Semantic Web Languages: RDF vs. SOAP Serialisation

Stefan Haustein
University of Dortmund,
Computer Science VIII,
D-44221 Dortmund, Germany
haustein@ls8.cs.uni-dortmund.de

ABSTRACT

Although RDF is considered *the* Semantic Web language, it may not be the only one. SOAP serialisation provides several advantages, especially if the Semantic Web is not just about providing meta data for existing web pages, but also about exchange of content that is machine-readable in the first place. This paper discusses some problems with the RDF syntax and data model. RDF is compared to SOAP, and some SOAP advantages like better integration with existing standards and systems, improved readability, and industry support are pointed out.

Keywords: SOAP Serialisation, RDF, RDFS, Object-Orientation

1. INTRODUCTION

What are the consequences if the term "Semantic Web" does not just mean HTML with some meta-data, but also content that is machine-readable in the first place, thus being suitable for applications like software agent communication? Although RDF is suitable for that purpose, its syntax and data model are clearly optimised for annotating existing documents with meta data, describing existing web resources using a machine readable format.

In contrast to HTML, the flexibility of XML allows storing all relevant (meta) data in a machine-readable format in the first place. With the increasing separation of content and layout into XML and XSLT files, and the dynamic generation of (X)HTML, the need for a separate meta model may decrease. Naturally, RDF is well suited for annotating the generated HTML with information like PICS, but is content annotation really all the Semantic Web is about?

This article discusses some serious issues concerning the RDF syntax and data model when used as a primary machine readable content format, instead of just adding meta-data to existing HTML or XML pages. It presents SOAP serialisation as an alternative. In contrast to [16], the suggestion is not to use SOAP just as a syntax encoding of the RDF data model but to build a part of the Semantic Web on SOAP serialisation in the first place.

Semantic Web Workshop 2001 Hongkong, China

2. RDF EVOLUTION

The *Resource Description Framework* (RDF) [15] was originally designed by the World Wide Web Consortium (W3C) as a meta-language for annotating existing web pages with additional machine-readable information. A typical RDF application is the Platform for Internet Content Selection (PICS) [6], that is intended to annotate existing web resources with meta-data about e.g. the suitability of the content for children.

Although RDF was originally designed for annotation of existing web pages only, it is currently widely considered as machine-readable format for the Semantic Web and the DARPA Agent Meta Language (DAML).

However, when using RDF as the primary information format, and not just for meta data annotation, some issues arise. These problems are described in detail in the following sections. For an overview of additional general RDF issues and inconsistencies, the reader is referred to [4] and [10].

3. RDF SYNTAX ISSUES

The requirement that RDF should be usable for annotating existing pages with meta-data without breaking browser compatibility for the actual content resulted in a syntax that is more complicated than necessary for plain RDF documents. Also, several alternative syntax forms exist. All alternatives are semantically equivalent, but have different effects on the rendering process in the browser that is used to view the document.

The general RDF syntax consists of simple resource descriptions ("properties") embedded in a "description" element. In order to illustrate RDF and SOAP syntax alternatives, a FIPA 2000 Agent Platform description [14] is used as a common example here, where the serialised instances are taken from the Paris Agentcities node ApDescription[1]. Although the Paris node Agent Platform Description consists of four small objects only, the corresponding RDF serialisation (figure 1) becomes rather verbose.

