# 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
gil@isi.edu, varunr@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. We are starting work on supporting 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.

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 with 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.

## 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, conjunction 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. XML is used as a syntax for RDF. RDF is not only used as a syntax for DAML+OIL, but also as a sublanguage since some expressions are written in RDF (e.g. instances).

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. 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 (2001) | RDF Schema (2000) | DAML+OIL (2001) |
|---|---|---|---|---|
| **Contexts** | **Contexts** | Yes | Yes | Yes |
| **Classes** | **Object Classes & Properties** | No (A Class *could* be any element & its property *could* be its child element – NO DEFINED SEMANTICS) | Yes (rdfs:Class & rdfs:Property) | Yes (daml:Class, daml:ObjectProperty, daml:DatatypeProperty) |
| | **Inheritance** | No (Although we may extend element Types) | Yes (Properties and Classes) (rdfs:subClassOf, rdfs:subPropertyOf) | Yes (Properties and Classes) Uses RDF syntax |
| **Property /Element Constraints** | **Property /Element Range** | Yes (Global & Local) | Yes (Global only – rdfs:range) | Yes (Global – rdfs:range & Local – daml:Restriction ,onProperty, toClass) |
| | **Property /Element Domain** | Yes (implicitly, the element in which the 'property element' is defined) | Yes (Global only) rdfs:domain | Yes (Global) – rdfs:domain |
| | **Property /Element Cardinality** | Yes (Local only – minOccurs, maxOccurs) | No | Yes (Local – minCardinality, maxCardinality, cardinality & Global – UniqueProperty, or Restriction a subClass of "#Resource") |
| **Data Types & Instances** | **Basic Datatypes** | Yes Variations of numerical, date and string datatypes. | No Only Literals (this version) | Yes Allows the use of XMLSchema Datatypes |
| | **Enumeration of Property /Element Values** | Yes <enumeration> | No | Yes <daml:oneOf> Can also point to XMLSchema enumeration datatype |
| | **Instances** | Yes | Yes <rdf:ID> | Uses RDF syntax |
| **Data Sets** | **Bounded Lists** | No | No | Yes <daml:collection> |
| | **Ordered Data Sets** | Yes Data Sets maintain order by default | Yes <rdf:Seq> | Yes <daml:list> |
| **Negation, Disjunction & Conjunction** | **Negation** | No | No | Yes <daml:ComplementOf> |
| | **Disjunctive Classes** | No (although we can have unions of element Types) | No | Yes <daml:disjointUnionOf> <daml:unionOf> |
| | **Conjunctive Classes** | No | Yes Multiple <rdf:subClassOf> | Yes <daml:intersectionOf> |
| **Definition** | **Nec. & Suff. Cond.'s for Membership*** | No (however, <unique> could be interpreted as an 'Unambigous Property') | No | Yes <daml:sameClassAs> <daml:UnambigousProperty> |
| **Types of Properties** | **Inverse** | No | No | Yes <daml:inverseOf> |
| | **Transitive** | No | No | Yes <daml:TransitiveProperty> |

**Table 1. Comparison summary**

## Contexts

Namespaces [Bray et al. 98], one of the most elegant 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.

## Classes

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 super-classes, this is interpreted as the conjunction of the super-classes. Cycles in the class hierarchy are not allowed, however class equivalence can be specified through cyclical descriptions (*Note: Future versions are expected to change this and allow cycles*). The RDF Schema definition may be revised and this could change in the near future.

DAML+OIL allows cycles in the class hierarchy. It allows definitions of subclasses using the <daml:subClassOf> 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), whereas RDFS makes no such distinction.

## 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 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, Conjunction 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". Conjunction is supported in DAML+OIL through the <daml:intersectionOf> tag, while multiple <rdfs:subClassOf> tags imply conjunction in RDF Schema.

### Definitions

Necessary and sufficient conditions for class membership can be expressed with <daml:sameClassAs>. Another interesting tag is the <daml:UnambigousProperty> tag which uniquely identifies the resource from the property (like a primary key). There is also a <unique> tag in XML Schema which is similar to the UnambigousProperty tag.

