# SWAP - A Framework for Ontology Support in Semantic Web Applications

Arijit Sengupta
Indiana University
Kelley School of Business
Bloomington, IN 47408, USA

Henry Kim
York University
Schulich School of Business
Toronto, Ontario M3J 1P3, Canada

**Abstract**
We present SWAP (Semantic Web Application Pyramid), a framework for incorporating ontologies in data-oriented semantic web applications. We have implemented this framework with a measurement ontology for a quality management web service. This quality management web service is built on top of a set of XML web services implementing agents representing quality management clients, quality management servers and vendors. SWAP facilitates data exchange between these web services with vendor data stored in databases, and the processing of the data using a combination of RuleML and SQL. The test bed implementation demonstrates the feasibility and scalability of the framework for any type of three-tier ontology-based semantic web applications involving low to moderate data exchange. We discuss methods for improving this framework for high data exchange volumes as well. The primary contribution of this framework is in the component-based implementation of real world semantic web applications.

## 1 Introduction

The semantic web, introduced by Berners-Lee [1] opens the door to intelligent web applications. The concept of the semantic web is still evolving, and needs the integration of several key technologies such as databases, XML web services, and rule processing. We present SWAP (Semantic Web Application Pyramid) - a framework with a three-tier architecture for developing semantic web applications, with or without agent technology, and possible integration to databases. To demonstrate the applicability of this framework, we use a measurement ontology to create a quality management web service for the semantic web using this framework. Thus this paper serves the dual purpose of presenting the framework as well as a prototypical application of this framework. The rest of this paper is organized as follows: in Section 2 we explore some background in quality management, measurement ontologies and semantic web architectures. Section 3 presents the measurement ontology that we use. Section 4 presents the SWAP framework and the process of integrating databases into the framework. Section 5 describes experiments with the framework, in particular our testbed application using the presented measurement ontology. Finally we conclude in Section 6.

## 2 Background and Literature Review

Because of the length restriction, a full-length literature review is not included in this article. The primary background of this paper is in software development protocols, and not simply ontology mediation, so here we summarize some of the current efforts in software development protocols for the semantic web, and on the development of quality measurement ontologies.

Application protocols for semantic web is not a highly researched topic. The most important problem in this domain which is actively researched is metadata management. Shah and Seth [10] propose a model for managing metadata in a distributed environment. Interoperation across ontologies is also heavily researched and implemented (see e.g., [9]). We concentrate on a framework for appropriately and meaningfully distributing both data and meta-data in SWAP, thereby creating a full environment where distributed semantic web applications can be developed. OWL (Ontology Web Language) [11] is the culmination of W3C

and other researchers' efforts at developing a standardized ontology language for the semantic web. SWRL (Semantic Web Rule Language) [6] combines the frame-based approach to knowledge representation of OWL with the rule-based approach of RuleML (Rule Markup Language) [2] for the semantic web. Unfortunately, automatic inference engines explicitly for these ontology languages are not as well-developed as XML query engines [8], thus making the use of a hybrid approach such as SWAP pragmatic.

Though not specifically designed for the semantic web, there are ontologies that support day-to-day business decisions such those made for quality control. These ontology-based enterprise modeling projects are the Enterprise [12] and TOVE [3] projects. The Enterprise Ontology is comprised of ontologies of activity, time, organization, strategy, and marketing. A \building block" approach is taken in the TOVE project to construct ontologies of higher-level core concepts such as product, activity, state, causality, and time, resource collectively called the activity-state ontology [4]. A fundamental domain necessary to execute ontology-based web services is measurement, and a measurement ontology is built from the TOVE core ontologies. Though other measurement ontologies do exist (e.g., [5]), they are not developed to support enterprise activities as would be required for quality management web services.

