based on this idea, providing an Eclipse [5] plugin which can generate OWL-S "skeletons" directly from Java code. However, there is no feature for ontology editing in Eclipse. An integration of Protégé with Eclipse would be ideal.

We are also looking forward to the planned support for simultaneously editing multiple knowledge bases in Protégé. We often want to edit service components spread across different subontologies, and the domain ontologies are normally separated from the service descriptions.

In conclusion, providing this tool to the community, our aim is to make it easier to understand the concepts of SWSs, and to create semantic descriptions of services. We believe that this can bring a fruitful cross-pollination between practice and theory. As more people start developing SWSs, important feedback on using the service ontologies in various projects, and on design and implementation aspects of SWSs, could benefit the knowledge in this field.

The OWL-S Editor is available for download in both binary and source formats on `http://owlseditor.semwebcentral.org`. We welcome all feedback on our mailings list.

REFERENCES

1. McIlraith, S., Song, T., Zeng, H.: Semantic Web services. IEEE Intelligent Systems, Special Issue on the Semantic Web **16** (2001) 46–53
2. Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., Parsia, B., Payne, T., Sabou, M., Solanki, M., Srinivasan, N., Sycara, K.: Bringing semantics to Web Services: The OWL-S approach. In: Proc. First Intern. Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004), July 6-9, 2004, San Diego, California, USA. (2004) `http://www.daml.org/services/owl-s`.
3. Knublauch, H., Fergerson, R., Noy, N., Musen, M.: The Protégé OWL plugin: An open developoment environment for semantic web applications. In McIlraith, S., Plexousakis, D., van Harmelen, F., eds.: Proc. 3rd Intern. Semantic Web Conference (ISWC 2004), Hiroshima, Japan, November 2004, Springer (2004) 229–243 LNCS 3298.

[4] `http://projects.semwebcentral.org/projects/owl-s-ide/`, formerly known as CODE

[5] `http://www.eclipse.org`