CSage: Use of a Configurable Semantically Attributed Graph Editor as Framework for Editing and Visualization

Lawrence Levin

Critical Architectures LLC
ljlevin@critarch.com

Abstract. Easy to use tools tailored to a specific semantic model or purpose may greatly reduce the barriers to the use of knowledge driven applications. They are, however, time consuming to create. In this paper, we describe a framework for the creation of semantically aware, application-specific editing tools. We first present an approach based on the linkage of an ontological model to a graph-grammar describing the characteristics and intended usage of the tool. We then discuss our experience using this approach to develop a suite of editors for a military decision support system (DSS).

1 Introduction

The ability to formulate rules and logic for reasoning within the context of an ontology is critical to the development of decision support systems. These rules, however, are more mutable than the underlying ontological framework on which they rest, as users often wish to modify rules as a result of 'lessons learned' from operational usage, or to adapt to changing situations or local conditions[5].

This paper describes an approach to the development of an authoring environment for the specification of SWRL rules and BPEL workflows. Two editors, each tailored to its intended product, are built on a common framework, both in terms of software and of the underlying domain ontology. Both rules and workflows are for use in an antisubmarine warfare (ASW) decision support system.

2 Goals

Our goal is to provide operational users and subject matter experts (SME) with the ability to create and edit the rules and logic which will be executed by the semantic applications. While the inputs to these applications will be specified as formal constructs such as SWRL rules or BPEL workflows, we can not expect the users to master these specification languages. Instead, authoring environments are required that lower the barriers to entry for use by SME[4]. At the same time, these tools must provide an environment in which the constructs being created are both executable and of value. Flexibility and ease-of-use must be

Proceedings of OWL: Experiences and Directions 2009 (OWLED 2009), Rinke Hoekstra, editor.
http://www.webont.org/owled/2009

balanced with effectiveness[2]. A toolkit that is able to guide users through the process of specifying workflow and rules for an anti-submarine warfare (ASW) DSS should be easily adapted to providing the same capability in the context of a DSS for humanitarian relief operations.

3 Approach

To address the issues described above we have created the Configurable Semantically Attributed Graph Editor (CSage). The generic CSAGE framework may be configured to guide users through the steps of specifying BPM workflows, decision processes, and other forms of business logic using the concepts and terminology familiar to the users and the application domain. The output of CSage is the executable specifications (e.g., SWRL, BPEL) that will be used by the reasoning components of the target application. The core of CSage is a visual editor that allows the user to create, edit, and navigate large graph-based specifications and assign properties and values to nodes and edges. This generic functionality is, however, constrained by two data structures: a domain ontology and a graph-grammar bound to the ontology.

The graph grammar is the formal specification of how a typed graph may be constructed (i.e., the allowed/required linkages between various node/edge types). The grammar also indicates which types may be assigned properties, if values may be assigned to the properties, and if any cardinality constraints are defined (e.g., at least one property MUST be defined for nodes of type 'X'). Thus, to customize CSage for use as a process workflow editor, one could specify a graph grammar based on Business Process Modeling Notation (BPMN). To create a rule editor we may define a graph grammar suited to the specification of decision trees.[1]

A CSage graph grammar also includes the rules for mapping node/edge types onto an ontological model. This is done by using a model/view/context paradigm. A *view* is simply an arbitrary collection of properties defined within the model. A *context* is an indicator of how properties are to be managed by CSage. Examples of a context might be 'Inputs', 'Conclusions', or 'Decision-Oversight'. One context might require that the user assigns a value to each property, while in another, the user may only indicate that the property exists. Partitioning and specifying contexts and constraints provides instructions to CSage when it guides users through the editing process and again when it is validating and exporting the specifications to the downstream applications.

Thus, for each node/edge type defined in the graph grammar, one or more contexts may be identified along with the views that may be assigned to a context when working with a node/edge of the specified type. When a user selects a node and/or action in a logic diagram, the CSage software will use the grammar to determine what the appropriate options are and then enable, disable, and configure user interface components (e.g., menus, buttons) accordingly. In

[1] see [1] for examples.

particular, CSage will use the view information to decide what properties and property ranges are available for a specific action. Thus CSage is able to reduce the probability that a user will get "lost in ontology space"[3].