In the RDF example, the `rdf:Description` elements are already replaced by their abbreviated form for improved

[1]see http://www.agentcities.org/Cities/paris_city.html, the dashes are replaced by "Camel" syntax for compatibility

```
<ApDescription id="1">
 <name>paris.agentcities.org</name>
 <dynamic>true</dynamic>
 <mobility>true</mobility>
 <transportProfile>

  <ApTransportDescription id="2">
   <availableMtps>
    <rdf:Bag>
     <rdf:li>

      <MtpDescription id="3">
       <mtpName>fipa.mts.mtp.iiop.std</mtpName>
       <addresses>
        <rdf:Bag>
 <rdf:li>iiop://leap.crm-paris.com:9000/paris.agentcities.org/acc</rdf:li>
 <rdf:li>iiopname://leap.crm-paris.com:9000/paris.agentcities.org/acc</rdf:li>
        </rdf:Bag>
       </addresses>
      </MtpDescription>

     </rdf:li>
     <rdf:li>

      <MtpDescription id="4">
       <mtpName>fipa.mts.mtp.http.std</mtpName>
       <addresses>
        <rdf:Bag>
         <rdf:li>http://leap.crm-paris.com:8080/acc</rdf:li>
        </rdf:Bag>
       </addresses>
      </MtpDescription>

     </rdf:li>
    </rdf:Bag>
   </availableMtps>
  </ApTransportDescription>

 </transportProfile>
</ApDescription>
```

Figure 1: RDF Syntax Example

readability. The example encoding is not the only RDF encoding option, though. RDF allows several syntax variants:

Resource description and type abbreviation: An `rdf:Description` element may be replaced by an element named like the type of the resource described, also obsoleting a corresponding `rdf:type` element. In the example, the abbreviated form was already used. All object descriptions in the example could be replaced by the corresponding standard form. For example,

```
<ApDescription id="1">
  ...
</ApDescription>
```

is equivalent to

```
<rdf:Description>
  <type resource="&fipaNS;#ApDescription" />
  ...
</rdf:Description>
```

Obviously, the second variant adds five extra elements to the example code.

Using attributes instead of elements: RDF elements may be replaced by attributes if they occur only once in their parent element, and contain only literal text without further substructures. For example, some of the `ApDescription` sub-elements could be replaced by attributes:

```
<ApDescription id="1">
  <name>paris.agentcities.org</name>
  <dynamic>true</dynamic>
  <mobility>true</mobility>
  ...
</ApDescription>
```

is equivalent to

```
<ApDescription id="1"
  name="paris.agentcities.org"
  dynamic="true" mobility="true">
  ...
</ApDescription>
```

Nesting instead of linking Instead of referring to an object using the `rdf:resource` attribute, the corresponding object can be embedded into the predicate element. In the original example, all objects are embedded for better readability.

```
<ApDescription id="1">
 <name>paris.agentcities.org</name>
 <dynamic>true</dynamic>
 <mobility>true</mobility>
 <transportProfile>
  <ApTransportDescription id="2">
       ...
  </ApTransportDescription>
 </transportProfile>
</ApDescription>
```

is equivalent to

```
<ApDescription id="1">
 <name>paris.agentcities.org</name>
 <dynamic>true</dynamic>
 <mobility>true</mobility>
 <transportProfile resource="#2" />
</ApDescription>

<ApTransportDescription id="2">
       ...
</ApTransportDescription>
```

The various RDF syntax options lead to two main problems: XSLT (and XML Schema) compatibility problems and problems with human readability.

3.1 XSLT Compatibility

The Extensible Stylesheet Language Transformations (XSLT) were designed with the main goal of separating the content and layout of Web pages. The basic idea is to design the original page using an XML language. The XML content is then converted to a "regular" (X)HTML page by an XSLT template.

The various RDF encoding options described above make development of XSLT templates for RDF difficult: In order to be fully applicable to RDF, XSLT templates would need to define a mapping covering all possible syntax alternatives. It would certainly be possible to design relatively simple XSLT templates for one concrete serialised form of RDF. But then the XSLT transformation would become either very fragile, or another processing step converting any RDF file to the form expected by the template would be necessary.

3.2 Human Readability

Another problem with RDF is human-readability. While one of the original ideas of XML is to provide some kind of compromise between machine and human-readability, RDF is actually difficult to read for humans. Again, the main reason are the meta language roots of RDF. With the various syntax options, it is even quite difficult to just see if two RDF documents are semantically equivalent. In order to read RDF documents, a human must be familiar with all syntax variants of RDF. When RDF is used to annotate an existing HTML page, the situation becomes even worse since it is often difficult to differentiate between RDF annotation and actual content. In addition, the verbosity of RDF makes it difficult to read when compared to other XML languages or SOAP.