**3 The TOVE Measurement Ontology**

For the purpose of our quality management case, we use the TOVE measurement ontology [7]. The TOVE measurement ontology is designed explicitly with quality control in mind, rather than only the basic process of measurement. A complete discussion of the ontology is out of the scope of this paper, here we only present some of the most important terms and axioms.

| | Expression | Description |
|---|---|---|
| Term-1 | $quality\_requirement(Qr)$ | $Qr$ is a quality requirement |
| Term-2 | $measured\_attribute(At)$ | $At$ is a measured attribute |
| Term-4 | $has\_sample\_sizing(At, Sz)$ | Measured Attribute $At$ has sample sizing plan $Sz$ |
| Term-8 | $has\_unit\_of\_measurement(At, U)$ | $At$ is measured using unit $U$ |
| Term-9 | $measuring\_resource(R)$ | Measurement performed by resource $R$ |
| Term-10 | $primitive\_measure(A)$ | $A$ is a primitive measure activity |
| Term-11 | $measure(A)$ | $A$ is primitive or collection of primitive measure activities |
| Term-12 | $inspect\_and\_test(A)$ | $A$ is an inspect and test activity |
| Term-13 | $measurement\_pt(Rt, At, Mp, Tp)$ | Attribute $At$ of a batch $Rt$ measured using measurement point $Mp$ at time point $Tp$ |
| Term-14 | $conformance\_pt(Q, Rt, At, Tp)$ | measurement of attribute $At$ of a batch $Rt$ taken at time $Tp$ shows that the batch conforms to the quality requirement $Q$. |
| Term-15 | $nonconformance\_pt(Q, Rt, At, Tp)$ | as above, does not conform |
| Term-16 | $conforming\_quality(X, Qr)$ | $X$ has a quality requirement $Qr$ |

Table 1: TOVE Measurement ontology - salient terms and axioms

TOVE Measurement Ontology terms are defined with propositions (or boolean terms) from the TOVE Core Ontologies. The TOVE measurement ontology consists of 19 core terms, 16 terms and 3 axioms. Table 1 shows some of the main terms and their descriptions.

**4 The SWAP Framework**

One of the most crucial parts of a semantic web application is the automation of the processing of ontologies. We now present an architecture that supports one way of processing ontologies in a semantic web application. This framework also has a three tier structure as shown in Figure 1.

1. The top tier is the client tier, consisting of clients or client agents, which are capable of sending requests to the next tier. Clients can be users interacting with a user interface, or automated intelligent software agents (ISAs). At this tier, clients pose queries using client ontologies and submit them to the next tier.
2. The next tier is the ontology processing tier. This tier uses the ontology, as well as any available mapping techniques to process the queries coming from the client tier. All rules and axioms are

available at this layer for processing. Facts are retrieved as needed by sending appropriate queries to the data layer. The retrieved facts can then be processed for the purpose of answering the client queries.

3. The data layer consists of all the facts included in the knowledge base. The ontology processing layer decides on which facts need to be retrieved, and sends appropriate queries to the data layer. The queries are processed at the data layer using any necessary mapping methods, and resulting facts are sent back to the ontology-processing layer.

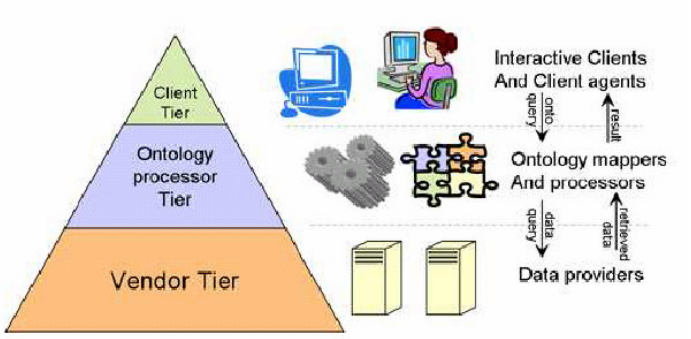


Figure 1: The SWAP pyramid showing the client, ontology and data layers, and the quality management

