

# A Comparison of (Semantic) Markup Languages

Yolanda Gil and Varun Ratnakar  
USC Information Sciences Institute and Computer Science Department  
4676 Admiralty Way  
Marina del Rey, CA 90292  
varunr@isi.edu, gil@isi.edu

## Abstract

Several languages have been proposed as candidates for semantic markup. We needed to adopt a language for our current research on developing user-oriented tools operating over the Semantic Web. This paper presents the results of our analysis of three candidates that we considered: XML, RDF, and DAML+OIL along with their associated schemas and ontology specifications. The analysis focuses on the expressiveness of each language, and is presented along several dimensions and summarized in a comparison table. A surprising result of our analysis is the decision to adopt XML(Schema) for practical reasons, since it is able to accommodate a relatively expressive set of constructs and is widely known and commercially supported. We also discuss how we plan to complement XML(S) with a small set of conventions, so that we will have an easier transition to other markup languages in the future.

## Introduction

The compelling vision of the Semantic Web [Berners-Lee 97; Berners-Lee et al 01] has motivated a significant amount of research on semantic markup languages. From the W3C, from funding agencies in Europe and the US, and from various research communities, a number of proposals for the underlying representations of the Semantic Web have been put forward. It is likely that many of these languages will be adopted, extended, or dropped (except for some key ideas) by the wider community of developers and users of these languages much as we saw happen to network design and protocols and in the early days of the Internet. The contribution of this paper is our review of some of these languages as candidates to support our research on interactive tools to create and use distributed semantic models [Gil 02; Gil & Ratnakar 01].

Our approach to this analysis is heavily influenced by our background and experience in Knowledge Representation, specifically with description logics. However, we depart from others in that in our view, a semantic markup language should provide a vehicle to represent knowledge, and reasoning engines would import the representations perhaps translated to their own representation languages.

It is not clear that all knowledge represented in semantic markup languages should be importable to all reasoning engines. One might envision a mixed-initiative system that involves humans in reasoning or processing some of the information, and where man and machine would each consume a different (but related) subset of the knowledge. The major issue from our standpoint was the relative expressiveness of the language we adopted for our work.

We view the Semantic Web in part as a vehicle for the widespread use of knowledge-based systems [Gil 02]. Our work on TRELLIS aims to support decision making by capturing the rationale of a user's analysis of information sources and tradeoffs, and by making that rationale available to others through the use of a semantic markup language [Gil & Ratnakar 01]. Our work on PHOSPHORUS investigates ontology-based agent matching and communication in the context of an agentized human organization [Chalupsky et al 01], and where agents could use a Semantic Web to communicate with each other. In a recently approved project, we are planning to support knowledge-based reasoning for earthquake research in a distributed collaborative environment grounded on the Semantic Web and Grid computation. In all these projects, our desire is to use the Semantic Web as a vehicle to publish and share knowledge, while developing our own reasoners and inference mechanisms within the environments in which our systems will work. Although we would take advantage of inference capabilities available in a Semantic Web and developed by others (for example to query and locate resources), that is nevertheless not our main need.

For our analysis, we considered RDF Schema [Swick & Guha 00] and DAML+OIL [Horrocks et al 01], because they have been influenced by the knowledge representation community in their design. We also considered XML Schema [Thompson et al 01] because of the widespread adoption of XML and the many commercial tools that make it attractive for practical use. A few on-line notes describe tutorials on the Semantic Web are available that summarize the features of these languages and point out briefly their differences and complementary strengths [Boley et al 00; Dean 01]. Also related are comparisons among ontology representation languages such as [Corcho et al. 01].

The paper is organized as follows. We begin describing the features that we considered in our comparison. Then we present a table that summarizes the results of our analysis and comment in detail the entries on the table. Finally, we discuss our conclusions from this analysis and reflect on our choice of languages to support our research.

## Dimensions for Comparison

In this section, we give a list of dimensions that we thought were useful in comparing languages for the Semantic Web.

- **Context**: Modularity is an important consideration, especially in large distributed environments. The same term should be interpreted according to the context in which it is defined. The same term can be defined differently in alternative contexts. It is important that the language can express the different contexts in which terms should be interpreted.
- **Subclasses and properties**: These express relations between object classes. Subclasses represent "is-a" relations. Properties relate different object classes. Classes and properties are sometimes called concepts and roles, or frames and slots, objects and attributes, or collections and predicates.
- **Primitive data types**: A common set of primitive data types, such as strings and numbers, that can be used directly or to compose new complex types.
- **Instances**: These objects denote individuals, which can be described in terms of their properties or specified to be members of a class.
- **Property constraints**: These constraints define each property. The *domain* specifies the classes that can have that property. The *range* indicates the classes that can be used to assign values to the property. *Cardinality constraints* restrict the cardinality of the range, i.e., how many values can be assigned. Properties can be described *with respect to a specific class*, which means that the property can only be described or used for that class. Properties can also be described as *independent* entities, which means that any classes that satisfy the constraints can be related with that property even if this is not indicated in the class definition.
- **Property values**: In addition to range and cardinality, the values that can be assigned to a property can be restricted further. A *default* value can be provided. A *enumeration of possible values* may be given as a set of choices. *Ordered sets* specify the order of the elements, either extensionally or intensionally.
- **Negation and disjunction**: Negated statements of any description allowed in the language are often useful, but the computational cost is steep and as a result only limited forms of negation are typically supported in a given language. Disjunctive expressions are often

used to describe relations among subclasses or property constraints.

