Towards a Modularized Semantic Web

Raphael Volz, Daniel Oberle
Institute AIFB,
University of Karlsruhe,
76128 Karlsruhe, Germany

Alexander Maedche
FZI Forschungszentrum Informatik
University of Karlsruhe,
76131 Karlsruhe, Germany

ABSTRACT

Modularization is an established principle in software engineering. It has also been considered as a core principle when developing the World Wide Web. Along the same lines, the Semantic Web has to be based on modularization principles. This paper provides a first step into a modularized Semantic Web. It provides an elaborated and carefully evaluated view on existing technologies for naming, referring and modularization in the Web. Based on this analysis we propose means to import and include (parts) of RDF models by extending the RDF(S) meta-model, introducing new primitives for modularity.

1. INTRODUCTION

One general principle of powerful software systems is that they are built of many elements. Thus, when designing a system, the features of a system should be broken into relatively loosely bound groups of relatively closely bound features. Power comes from the interplay between the different elements. This interplay results in essential interdependencies and increases the ability to reuse and modify. Hence, future changes and consecutive testing can be limited to the relevant module. This will allow other people to independently change other parts at the same time. Modular design hinges on the simplicity and abstract nature of the interface definition between the modules. Notably, modularity was one of the core design goals for the World Wide Web.[1]. Along the same lines, the Semantic Web will not consist of neat ontologies that expert AI researchers have carefully constructed. Instead of a few large, complex, consistent ontologies that great numbers of users share, one will see a great number of small ontological elements consisting largely of pointers to each other [1].

We agree with this view and carefully evaluate existing technologies for naming, referring and modularization in the Web. We show that these technologies do not suffice for the task of building a truly modular Semantic Web. Based on our analysis we propose means to support this vision by extending the RDF meta-model, introducing new primitives for modularity. Namely, means for import and inclusion of (parts) of other RDF models. Additionally, we present an architectural setting for tools implementing this kind of modularization and finally give a set of engineering guidelines for building modularized Semantic Web applications on the basis of these novel means.

The structure follows the outlined procedure, thus section 2 provides an elaborated overview and evaluation of technologies relevant for modularization. Section 3 presents requirements for a modular Semantic Web and crafts an extension of the RDF meta-model to reflect these requirements. Section 4 provides reference applications using modularized ontologies and the aforementioned engineering guidelines for modularized Semantic Web applications from an ontology building perspective. Before we conclude and recapitulate our contribution in section 6, we give a short survey on related work in section 5.

[1] see http://www.w3.org/DesignIssues/Principles.html

Semantic Web Workshop 2002 Hawaii, UAS

2. TECHNOLOGIES

This section provides an elaborated overview and evaluation of technologies providing modularization technology for the Web in general. Our overview is roughly separated into referring technologies and modularization technologies defined for XML.

2.1 Referring technologies

Links between Web resources are commonly called arches. Using or following a link for any purpose is called traversal. The traversal's origin is called the starting resource and the destination is the ending resource.

HTML Arches. present the simplest and oldest mechanism for referring in the Web. They provide simple outbound links between different resources which can be named for display purposes. Links to points inside other HTML documents are implemented using URI fragments.

XLink. [3] presents the referring technology adopted for XML and extends HTML's possibilities tremendously. Xlink allows to specify binary relations as found in RDF (see figure 1).

Additionally, it allows to link sets of elements with another by using arcs. Each set is qualified using a string label, all set elements use xlink:label with the same attribute value to specify set membership. In figure 2 for example, a one-to-many link has been generated, something neither possible in HTML nor in RDF. Interesting is also that XLink permits both inbound and outbound links. An inbound link is constituted by an arc from an external resource, located with a locator-type element, into an internal resource and is one possibility for modularity in XML.

Arc-type elements may have traversal attributes, one category are behavioral attributes which allows a certain kind of modularity, namely presentation time inclusion. If the attribute *show="embed"*

```
<!-- A local resource -->
      <actress xlink:label="maria">
          <firstname>Brigitte</firstname<
          <surname>Helm</surname<
     </actress>
<!-- A remote resource -->
      <movie xlink:label="metropolis"
             xlink:href="metropolis.xml"/>
<!-- An arc that binds them -->
      <acted xlink:type="arc" xlink:from="maria"
             xlink:to="metropolis"/>
```

Figure 1: Binary links with XLink

```
<divas xlink:title="German divas 1920s">
  <actress xlink:label="maria">
     <firstname>Brigitte</firstname>
     <surname>Helm</surname>
  </actress>

  <movie xlink:label="silent" xlink:title="Metropolis"
               xlink:href="metropolis.xml"/>
  <movie xlink:label="silent" xlink:title="Alaraune"
               xlink:href="alaraune.xml"/>
  <acted xlink:type="arc" xlink:from="maria"
                          xlink:to="silent"/>
        ...
<divas>
```

Figure 2: N-ary links for XLink

is stated for an Xlink arc, the referenced resource is embedded into the current document at interpretation-time (this is kínd of a lazy-load). The additional attribute *actuate* controls the event when the arc should be traversed[2].