4. RDF DATA MODEL ISSUES

While it seems relatively simple to fix the problems concerning the RDF syntax, this is far more difficult for the RDF data model.

The RDF data model is very simple. It is basically a labelled graph consisting of `(subject predicate object)` triples. With the RDF Schema language (RDFS [5]), the data model becomes significantly more structured. RDFS introduces a type system that can be used to express property constraints. Figure 2 shows the RDFS diagram corresponding to the RDF example. An corresponding *Unified Modelling Language* (UML) [17] diagram of the ontology is shown in figure 3.

To some extent, RDFS is similar to object oriented structures, except that properties must have globally unique names. The RDFS specification claims that the property centric approach makes it "very easy for anyone to say anything they want about existing resources, which is one of the architectural principles of the Web". However, deviating from "standard" object orientation also raises some interoperability issues with existing system or modelling tools.

4.1 Compatibility to Object Oriented Systems

Unfortunately, treating properties as first class members of the data model makes it impossible to map existing object hierarchies or database systems to RDFS automatically, without additional handling of property names to ensure global uniqueness. This is not just a problem with object-oriented systems or relational databases, also knowledge systems like Ontobroker [11] or Protégé [12] are seriously affected. When properties have a global domain and range definition, it is not possible to refine the definition in a subclass. It is also not possible for different classes to use the same property name with different value and domain restrictions [19].

In order to work around this problem, different mappings already exist, all having their own advantages and disadvantages, and it is quite simple to invent new ones. Possibilities are:

Facets: Facets were added to RDF by Stefan Decker to simplify RDF compatibility of the Protégé system [18]. The idea behind facets is to allow multiple ranges and refinements for properties. The mapping problem for other OO systems could be solved by introducing conflicting properties at a common base class and introducing the actual restrictions later where needed.

Name Concatenation: Stephen Cranefield designed an XSLT template mapping the property names to globally unique names by just concatenating them with the class names [8]. Applied to the UML diagram representing the course sample schema shown in Figure 3, the generated RDF Schema would be identical to Figure 2, except that all property names were concatenated the corresponding domain name. For example, `name` would be renamed to `ApDescription.name`.

Other options: It is quite simple to invent other mechanisms to ensure globally unique property names. For example, a dedicated XML namespace could be assigned to each object, preserving the original name but requiring extensive usage of XML namespaces.

The main problem is that there is no intuitive mapping. All mappings have their own advantages and disadvantages, without one being clearly preferable to the others. Moreover, except from the facets solution, which has the disadvantage of extending RDFS, it is not possible to apply the inverse mapping to any RDFS schema without preconditions. The inverse mapping is only possible if the RDFS is already of the right "form". It is not possible to generate compatible RDF among different mappings by using the output from one mapping as input for another (inverse) mapping. So even if all mappings are using RDFS as their target format, that does not help for interoperability at all.

Furthermore, this problem does not only affect connecting existing systems to the Semantic Web, but also ontology design using UML [9]. When using UML in the ontology design process, it becomes necessary to take special care of property names again.

Concatenating a property name with the domain name or a namespaces may also create problems for derived classes inheriting that property. All derived classes are a valid domain for the property, too, but one would need to remember the domain where the property was defined for constructing the right name and thus being able to access the property.

Another significant difference between the RDFS data model and standard object oriented systems is that a resource can have more than one type. For example, the Protégé system is not able to handle this without workaround. Protégé was designed to allow only one class for each instance because of user interface considerations. Protégé solves the problem by internally creating artificial concepts that are merged from the different types of a resource.