- **Inheritance**: Inheritance indicates that the constraints and values of properties in parent classes are true of the subclasses. Multiple inheritance allows inheritance from multiple parent classes.
- **Definitions**: Whether necessary and sufficient conditions for class membership can be specified. The system can use these definitions to reason about class-subclass relations (subsumption) and to determine whether instances are members of a class (recognition), as in description logic systems.

There are other dimensions with respect to expressiveness that could be considered, but these are the main ones that we were concerned about for our work.

## Analysis and Comparison

We analyzed and compared RDF Schema and DAML+OIL along the dimensions stated above. We also compared XSD, the XML Schema Definition Language, so we could understand better its features and limitations, even though it was not designed as a semantic markup language. Notice that the three languages are closely related in that XML is used to define RDF and RDF is used in turn to define DAML+OIL.

It is important to note that each language is in a different state of development. We used the XML Schema Definition Language [Thompson et al 01], which is now a recommendation of W3C and thus effectively a standard. We used the RDF Schema Specification 1.0, which has a Candidate Recommendation status in W3C and is still undergoing major modifications. We used the DAML+OIL version of May 2001, which is one of the initial proposals of a language that will continue to be extended and modified in the coming years.

We have not included in our analysis other alternative languages for semantic markup, such as topic maps and XQUERY, although we intend to do so in future work. A very active area of work is the development of languages to express and reason about services on the Semantic Web such as WSDL and DAML-S, which we have not included in the analysis presented here.

The analysis and comparison is presented in Table 1. The table is available on-line [Ratnakar and Gil 01], including detailed examples of descriptions in each language and pointers from every entry to the specific portion of the examples that illustrate the usage for that particular language. The rest of this section describes the results of our analysis and comparison in detail for each of the dimensions described above.

Dimension	Details	XML Schema	RDF Schema	DAML+OIL
Contexts	Contexts	Yes	Yes	Yes
Class based Model	Object Classes & Properties	No (A Class <i>could</i> be any element & its property <i>could</i> be its child element – NO DEFINED SEMANTICS)	Yes ( <code>rdfs:Class</code> & <code>rdfs:Property</code> )	Yes ( <code>daml:Class</code> , <code>daml:ObjectProperty</code> , <code>daml:DatatypeProperty</code> )
	Inheritance	No (Although we may extend element Types)	Yes (Properties and Classes) ( <code>rdfs:subClassOf</code> , <code>rdfs:subPropertyOf</code> )	Yes (Properties and Classes) Uses RDF syntax
Property /Element Constraints	Property /Element Range	Yes (Global & Local)	Yes (Global only – <code>rdfs:range</code> )	Yes (Global – <code>rdfs:range</code> & Local – <code>daml:Restriction</code> , <code>onProperty</code> , <code>toClass</code> )
	Property /Element Domain	Yes (implicitly, the element in which the ‘property element’ is defined)	Yes (Global only) <code>rdfs:domain</code>	Yes (Global) – <code>rdfs:domain</code>
	Property /Element Cardinality	Yes (Local only – <code>minOccurs</code> , <code>maxOccurs</code> )	No	Yes (Local – <code>minCardinality</code> , <code>maxCardinality</code> , <code>cardinality</code> & Global – <code>UniqueProperty</code> , or <code>Restriction</code> a subClass of “ <code>#Resource</code> ”)
Data Types & Instances	Basic Datatypes	Yes Variations of numerical, date and string datatypes.	No Only Literals	Yes Allows the use of XMLSchema Datatypes
	Enumeration of Property /Element Values	Yes <enumeration>	No	Yes Point to XMLSchema enumeration datatype
	Instances	Yes	Yes < <code>rdf:ID</code> >	Uses RDF syntax
Data Sets	Bounded Lists	No	No	Yes < <code>daml:collection</code> >
	Ordered Data Sets	Yes Data Sets maintain order by default	Yes < <code>rdf:Seq</code> >	Yes < <code>rdf:Seq</code> >
Negation & Disjunction	Negation	No	No	Yes < <code>daml:ComplementOf</code> >
	Disjunctive Classes	No (although we can have unions of element Types)	No	Yes < <code>daml:disjointUnionOf</code> >
Definition	Nec. & Suff. Cond.’s for Membership*	No	No	Yes < <code>daml:sameClassAs</code> > < <code>daml:UnambiguousProperty</code> >