This kind of lazy evaluation is problematic with regard to ontologies. As ontologies provide logical theories, "all knowledge" of any inferencing or deduction task must be gathered a-priori to ensure logical correctness. Additionally, the handling of an XLink is left to the application[3]

2.2 Inclusion technologies

In this subsection we focus on the modularization technologies recently proposed for XML. These technologies may be distinguished into:

1. External parsed (or text) entities, as defined by the XML 1.0 Recommendation [2]
2. XLinks (with embed behavior), as defined by the W3C XLink Working Draft.
3. XInclusions [10], as defined by the W3C XInclusion Working Draft.

External entities. An external parsed entity is declared in an XML (or SGML) document by an entity declaration without a notation. A reference to a parsed entity may occur practically anywhere in a document, between elements or within them, using the syntax &entity;. The entity itself may contain text, complete elements, or a mixture of them. It may not contain any declarations. XML does require that entity content be well-formed XML. In other words, one cannot have an element start tag in the document whose end tag is in a referenced entity, or vice versa. This is necessary so that a document may be checked to be well-formed even if the entity references are not replaced. This technology is not suitable for the Semantic Web, as the declaration of external entities requires DTDs. Additionally, the DOCTYPE declaration requires that the document element must be named, which is a unnecessary requirement.

XInclude [10]. is a processing model and syntax for general purpose inclusion. Inclusion is accomplished by merging a number of XML Infosets into a single composite Infoset. This implies that such processing occurs ̈below ̈the application level and without its intervention. Thus, the XInclude processor is responsible for validating the result infoset. The merging of infosets possibly leads to ID/IDREF conflict resolution and namespace preservation issues, not addressed by the working draft. Furthermore, the possibility to use XPOINTER range references exists, which makes maintainability questionable again.

XML Schema. The need for inclusion was also recognized for XML Schema. Due to the non-existence of general solutions at the time of creation a proprietary solution was sought. XML Schema [11] provides means to export certain elements of the schema for public usage. Additionally, facilities for importing elements from other schemas and a mechanism to completely include a referenced schema exist. This inclusion is not made visible to agents consuming the composite schema. This raises severe digital rights problems as the original provider is not recognizable anymore. The idea of defining a set of exported elements, stemming from experience with programming languages and distributed database schema definition systems, does not make sense for the Semantic Web. This export set suggests that there is value in distinguishing the internal implementation of a module from the features or interfaces that it provides for reuse by others[4], which is surely not the intent for ontologies.

3. EXTENDING RDF

This section is separated into two main parts. First, we collect different requirements for enabling a modularized Semantic Web. Second, these different requirements serve as input for defining a language extension to RDF that enables a modularized Semantic Web that recognizes the means offered by existing technologies.

3.1 Requirements

3.1.1 Import mechanisms

RDF only supports binary named links between different resources. While this is sufficient for schemaless metadata it only presents the basic technologies for conceptual relations, viz. the means for references between concepts in ontologies, but no means exist to determine whether referenced entities are actually defined and conceptually valid structures.

For example (see figure 3) the property *hasFather* in the *people ontology* is supposed to be a sub property of *hasParent*, which was defined in the *animal ontology*. Neither can one assure that *hasParent* exists nor that it is a property[6].

[2] traversal can take place onload or onactivate

[3] "...embedding affects only the display of the relevant resources; it does not dictate permanent transformation of the starting resource" [3]. If for example the ending resource is XML, it is not parsed as if it was part of the starting resource. Thus, embedded functionality of XLink is aimed at display behavior and not at true inclusion.

[4] [11] : "For example, a schema defined to describe an automobile might intend that its definitions for 'automobile' and 'engine' be building blocks for use in other schemas, but that other constructs such as 'screw' or 'bolt' be reserved for internal use.".