Please note that two different descriptions of one object can exist without requiring that an object is allowed to have several types. The described object and the descriptions just need to be separate objects.

4.2 Statements about Statements

The RDF data model is a set of `(subject predicate object)` statements, where the statements themselves do not have an address. In order to be able to make statements about statements, it is necessary to model the original statement as a resource having a subject, a predicate, an object, and a type.

5. SOAP

A potential alternative to RDF may be contained in the *Simple Object Access Protocol* (SOAP) [3] specification. SOAP is a specification covering remote procedure calls over HTTP. It contains an object serialisation format that can be compared to the Resource Description Format (RDF) to some extent, even if RDF is not just an object serialisation format. Although RDF was already existing when SOAP was being specified, RDF was not chosen as the default serialisation format for SOAP. Instead, SOAP introduces a completely new format defined from scratch.

SOAP is supported by computer industry leaders like Microsoft, IBM and SUN. The simplicity of SOAP together with the support from the industry suggests that many SOAP-based services will be available in the near future. While industry support is usually not really relevant for research, research in Artificial Intelligence may take significant advantage from the amount of structured data provided by

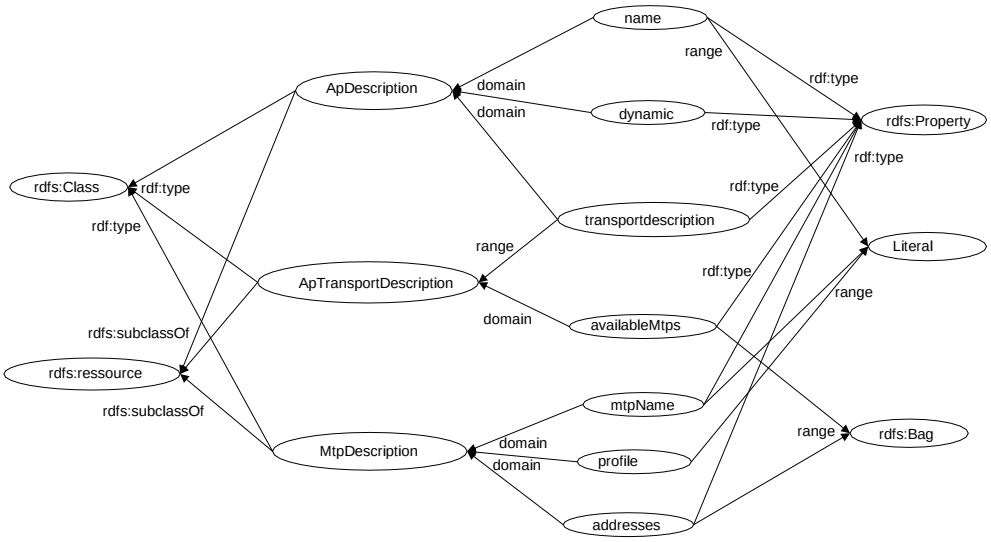

Figure 2: RDF Schema of the FIPA AP description

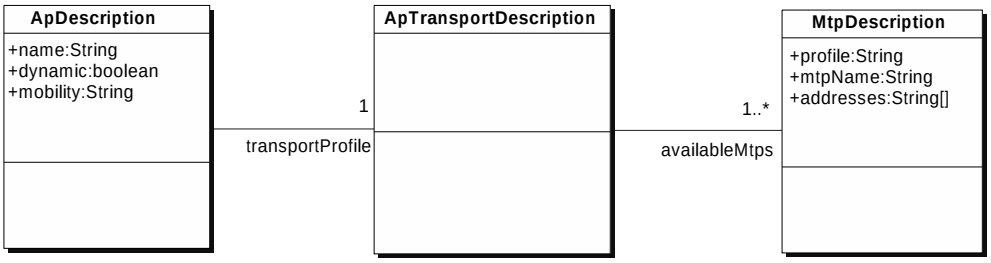

Figure 3: UML Diagram of the FIPA Agent Platform Description

a Semantic Web that is widely accepted and used. SOAP Implementations are available for a wide range of programming languages (C++, Java, Perl, Python)[2]. In contrast to HTML, SOAP is defined for direct communication between machines over the Internet, so it may affect the Semantic Web in many ways or even become a significant part of it. SOAP is suitable for all kinds of automated Internet services like weather forecasts, traffic services, or logistics coordination. Although SOAP supports alternative content formats, it is likely that most of the content will actually be encoded using SOAP serialisation. Thus, some questions arise: How can SOAP be integrated into the Semantic Web? Can the Semantic Web profit from SOAP services?

In his WWW9 presentation, Henrik Frystyk Nielsen ([16]) demonstrated how RDF can be encoded utilising the SOAP serialisation syntax. While he states in his presentation that SOAP is not meant to replace RDF, this is clearly one of the possible future scenarios in the areas where "Semantic Web" means exchange of information that is machine readable in the first place.

5.1 SOAP and CORBA

Although SOAP was originally designed as a remote method invocation protocol running over the Internet, SOAP is not just another *Common Object Request Broker Architecture* (CORBA). SOAP differs from CORBA in significant points:

Human Readability: In contrast to the CORBA Internet Inter ORB Protocol (IIOP), SOAP is not a binary format but an XML-based format that is human-readable. Even if SOAP is mainly intended to be read by machines, human readability is very helpful for debugging purposes and quick implementation.