As an illustration of the above framework, let's consider a simple ontology for processing family trees. This sample ontology consists of a single class Person, having properties hasSex, and hasChild. Represented in a prolog-like format, a sample set of facts and rules in this ontology are shown in Figure 2:

```
Person('Joe').                  |
hasSex('Joe', 'male').          |
Person('Jill').                 |
hasSex('Jill', 'female').       |
hasChild('Joe', 'Mike').        | Father(X,Y) :- hasChild(X, Y), hasSex(X, 'male').
hasChild('Jill', 'Mike').       | Mother(X,Y) :- hasChild(X, Y), hasSex(X, 'female').
Person('Mike').                 | Spouse(X,Y) :- hasChild(X, Z), hasChild(Y, Z).
hasSex('Mike', 'male').         | ancestor(X,Y) :- hasChild(X,Y).
hasChild('Joe, 'Lucy').         | ancestor(X,Y) :- hasChild(X,Z), ancestor(Z, Y).
hasChild('Jill', 'Lucy').       | descendant(X,Y) :- ancestor(Y,X).
Person('Lucy').                 |
hasSex('Lucy', 'female').       |
Person('Tim').                  |
hasSex('Tim', 'male').          |
```

Figure 2: A simple Family Tree Facts and Rules

In our framework, the client will issue a query such as ancestor(X,'Mike'), and would expect a response from the ontology processor returning all possible substitution for the variable X. The ontology processor has all the rules, and the data tier has all the facts. During the processing of the rules at the ontology processor, whenever facts are needed, they are retrieved from the data tier. For example, in processing the above query, the system will need to send the following fact queries to the data layer: (i) hasChild(X,'Mike'), (ii) hasChild(X, 'Joe') and (iii) hasChild(X, 'Jill').

**4.1 Integrating Databases**

In the above discussion, we have not made any specific assumption about the data tier. Typically organizational data is stored in relational databases, and agents in this tier would need to translate the fact retrieval queries into SQL. This process is fairly trivial, since a fact retrieval can be translated into SQL by

simply placing constants in the query in the WHERE clause of the SQL statement. For example, in the above example, a fact retrieval such as hasChild(X,'Mike') translates to the SQL query SELECT * from hasChild WHERE col2='Mike' (assuming that the database has the hasChild stored in a table hasChild with columns col1 and col2).

Integrating databases into the data tier enables the use of database query optimization techniques to speed up the retrieval of the facts, which helps in the overall performance of the system in general. As shown in the measurement ontology case above, the use of multiple agents at one or more levels also increases the scalability of the system. Databases can be distributed over different agents and can be merged during the post retrieval process. All of these advances are possible because of the separation of the different tiers, enabling a form of data independence in semantic web applications.

**5 Experiments with the SWAP Framework**

We have implemented the SWAP Framework on several ontology-based applications, including test applications like the simple family tree ontology above, as well as a complex quality management web service using the measurement ontology discussed above. Here we describe our primary prototype case with a quality management web service.

**5.1 A Quality management web service using SWAP**

A prototype application for simulated quality mediation between organizations has been developed, completely using SWAP. The readers should note that the functionality of the mediation system was less critical than the applicability of SWAP in its development, and as a generalized semantic web application development protocol. In this section, we present a scenario that explains how the ontology and data queries flow between the different layers. In the prototype system, we implemented all the SWAP layers using agents implemented using J2EE web services, with two independent producer agents comprising the data tier, the customer agents at the client tier, and the QM agent is at the ontology tier.

Information flow between SWAP Layers First the customer agent sends the quality requirements for a receiving product to the Quality Management (QM) agent. The QM agent then classifies and stores these requirements along with other customers. The QM agent can then play the role of a third-party responsible for independent quality auditing, assurance, and control for the customer, automatically working with producer agents to ensure compliance to quality requirements. The following provides a detailed excerpt of this scenario.