A graph grammar takes the form of an XML specification. The example shown is from a graph grammar used to specify SWRL rules for an undersea warfare decision support system (USW-DSS):

```
<GraphNode key="DFBranch" maxProp="1">
  <ViewMap context="Battlespace" values="N">
      <SetRef key="BSpace1" />
  </ViewMap>
  <linkset direction="to" edgeType="SeqFlow">
      <constraint context="usw" inherit="all" exclusive="Y"/>
      <NodeRef key="DFBranch" />
      <NodeRef key="DFQuant" />
  </linkset>
</GraphNode>
```

This example illustrates a key feature of the graph grammar: how one node or edge is bound to the ontology may be dependent on how the nodes or edges it is connected to have been bound. We have defined a node of type 'DFBranch' and indicated that an edge of type 'SeqFlow' may be used to connect a DFBranch to either another DFBranch node or a DFQuant node. Note, however, that a `<constraint>` element is defined which indicates that when a SeqFlow is specified it must inherit the same properties as those assigned to the DFBranch node that is its point of origin. Furthermore, while all SeqFlow originating from a common DFBranch must have the same properties, the constraint requires that the values assigned to each edge must be exclusive.

At any given point, the graph grammar is used to determine what options are available to a user. CSage guides the user by enabling, disabling, and configuring user interface components. A user may at any point request that a specification be validated and checked against the grammar for errors and omissions. Validation is also performed automatically prior to exporting a completed specification (e.g., as SWRL rules to Protégé). Only a specification that has passed validation may be exported.

A key point to note is that the graph grammar does not directly reference the domain ontology. Rather it refers only to views. Thus, it is possible to switch to a new domain (e.g., from ASW to disaster response) by adding the required view definitions to an appropriate ontology. This is a relatively simple procedure since, as noted previously, a view is nothing more then a named set of properties.

4 Results to Date

CSage has been developed for use with a decision support system (DSS) focused on ASW. In this context, two types of specifications must be produced. Logic specifications are intended for use by reasoning agents. These may be exported

in the form of SWRL rules which may then be loaded into a component based on the Protégé-OWL framework. Process/workflow specifications are used to coordinate the overall decision making process. These are exported in a BPEL-based syntax for use by orchestration and workflow management components. Both types of specifications use the same ASW domain ontology but have very different structures and concerns. Using CSage, however, we were easily able to create an editing tool tailored to the creation of each type of specification. Since both tools operate from the same base ontology, proper behavior from a system using the combined exported constructs is easier to ensure. At the same time, the ability to create an editor tailored to the specific role and concerns of a given type of user can significantly lower the barriers to entry and increase user acceptance.

In terms of user acceptance, the key is the proper specification of the views. Once a view is assigned to a node's context, the user is constrained by that view when assigning properties. While beneficial in terms of reducing the effort of working with large ontologies, this can be an obstacle if a user wishes to make use of a property that is not included in the assigned view. Thus, when defining views into an ontology, it is important to strike a balance between those that are too restrictive and those that are so broad as to provide no added value in terms of ontology navigation, guidance of user actions, or validation.

5 Conclusion

We have described an approach to developing authoring environments that can reduce the effort associated with working with large and complex ontologies. The linkage of a graph grammar with an ontology using the model/view/context paradigm allows for the creation of interactive tools that can simplify the complex process of logic specification. Simultaneously, the developer benefits from the ability to easily create high-value knowledge engineering tools.

References

1. http://www.criticalarchitectures.com/projects/CSage/owled2009/
2. D. McGuinness, P. Fox, L. Cinquini, P. West, J. Benedict, and J. Garcia: Current and future uses of OWL for Earth and Space science data frameworks: successes and limitations. OWLED 2007, Innsbruk Austria
3. M. Wessel and R. Moller: Design Principles and Realization Techniques for User Friendly, Interactive, and Scalable Ontology Browsing and Inspection Tools.OWLED 2007, Innsbruk Austria
4. M. O'Connor, C. Nyulas, R. Shankar, A. Das, and M. Musen: The SWRLAPI: A Development Environment for Working with SWRL Rules. OWLED 2008, Karlsruhe Germany
5. S. Stoutenburg, L. Obrst, D. Nichols, K. Samuel, and P. Frank: Dynamic Service Oriented Architectures through Semantic Technology. Springer-Berlin Service-Oriented Computing – ICSOC 2006 4th International Conference, Chicago, IL.