Simple Installation: While CORBA requires huge software packages and does not provide a commonly accepted bootstrapping mechanism, SOAP is based on HTTP and can be implemented with little effort on top of existing libraries for XML and HTTP.

Even if SOAP still lacks a reasonable security model, it has the potential to become the connecting point between Java, Perl and Microsoft's .NET architecture by just offering a suitable feature set, while still being simple enough to be implemented by a broad range of programmers.

5.2 SOAP Syntax

Probably the main reason for SOAP becoming quite popular in the very short time it is available now is its simplicity. The serialised format of the FIPA example encoded in SOAP is shown in Figure 4. The format is much more similar to an XML special purpose format designed "by hand" than the RDF serialisation.

The main difference visible on first glance is the reduced nesting level of XML elements. In RDF, both objects and properties have their own tags. In SOAP, the nested tags starting a new object are always merged with the property tags if possible. This also has the advantage that navigating through a SOAP serialised document using path expressions becomes very similar to usual path expressions in object oriented programming or query languages. If the type of a property is not fixed, it can be resolved by adding a type attribute (e.g. `<transportProfile xsi:type="ApTransportDescription">`). Moving the type information into an attribute maintains the advantage of the reduced nesting level when compared to RDF.

Similar to RDF, SOAP allows alternative syntax forms for embedded and referenced objects, but in contrast to RDF the specification contains clear rules when an object is embedded and when it is referenced: Objects may be embedded if there exists only one referenced to them, otherwise they are linked. And in contrast to RDF, SOAP serialisation does not allow additional abbreviated or alternative XML syntaxes. Actually, there is no need for an abbreviated syntax since SOAP serialisation is compact enough in the first place.

The complete specification of SOAP serialisation syntax is given in [3].

5.3 SOAP Data Model

SOAP is based on a simple object oriented data model. The SOAP Data Model consists of structured objects having properties and a type. Thus, the basic building blocks are more complex than plain RDF. However, when compared to RDFS, there is no significant difference. Larger basic building blocks may have advantages e.g. when tracking the source of statements: The source information would be attached to just one object instead of needing reification for a lot of RDF statements. Also, when constructing URLs from OIDs, it becomes very simple to make statements about statements, again avoiding explicit reification that would be required in RDF.

In contrast to RDF, SOAP does not come along with its own schema language. Instead, XML Schema is used for validation of the syntactical correctness of SOAP serialised objects. While XML Schema does not seem the appropriate level of ontology modelling, SOAP serialisation fits well into UML modelling without the property naming problems of RDF. And when comparing the RDFS and UML diagrams (figure 3 and figure 2), UML seems significantly more appropriate for modelling ontologies.