[6] Thus, the people ontology cannot be validated to be correct RDF

Animal ontology (available at &animal; [5]):

```
<rdf:RDF
  xmlns="&animal;#"
  xmlns:rdf="&rdf;#"
  xmlns:s="&s;#">
<s:Class rdf:about="#Animal"/>
<s:Class rdf:about="#Male">
  <s:subClassOf rdf:resource="#Animal"/>
</s:Class>
<s:Class rdf:about="#Female">
  <s:subClassOf rdf:resource="#Animal"/>
</s:Class>
<s:Class rdf:about="#Human">
  <s:subClassOf rdf:resource="#Animal"/>
</s:Class>
<s:Class rdf:about="#Lion">
  <s:subClassOf rdf:resource="#Animal"/>
</s:Class>

<rdf:Property rdf:about="#hasParent">
  <s:domain rdf:resource="#Animal"/>
  <s:range rdf:resource="#Animal"/>
</rdf:Property>

<rdf:Description rdf:about="#Marjan">
  <rdf:type rdf:resource="#Lion" />
</rdf:Description> </rdf:RDF>
```

People ontology

```
<rdf:RDF
  xmlns="&people;#"
  xmlns:rdf="&rdf;#"
  xmlns:s="&s;#">
<s:Class rdf:about="#Man">
  <s:subClassOf rdf:resource="&animal;#Human"/>
  <s:subClassOf rdf:resource="&animal;#Male"/>
</daml:Class>

<s:Class rdf:about="#Woman">
  <s:subClassOf rdf:resource="&animal;#Human"/>
  <s:subClassOf rdf:resource="&animal;#Female"/>
</s:Class>

<rdf:Property rdf:about="#hasFather">
  <s:subPropertyOf
        rdf:resource="&animal;#hasParent"/>
  <s:range
        rdf:resource="&animal;#Male"/>
</rdf:Property> </rdf:RDF>
```

Figure 3: Two Example RDF ontologies

Thus, in order to enable conceptual references across RDF models in a Web-like manner, we need a means to import entities that are defined somewhere else to take RDF out of ontological opaqueness. Clearly such an import primitive must locate the point of import. The established URIs without fragments suffice for this task, of course. We do not consider URNs here as they are not widely used.

3.1.2 Inclusion mechanisms

Inclusion mechanisms are different from import mechanisms with respect to the extension: Here, the complete RDF model is included whereas only specific parts are included with respect to importing.

Inclusion allows the decomposition of ontologies into individual parts and should therefore be a requirement for the Semantic Web, as it minimizes the effort to construct new ontologies. First, the overall effort required for the engineering of ontologies can be split among many shoulders. Second, decomposition not only simplifies construction and maintenance of ontologies, but also facilitates that ontologies become logically cohesive. Therefore leading to loosely coupled modules that denote single abstractions - a requirement for reusability in other applications.

Schema using tools such as the validating RDF parser [12]

3.1.3 Digital rights

It is clear that when using modularization in the Semantic Web one has to provide means for copyright management. This is needed in order to know who provided which information and created which artefact.

3.2 RDF Language Extension

3.2.1 rdfm:include

We propose to extend the basic RDF vocabulary by a new property rdfm:include (cf. figure 4) for the inclusion of another RDF model into the calling RDF model. This primitive can be understood by RDF parsers and specialized processors (working after parse time).

```
<rdf:RDF
  xmlns="&rdfm;#"
  xmlns:rdf="&rdf;#"
  xmlns:s="&s;#">

<!-- Source tagging -->

<rdf:Property rdf:ID="source">
  <s:comment>
  Identifies the source URI of a statement
  </s:comment>
  <s:domain rdf:resource="#Statement" />
</rdf:Property>

<!-- Inclusion mechanism -->

<rdf:Property rdf:ID="include" />

<!-- Import mechanisms -->

<rdf:Property rdf:ID="importFrom" />

<rdf:Property rdf:ID="transitiveImportFrom">
  <s:subPropertyOf rdf:resource="#importFrom"/>
</rdf:Property>

<rdf:Property rdf:ID="schemaAwareImportFrom">
  <s:subPropertyOf rdf:resource="#transitiveImportFrom"/>
</rdf:Property>
</rdf:RDF>
```

Figure 4: Modular RDF: the extended RDF vocabulary [7]

This property could be used in any RDF model to include the statements declared in another RDF model. Thus, the following statement would include the content of the animal ontology (cf. figure 3) in another RDF model:

```
<rdf:Description rdf:about="">
  <rdfm:include rdf:resource="&animal;"/>
</rdf:Description>
```

To meet the aforementioned requirement of digital rights, all statements found in the included RDF file have to be identified with their source. This can be implemented by tagging statement with their respective source URIs. Tagging statements can only be achieved using reification. Thus, we need to augment the RDF vocabulary (cf. figure 4) with a new property rdf:source that can only be validly applied to Statements [8].