1. The customer agent, org1, sends its quality requirements to the QM agent, qm0:
   *agent sends(org1,qm0,q requirement bundle from org1).*
   *Q requirement bundle from org1 is a pointer to a hierarchy of quality sub-requirements.*

2. *The QM agent represents a hierarchy of requirements in the following exemplar way:*
   *quality requirement(q requirement bundle from org1),*
   *has requirement(q requirement bundle from org1,qreq1),*
   *has requirement(q req1,q req1 1).*

3. If a requirement has no sub-requirements, e.g. q req1 1, then the QM agent translates the contents of the requirement in the following exemplar way
   *primitive requirement measures attribute(q req1 1,widget length),*
   *has standard value(widget length,15),*
   *has specification set(widget length,[14.5,15,5]),*
   *has unit of measurement(widget length,cm).*
   Standard value is akin to mean; specification set, tolerance specifications.

4. These requirements are sent to producer agents, and results of their quality control measurements are sent back to the QM agent:
   *measurement point(batch22,widget length, 14.8,10),* where 14.8cm is the value of the measurement and 10s the time of measurement.

5. Each measurement point is assessed by the QM agent as a conformance or nonconformance point, *e.g. conformance pt(q req1 1,batch22,widget length, 10)*. Reports of conformance are sent to the customer agent for immediate action or periodic reporting.

**6 Conclusion and Future Work**

As shown in the quality management web service, the framework can be easily augmented with agents to automate the process of exchange and retrieval. These experiments show the applicability of this framework as a generalized method for implementing semantic web applications, with or without major data retrieval tasks. We believe a generalizable framework for ontology and data-oriented semantic web applications is a basic necessity for efficient and organized development, and SWAP is an ideal step towards that direction. The SWAP layers can be extended and merged to fulfill most multi-tiered business applications using ontologies and web service integration. Detailed analysis of merging layers in SWAP for different business needs is part of the ongoing and future research. We intend to develop other testbed applications using SWAP, and run empirical studies to determine its effectiveness.

**References**

[1] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scienti¯c American*, May 2001.
[2] H. Boley, S. Tabet, and G. Wagner. Design rationale of RuleML: A markup language for semantic web rules. In *Proc. Int. Semantic Web Working Symposium*, 2001.
[3] M. Fox and M. GrÄuninger. Enterprise modeling. *AI Magazine*, Fall:109{21, 1998.
[4] M. GrÄuninger and M. Fox. An activity ontology for enterprise modeling. In *Proc. Workshop on Enabling Technologies*, Apr 1994.
[5] J. He°in. Measurement ontology (draft). Available - http://www.cs.umd.edu/projects/plus/SHOE/onts/measure1.0.html, (Accessed: October 30, 2001), 2000.
[6] I. Horrocks, P. Patel-Schneider, H. Boley, et al. SWRL: A semantic web rule language combining owl and ruleml. W3C Member Submission, 21 May 2004.
[7] H. Kim and M. Fox. Towards a data model for quality management web services: An ontology of measurement for enterprise modeling. *LNCS*, 2348:230{44, 2002.
[8] J. Lee and M. Sohn. The eXtensible rule markup language. *Comm. ACM*, 46(5):59{64, May 2003.
[9] E. Mena et al. Observer: An approache for query processing on global information systems. In *Proc. Intl. Conf. Cooperative Info. Systems*, pages 14{25, 1996.
[10] K. Shah and A. Sheth. Infoharness: Managing distributed, heterogeneous information. *IEEE Internet Computing*, pages 18{28, Nov-Dec 1999.
[11] M. Smith, C. Welty, and D. McGuinness. OWL web ontology language guide. W3C Recommendation - http://www.w3.org/TR/owl-guide/, 10 Feb 2004.
[12] M. Uschold, M. King, S. Moralee, et al. The enterprise ontology. *The Knowledge Engineering Review*, 13, 1998.