6. IS SOAP SUITABLE FOR THE SEMANTIC WEB?

As shown in the preceding sections, SOAP has advantages over RDF in several areas. But does that mean that SOAP is sufficient to build a "Semantic Web"? The Semantic Web is meant to be more than just turning some existing object-oriented systems into SOAP services. How can a collection of SOAP services evolve into a Semantic Web? Are there any RDF core features missing in SOAP? Can the advantages of SOAP and RDF be combined to accelerate or simplify the process of building a Semantic Web?

6.1 Using SOAP Syntax for the RDF Data Model

Henrik Frystyk Nielsen [16] suggests to apply SOAP serialisation to the RDF data model to simplify integration of the SOAP RMI protocol with RDF data. Since this would mean yet another alternative syntax, the RDF syntax situation would not be simplified, except if the new syntax became the only one. But limiting RDF syntax variants

[2]http://www.superopendirectory.com/directory/4/standards/~23~/implementations

```
<ApDescription>
  <name>paris.agentcities.org</name>
  <dynamic>true</dynamic>
  <mobility>true</mobility>

  <transportProfile>
    <availableMtps>

      <MtpDescription>
        <mtpName>fipa.mts.mtp.iiop.std</mtpName>
        <addresses>
          <url>iiop://leap.crm-paris.com:9000/paris.agentcities.org/acc</url>
          <url>iiopname://leap.crm-paris.com:9000/paris.agentcities.org/acc</url>
        </addresses>
      </MtpDescription>

      <MtpDescription>
         <mtpName>fipa.mts.mtp.http.std</mtpName>
         <addresses>
           <url>http://leap.crm-paris.com:8080/acc</url>
         </addresses>
      </MtpDescription>

    </availableMtps>
  </transportProfile>
</ApDescription>
```

Figure 4: SOAP Syntax Example

in general would generate problems when using RDF for its original purpose of annotating existing documents. Also, the proposed syntax would not help for a seamless integration of other SOAP-based services since the property mapping problem would persist.

6.2 Saying Anything about Anything

In [2], Tim Berners-Lee justifies the usage of the property-centric data model instead of an "usual" object-oriented data model by claiming that object-oriented systems generally assume that information about an object is stored inside that object. So, in order to be able to say anything about anything, it becomes necessary to store the properties apart from the object.

While for object oriented systems it is true that property information is stored with the objects, this does not mean that objects cannot hold information about other objects. So there is no real requirement that e.g. the PICS rating of an HTML page is a property of that page. It could also be a separate object holding a pointer to the original page. A more complex example could be a car being the primary object and its description stored at an insurance company.

Even if a "standard" object-oriented data model is suitable for the Semantic Web, there is still a problem with SOAP. The SOAP specification does not specify how to assign URLs to objects. There is also no general mechanism to say "something" about a given URL. However, it seems relatively simple to assign URLs to objects built from the service address and a local unique number (OID). If an object is intended to describe a resource, it could have an "about" property like in the SOAP-encoded RDF syntax proposed by Henrik Frystyk Nielsen [16].

6.3 Schema Language

The SOAP specification does not contain any schema language, but refers to XML Schema for syntax validation. However, a syntax specification language like XML Schema is not really suitable for modelling ontological elements like classes, attributes and associations. Actually, the SOAP specification just describes how to get an XML Schema from the data structure and not vice versa.

But even if SOAP does not come along with its own schema language, the SOAP data model fits quite well into UML. Thus, UML tools can be used for modelling, without the limitations shown for the RDF case.

Furthermore, even if SOAP describes instance serialisation only, the UML meta model [17] can still be utilised to serialise UML models using SOAP serialisation syntax. Some UML constructs would require the definition of a concrete mapping, but there is no general incompatibility in the data model. Distinction between different types of associations in UML could also be utilised to define the cases where embedded objects are allowed in SOAP serialisation more clearly.