Table 1. Comparison summary

## Contexts

Namespaces [Bray et al. 98], one of the most powerful features adopted by the XML standard, can be used as a limited form of context. This is done using the `<xmlns:*label*="URI">` tag, where `*label*` is the prefix used to refer to elements in the particular context specified by the URI. `<xmlns="URI">` specifies the *default* namespace when the prefix is omitted within the document. A very subtle but important point is that the URI specified in the tag is not required to point to a schema that defines the terms to be used in that namespace, and so effectively the tag just provides a label to refer to the context. To specify the location of a schema to be used to interpret the terms, the tag `<xsi:schemaLocation>` must be used. In summary, namespaces by themselves provide a very simple notion of context, and only the additional specification of a schema location provides the definitions of terms.

RDF uses XML namespaces to refer to schemas. An important difference from XML is that the namespace URI reference also identifies the location of the RDF schema. Thus, the use of namespaces in RDF seems to be a more clean mechanism to represent contexts.

DAML+OIL also uses XML namespaces. Unlike RDF, the schemas (called ontologies) specified in a namespace tag are not used. In order to use their definitions, they have to be explicitly imported using the `<daml:imports>` tag.

## Class based Model

In XML(S), there are no explicit constructs for defining classes and properties. There are only elements and subelements, which can be given an ad-hoc interpretation as a class/subclass or as a class/property statement. However, lack of a standardized way to represent classes and properties in XML results in ambiguity when mapping from the XML data model to a semantic model. XML allows the definition of new types by extending or restricting existing simple or complex element types. However, the extended and restricted types are not subclasses of the types used in their definition.

RDF Schema provides explicit tags to define new classes and properties. Classes can be specified with the `<rdfs:class>`. Subclasses and subproperties can be specified using `<rdfs:subClassOf>` and `<rdfs:subPropertyOf>` (the top class defined in the schema is `Resource`). When a class is a subclass of several superclasses, this is interpreted as the conjunction of the superclasses. Cycles in the class hierarchy are allowed. Class equivalence can be specified through cyclical descriptions. The RDF Schema definition may be revised and this could change in the near future.

DAML+OIL provides its own `<daml:class>` tag. The difference between the DAML class and RDF class is that DAML+OIL does not allow cycles in the class hierarchy and provides an explicit `<daml:sameClassAs>` tag. DAML+OIL classifies properties into Object Properties (that relate a class to another class) and Datatype Properties (that relate a class to a primitive data type).

## Data Types and Instances

XML Schema offers a wide range of data types, compatible with databases. DAML+OIL adopts the primitive data types of XML Schema, whereas RDF provides only literals (strings). Instances in DAML+OIL are specified (at least for now) using RDF syntax.

## Property Constraints

XML Schema provides range constraints on elements by the “type” attribute of the element definition. The domain of an element is implicitly the parent element within which it is defined or referred to. A property, and hence its range, is global if the property (or element) is a top-level element (no parent elements), otherwise it is local. Local cardinality constraints on properties can be specified using “minOccurs” and “maxOccurs” attributes while referring to the property inside a parent element. However, it is not possible to define cardinality constraints globally. Note that XML has elements and attributes, so the support for properties is not part of the XML Schema definition.

RDF Schema allows range and domain constraints on properties, which are declared globally. Multiple range statements, which were not allowed until recently, imply conjunction, as do multiple domain statements, i.e. all of the constraints have to be satisfied. RDF Schema does not provide cardinality constraints in its specification.

DAML+OIL specifies similar semantics for multiple range and domain statements as in RDF Schema. DAML+OIL however also allows for local range constraints using:

```
<rdfs:subClassOf><daml:Restriction>
  <daml:onProperty.../><daml:toClass.../>
</daml:Restriction></rdfs:subClassOf>
```

The `<daml:Restriction>` tag forms an anonymous class consisting of all instances that satisfy the restriction.

## Property Values

Elements are by default ordered in XML. However, we can impose a particular order on the occurrence of elements in XML Schema by the `<xsd:sequence>` tag.

In RDF, we can order a set by using the `<rdf:Seq>` tag. We can use the same in DAML+OIL.

Neither XML Schema nor RDF provides constructs for defining a bounded list. DAML+OIL has a tag `<daml:collection>` which specifies this.

## Negation and Disjunction

Only DAML+OIL supports some kind of negation and disjunction, through `<daml:complementOf>` (negation) and `<daml:disjointUnionOf>` (disjunction). XML Schema also provides a `<union>` tag which gives the disjoint union of various element “types”.

