The OWL-S Editor – A Domain-Specific Extension to Protégé

Daniel Elenius

SRI International, Menlo Park, California, USA
`elenius@csl.sri.com`

1 Introduction

Web service technology establishes a common, vendor-neutral platform for integrating distributed computing applications, in intranets as well as on the Internet at large. 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[1] 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. We introduce the main features of this tool, from the perspective of the Protégé user.

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 [2]. Protégé allows editing of domain ontologies out-of-the-box, and we added plugins for specific service-related editing tasks, wherever we deemed it useful.

3 Overview of Features

The OWL-S Editor consists of a *tab widget*, a number of *slot widgets*, and some additional functions reached from a toolbar on the tab widget.

The OWL-S tab serves as the user's main point of interaction, providing a more direct view of the OWL-S classes and instances than what Protégé provides by default. It contains four *instance panels* listing all instances of the main OWL-S classes: Service, Profile, Process, and Grounding. It also contains an *editing panel* that changes depending on the selection in the instance panels, 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 panel will show all properties of the profile, allowing the user to create fine-tuned service advertisements. If a composite process is selected, the editing panel changes to a graphical process editor (see below).

A number of slot widgets perform less obvious but nonetheless essential functions in OWL-S Editor.

Additional functionality provided by the OWL-S Editor includes:

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[1] into the OWL-S Editor. Slot widgets support management of 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 **??**) allows users to a) get an overview of, and edit the properties of all IOPRs in the knowledge base and b) manage the sometimes complex relationships between how different processes and profiles utilize the IOPRs.

Graphical Overview A special window shows the "forest" of 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

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

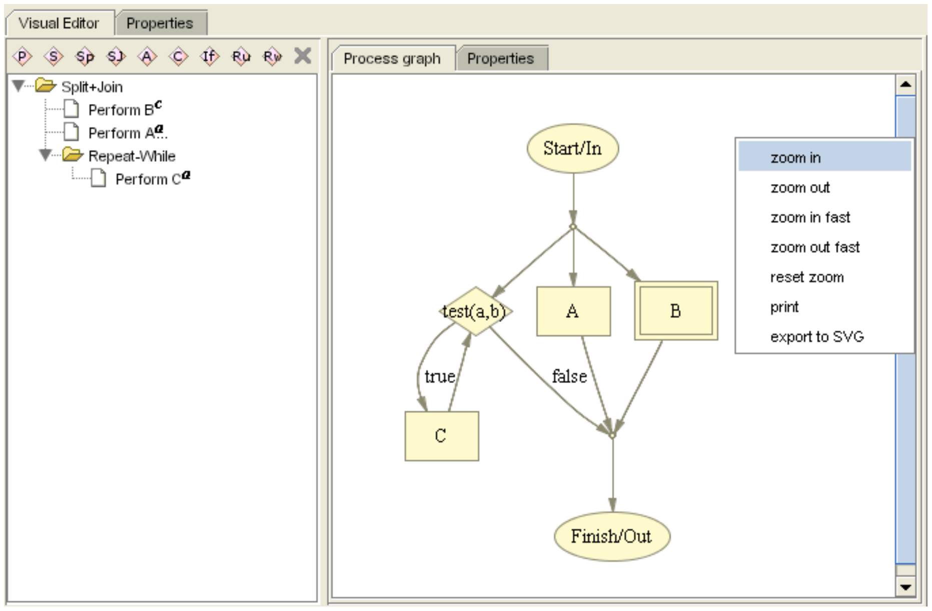

Fig. 1. A composite process, its tree structure shown to the left, and its graph representation to the right.

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. We visualize these control flows graphically (see Figure 1), 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[2]

In addition to control flow, we can specify the data flow of processes. For example, we can state that a certain input of Process B should be taken from a certain output of Process A (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.

[2] This feature makes use of the GraphViz package, `http://www.research.att.com/sw/tools/graphviz/`, also used by the OWLViz and OntoViz Protégé plugins.

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.

In the long term, we would 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[3] is to some extent based on this idea, providing an Eclipse[4] 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 for our purposes.

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.

Acknowledgements

This work is 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).

References

1. D. Martin et al: 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`.
2. 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.

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

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