6.4 Integration of existing Services and Systems

In his XML2000 keynote, Tim Berners-Lee suggested to use "screen-scraper"-like XSLT templates to convert XML into RDF for the starting century of the a Semantic Web [1], like shown in figure 5. However, the SOAP serialisation syntax is simple enough to be widely accepted as a general XML modelling convention. So it may be possible to eliminate the need for an additional "screen scraping" template in many cases. Also, the compatibility with XSLT allows building HTML or WML transformation templates, just like

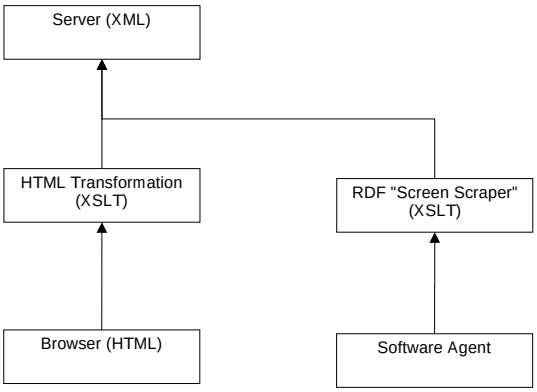

Figure 5: RDF screen scraping

	RDF	SOAP
Serialisation	RDF	SOAP
HTML/WAP generation	?	XSLT
Query Language	?	OQL
Syntax Validation	?	XML Schema
Schema Serialisation	RDFS	UML Meta-model
Schema Modelling	?	UML

Table 1: Integration with existing standards

for XML languages designed "by hand".

Moreover, since SOAP fits nicely into existing object oriented and relational database systems, it may simplify the schema generation process. Instead of needing to design an XML DTD or XML Schema, it would be sufficient to agree on an UML diagram, allowing a much more appropriate level of abstraction.

Another SOAP advantage is that SOAP quite nicely integrates with a lot of existing standards (Table 1). This was already discussed for XSLT and UML, but we also have a well designed query language for object oriented systems. The Object Query Language (OQL) [7], designed by the Object Data Management Group (ODMG) fits perfectly with SOAP and preserves a high degree of SQL compatibility.

7. WHAT IS MISSING

For a real "Semantic Web", the upper logical levels are missing from SOAP. But this holds for RDF as well, and looking at existing inference systems, it does not seem more difficult to build them for SOAP than for RDF [13].

Also, mechanisms for schema translations may become necessary. For example, if a legacy system with a direct SOAP mapping needs to be adapted to a SOAP based standard, or if SOAP based standards for different areas need to interoperate. However, this will probably be more simple than general XML-XML translations, where XML-XML translations are possible today using XSLT. Again, the same would be necessary for RDF.

8. CONCLUSION

Obviously, we have two formats and data structures that are similar to some extent, and both are relevant for the "Semantic Web", especially when the term means something different than annotation of existing pages with additional meta data. The main problem is that both approaches are difficult to translate into each other.

In contrast to SOAP, RDF is suitable for "in place" annotation of existing web pages. For being able to say "anything about anything", RDF uses URLs as global unique identifiers.

Although SOAP is not suitable for "in place" annotation and would require an extension assigning URLs to objects in order to become really suitable for the Semantic Web, it has several advantages in other areas. SOAP fits quite well into other standards like UML, XSLT and OQL. It provides a clear and simple syntax. The SOAP Serialisation syntax nearly reaches the quality of a "hand-made" one. Thus, SOAP serialisation is suitable as content encoding convention in the first place. So, in contrast to RDF, an extra conversion step can be avoided completely.

The logical level is not yet covered for both approaches, RDF and SOAP.