### Types of Properties

DAML+OIL supports different types of properties. A property can be defined in DAML+OIL as 'InverseOf' another property. A property can also be defined as a 'TransitiveProperty'.

## Discussion

An important difference among these languages is that XML(S) has no associated semantics, while RDF(S) and DAML+OIL do. Without semantics, there is no standard way to interpret the constructs in the language and develop reasoners that will reach the same inferences from a given set of expressions.

The languages also have different underlying object models. This has more implications with respect to mapping and translating expressions in the different languages. Research is under way to establish correspondences across models.

DAML+OIL has a number of useful features that are not included in XML and RDF Schema. It is based on description logics, which have been used in a variety of applications and domains. However, the first release of the language is still recent, and basic tools such as parsers are undergoing development. The widespread use of DAML+OIL will not likely happen without such tools. RDF(S) offers simpler constructs that may perhaps be more accessible to a wider range of users, and there are more advanced tools for it. But RDF(S) syntax is considered cumbersome by many, and efforts such as Notation 3 [Berners-Lee 99] aim at providing an alternative syntax while not being established as standards. 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.

An important practical consideration is the ease of use of XML. There are a variety of tools available that make XML very usable and accessible to a wide range of users. There is also a general widespread acceptance of XML among Web users at large, mostly as a data exchange syntax. An important reason is the emphasis on simplicity that guided the design of XML. We believe that ease of use is a very important feature of any technology underlying the Semantic Web.

It turns out that accessibility and ease of use is a prime concern in our work. In one of our projects, we face the challenge of enabling a sizeable community of scientists to develop detailed consensual models of geophysics. Current ontology editors and other interactive acquisition tools still have severe limitations, for example with respect to their ease of use and required training. Yet, the community is ready to start developing some initial representation of their community's models, and they need tools that are accessible to them. Another reason why accessibility is a prime concern is that it is, in part, a practical requirement imposed by the data sources available, which are developed by authoritative government and commercial sources that already use XML or are likely to use it in the near future. In order for us to do reasoning and problem solving with the data we need to be able to import or derive models that we can use in a knowledge representation system or a problem solver. We cannot in practice design a system that requires the development of an RDF or DAML+OIL model for each data source that it needs to use. We would expect that the developers of those data sources will be eager to upgrade and use a semantic markup language, as long as they perceive it as accessible to them as they know XML is.

XML(S) has no semantics. Yet, the language has a surprisingly useful number of features and constructs that can be exploited to build models. A viable option for the time being may be to use XML(S) with a suite of conventions that would help the developers of content (data sources, ontologies, models) express their models (or skeletal versions of them) in a way that would facilitate future mappings/translations of these models to semantic markup languages. A few examples of such conventions could be:

- ***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. For example a tag could be used to indicate that the sub-elements are to be interpreted as 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>".

Our aim is to design these conventions so that they facilitate the (expected) transition to a semantic markup language in the future, while supporting the development of content in the immediate 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. Although XML(S) has no semantics, it may help bootstrap the development of content and tools for the Semantic Web, because of its wider acceptability and ease of use compared to RDF(S) and DAML+OIL. New extensions and alternatives to these languages are likely to emerge. One could see in the near future a situation analogous to the days where network protocols were being developed, where many proposals were developed with different degrees of formality, complexity, and associated implementations. Standards and widespread use result from the exploration of different designs, implementations, and evaluations. With this in mind, a viable option to bootstrap content development for the Semantic Web may be to extend XML(S) with some conventions that facilitate translation and mapping to knowledge representation systems, and that will ease transition to semantic 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 Hart Lewis, Ian Horrocks, Ken Sall, Mark Butler, Ora Lassila, Oscar Corcho, Pat Hayes, 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.w3c.org/rdf.

Thompson, H. et al. 2001. XML Schema Part 1: Structures, W3C Recommendation, http://www.w3.org/TR/xmlschema-1/