The Ontology Definition Metamodel (ODM) and Motivation for a Semantic Meta-Object Facility (SMOF)

Elisa Kendall, Mark Dutra
Sandpiper Software, Inc., Los Altos CA

Jay Jacobs, Stephen Schwab
Sparta, Inc., El Segundo, CA

{ekendall@sandsoft.com, med@sandsoft.com, jay.jacobs@sparta.com, stephen.schwab@sparta.com }

Abstract. The Ontology Definition Metamodel (ODM)[1] has been used as a basis for ontology development as well as for generation of OWL ontologies and Java-based production rules that together form a part of the run-time fabric of a context-aware application. Issues encountered in ODM-based ontology, source code and rules generation highlight the need for enhancements to the Meta Object Facility (MOF) to support multiple classification and properties as first class citizens. Work-arounds have been proposed as a stop-gap in the ODM, but the problems we have encountered are not unique to ontology development – they are prevalent in business process, conceptual, and general object-oriented modeling as well.

1 Application Development Environment

Complex computing environments provide significant challenges for current semantic interoperability research, currently limited in nature and scope to relatively self-contained projects within bounded domains. Key barriers to broader interoperability include: (i) The heterogeneity of information representations, computer systems, and business processes; (ii) Dynamic, situation- and context-specific requirements for information; and (iii) The perceived, often actual, conflict between local and global requirements for information discovery, information fusion, and information sharing. Semantic web and related knowledge representation (KR) and reasoning technologies address some of these interoperability issues, but significant gaps remain. Multidisciplinary approaches blending sophisticated KR technologies with new research and best practices from software engineering are likely to be most successful in filling these holes.

Sandpiper's implementation of the Ontology Definition Metamodel (ODM) is used for ontology development and analysis on research in context-aware systems. Ontology components are developed in UML and exported in OWL for analysis using OWL DL reasoners. Validated vocabularies are then directly exported as Java source code

[1] Draft specification is available at: http://www.omg.org/cgi-bin/doc?ad/06-05-01.

components (primarily classes and interfaces, augmented with methods to enforce restrictions) for use at run time by services that leverage production rules engines such as the Java Expert System Shell (JESS) and Fair Isaac's Blaze Advisor. This approach was selected after determining that use of JESS as an extension to other frameworks, such as Protégé's JESSTAB, was insufficient in scalable architecture and real-time performance.

2 Generating Java Source Code from OWL Ontologies

One of the challenges in Java source code generation is determining how to deal with multiple inheritance. One option would be to generate a library specifically for a single ontology, optimized for the target run-time application, with little consideration of reusability. Alternatively, the approach we advocate is that the Java components should be as reusable as the ontologies they reflect. Thus, each ontology class has a Java interface and matching implementation. Only the classes needed are loaded in the JVM at run-time, keeping the memory footprint to a minimum. Also, method overloading requires a common return type; however, methods can be generated to enforce type checking while still using the generic classes.

Open research questions include (1) a descendant inheriting the same method from different ancestors, and (2) how best to generate/manage Java objects for individuals. Some production rule engines perform with decreasing efficiency as the number of facts increases. Decisions on how to address this are complicated by the fact that ontologies may contain large numbers of 'reference facts'; some contain facts that are useful for analysis, consistency checking, co-reference, or other purposes but are not used by production rules. Ideally in our case, individuals will be asserted dynamically and cached in a knowledge base. A set of house-keeping rules may be applied to periodically predict whether cached facts are still needed and retract them when feasible.

3 Motivating Semantic MOF

The Meta-Object Facility (MOF) specifically prohibits multiple classification in MOF reflection, which required a number of work-arounds for the ODM and reflects the Java programming bias of MOF authors. An RFP[2] recently issued by the OMG requests proposals to augment MOF for ontology, business process, or other general conceptual modeling paradigms. We plan to participate from an end user perspective and would like to solicit additional requirements from this community in order to maximize the impact of the projected change.

[2] See http://www.omg.org/cgi-bin/doc?ad/06-06-08.