For the future, it seems to make sense to work on both approaches without dooming the other. Even if looking at SOAP may split the Semantic Web community to some extend, competition may also set free a lot of development energy. Perhaps the logical layer to be built on top of RDF and SOAP may allow for an integration of both approaches at some future point of time.

9. REFERENCES

[1] Tim Barners-Lee. RDF and the Semantic Web. In *XML 2000*. GCA, 2000.

[2] Tim Berners-Lee. What the smantic web can represent, September 1998. http://www.w3.org/DesignIssues/RDFnot.html.

[3] Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn, Henrik Frystyk Nielsen, Satish Thatte, and Dave Winer. Simple Object Access Protocol (soap) 1.1. Note, World Wide Web Consortium, 2000. http://www.w3.org/TR/2000/NOTE-SOAP-20000508.

[4] Dan Brickley. RDF interest group - issue tracking. Technical report, World Wide Web Consortium, 2000. http://www.w3.org/2000/03/rdf-tracking/.

[5] Dan Brickley and R. V. Guha. Ressource Description Framework (RDF) Schema specification 1.0. Technical report, World Wide Web Consortium, 2000. http://www.w3.org/TR/CR-rdf-schema-20000327.

[6] Dan Brickley and Ralph R. Swick. PICS rating vocabularies in XML/RDF. Technical report, World Wide Web Consortium, 2000. http://www.w3.org/TR/2000/NOTE-rdf-pics-20000327.

[7] R. G. G. Cattell, editor. *The Object Database Standard: ODMG 2.0*. Morgan Kaufmann, 1997.

[8] S. Cranefield. Networked knowledge representation and exchange using UML and RDF. *Journal of Digital Information*, 1(8), 2001. http://jodi.ecs.soton.ac.uk/.

[9] S. Cranefield and M. Purvis. Uml as an ontology modelling language. In *Proceedings of the Workshop on Intelligent Information Integration, 16th International Joint Conference on Artificial Intelligence (IJCAI-99*, 1999.

[10] Stefan Decker. Proposed updates of RDF, 1999. http://www-db.stanford.edu/~stefan/updates.html.

[11] Stefan Decker, Michael Erdmann, Dieter Fensel, and Rudi Studer. Ontobroker: Ontology based access to distributed and semi-structured information. In R. Meersman and other, editors, *Semantic Issues in Multimedia Systems, Kluwer Academic Publisher, Boston, 1999*. Kluwer Academic Publisher, Boston, 1999.

[12] H. Eriksson, R. W. Fergerson, Y. Shahar, and M. A. Musen. Automatic generation of ontology editors. In *Twelfth Banff Knowledge Acquisition for Knowledge-based systems Workshop*, Banff, Alberta, Canada, 1999.

[13] A. S. Evans. Reasoning with UML class diagrams. In *Proceedings of the Workshop on Industrial Strength Formal Methods (WIFT'98)*. IEEE Press, 1998. http://www.cs.york.ac.uk/puml/papers/evanswift.pdf.

[14] Foundation For Intelligent Physical Agents (FIPA). *FIPA Agent Management Specification*, 2000. http://www.fipa.org/specs/fipa00023/XC00023F.pdf.

[15] Ora Lassila and Ralph R. Swick. Ressource Description Framework (RDF) model and syntax specification. Technical report, World Wide Web Consortium, 1999. http://www.w3.org/TR/1999/REC-RDF-SYNTAX-19990222.

[16] Henrik Frystyk Nielsen. Soap, RDF and the Semantic Web. In *WWW9*, 2000.

[17] Object Management Group. *OMG Unified Modeling Language Specification*, June 1999. Version 1.3.

[18] Stanford University. *Using Protégé-2000 to Edit RDF*, June 2000. http://www.smi.stanford.edu/projects/protege/protege-rdf/protege-rdf-20000629.html.

[19] Frank van Harmelen and Dieter Fensel. Practical knowledge representation for the web. In *IJCAI'99 Workshop on Intelligent Information Integration*, 1999.