## Definitions

Necessary and sufficient conditions for class membership can be expressed with `<daml:sameClassAs>`. Another interesting tag is the `<daml:UnambiguousProperty>` tag which uniquely identifies the resource from the property (like a primary key).

## Discussion

It seems that DAML+OIL would be our best choice, given that its expressiveness is comparable with description logics, which we have been using in the past for our work. However, the language is still evolving, and important tools such as parsers (e.g. the DAML validator) are still limited and undergoing development. RDF(S), although not as expressive as DAML has its advantages in the simplicity of its design. But RDF(S) syntax gets in the way, and efforts such as Notation 3 [Berners-Lee 99] aim at providing an alternative syntax and it is not clear how they relate to the standardization efforts. Another problem with RDF(S) is that it remains in an evolutionary phase as a candidate recommendation for W3C and has not yet reached the status of a proposed recommendation. The problem with XML(S) is the lack of semantics. XML(S), on the other hand, emerges as a surprisingly expressive language. In addition, it is easy for people to understand so its use is widespread and well known. There are also a lot of commercial tools already available for it. For these reasons, we chose XML and XML(S) in our work for the near future.

In order to handle the shortcomings of using XML(S), we adopt a few conventions to supplement the standard XML(S) specification:

- **Conformity with the schema definitions.** There is no requirement in the XML specification that data should conform to schema definitions. Also, namespaces in XML are just unique identifiers and are not checked for the presence of valid schemas. We will require that any data handled by our systems conform to the schema definitions, indicated via namespace URI's, associated with the data.
- **Adoption of meta-tags to indicate semantic distinctions.** A few selected tag names will be adopted to indicate the semantic distinctions in the sub-elements. Thus, a tag such as "`<kw:properties>`" will indicate that its sub-elements were properties of the parent element.
- **Elimination of tags included within text.** It will not be possible to include tags within text, such as

"`<author>`Complete name not specified in the book`<name>`J.R. Rowlings`</name><author>`".

We propose these conventions to supplement the current XML specification in order to facilitate our expected transition to a semantic markup language in the future.

## Conclusions

We have discussed in detail how XML(S), RDF(S), and DAML+OIL compare across a selected number of features regarding their expressiveness. Our analysis leads us to choose XML(S) as the current best candidate to continue our research on tools for collaborative distributed environments within the Semantic Web. As new extensions, additions, and alternatives to these languages emerge, we should transition our efforts as appropriate. With this in mind, we have described some conventions that we have adopted and that we hope will ease our transition to other markup languages in the years to come.

## Acknowledgments

We would like to thank present and past members of the Interactive Knowledge Capture Project at USC/ISI for their comments and insights, including Geoffrey Plitt, Jihie Kim, Jim Blythe, Marcelo Tallis, and Larry Kite. Our analysis was improved thanks to the insights and comments of (alphabetically) Hart Lewis, Ian Horrocks, Ken Sall, Mark Butler, Ora Lassila, Oscar Corcho, Peter Patel-Schneider and Peter Crowther. We would like to acknowledge the support of the Air Force Office of Scientific Research (AFOSR) with award number F49690-00-1-0337 and the DARPA Control of Agent-Based Systems (CoABS) with award number F30602-97-1-0195.

## References

- Berners-Lee, T. 1997. *Weaving the Web*. Reading, Mass.: Addison-Wesley.
- Berners-Lee, T.; Hendler, J.; and Fensel, D. 2001. The Semantic Web. *Scientific American* 78(3):20–88.
- Boley, H., Decker, S., Sintex, M. <http://www.semanticweb.org/knowmarktutorial/>
- Bray, T; Paoli, J and McQueen, C.M., 1998. The Extensible Markup Language (XML) Specification, W3C Recommendation, <http://www.w3.org/TR/REC-xml.-19980210>
- Bray, T.; Hollander, D. and Layman, A. 1999. Namespaces in XML, W3C Recommendation, <http://www.w3.org/TR/REC-xml-names>.
- Dean, M. 2001. Language Feature Comparison, <http://www.daml.org/language/features.html>
- Gil, Y. and Ratnakar, V. 2001 "Trellis: An Interactive Tool for Capturing Information Analysis and Decision Making", Internal Project Report.
- Horrocks, I. et al. 2001. DAML+OIL Language Specification, <http://www.daml.org/2001/03/daml+oil-index>
- Ratnakar, V. and Gil, Y. 2001. "A Comparison of (Semantic ) Markup Languages". <http://trellis.semanticweb.org/expect/web/semanticweb/comparison.html>

- Swick, R and Guha R.V. 2000. The Resource Description Framework (RDF) Specification, W3C Recommendation, <http://www.w3.org/rdf>.
- Thompson, H. et al. 2001. XML Schema Part 1: Structures, W3C Recommendation, <http://www.w3.org/TR/xmlschema-1/>