[8] The reader may note that multiple source tags can be defined if identical statements occur in different source files

Thus, for each statement in an included RDF model, new reified statements are added to the calling RDF model. Consider the following statement in the animal ontology (cf. figure 3) for example

```
<s:Class rdf:about="&animal;#Animal"/>
```

This statement would be added in reified and tagged form, using the following set of statements:

```
<rdf:Description>

<rdf:subject rdf:resource="&animal;#Animal" />

<rdf:predicate
rdf:resource="&rdf;#type"/>

<rdf:object rdf:resource=&s;#Class"/>

<rdf:type
rdf:resource="&rdf;#Statement"/>

<rdfm:source rdf:resource="&animal;"/>

</rdf:Description>
```

3.2.2 *rdfm:import*

Notably, rdfm:include includes all statements found in another RDF file. This does not meet the demands of an import mechanism, which has to work on a lower granularity. To meet this demand, the RDF vocabulary has to be augmented with several new primitives that provide:

- A means to import statements about a given resource
- A means to transitively import statements about a given resource
- A means to schema-aware import of statements

rdfm:importFrom. In the simplest case only all statements on a given resource should be imported into the calling RDF model. We introduce a new property rdfm:importFrom to achieve this (cf. figure 4). The subject of a statement using the property[9] specifies the resource which should be imported into the calling RDF model whereas the object of the statement specifies the source RDF model, where statements about the subject should be taken from. Imported statements are represented in reified form with additional source identification to meet the digital rights requirement.

For example, the following statement

```
<rdf:Desciption rdf:about="&animal;#Male">
    <rdfm:importFrom rdf:resource="&animal;" />
</rdf:Description>
```

would add all statements about the resource *Male* from the animal ontology to the calling RDF model. In our example only one statement is found about Male, namely that it is a class. Hence, only one statement would be added:

```
<rdf:Description>

<rdf:subject rdf:resource="&animal;#Male"/>

<rdf:predicate
rdf:resource="&rdf;#type" />

<rdf:object
rdf:resource="&s;#Class"/>

<rdf:type
rdf:resource="&rdf;#Statement"/>

<rdfm:source rdf:resource="&animal;"/> </rdf:Description>
```

[9] These are statements of the following form: (resource, rdfm:importFrom, source)

rdfm:transitiveImportFrom. In many applications the importFrom is not sufficient as we can only embed the first level of resource references using importFrom. Consider the following example:

```
<rdf:Desciption rdf:about="&animal;#hasParent">
    <rdfm:transitiveImportFrom
       rdf:resource="&animal;"/>
</rdf:Description>
```

Here, the importFrom operation would only add statements that have hasParent as a subject, viz. the information that hasParent is a property whose domain is a resource Animal and whose range is again Animal. No information about Animal, e.g. that it is a class, would be included by importFrom.

The operation transitiveImportFrom targets this issue by additionally importing all statements on referenced resources. Thus, for the given example the following set of statements would additionally be pasted into the calling RDF model[10]:

```
<rdf:Description>

<rdf:subject rdfm:resource="&animal;#Animal"/>

<rdf:predicate
rdf:resource="&rdf;#type"/>

<rdf:object
rdf:resource="&s;#Class"/>

<rdf:type
rdf:resource="&rdf;#Statement"/>

<rdfm:source rdf:resource="&animal;"/>

</rdf:Description>
```

rdfm:schemaAwareImportFrom. One can easily see that even transitiveImportFrom is not sufficient to take RDF out of ontological opaqueness. For example if one transitively imports all information on the lion Marjan[11], of course all information on Marjan as well as all super classes of Lion are imported into the calling ontology but not the properties that are valid for any of these classes, thus another primitive is required to take the semantics of a RDF schema into account. Eventually

```
<rdf:Desciption rdf:about="&animal;#Marjan">
    <rdfm:schemaAwareImportFrom
    rdf:resource="&animal;" />
</rdf:Description>
```

would therefore add all statements on the property hasFather. Of course, the implementation of schemaAwareImportFrom is the most complex operation a processor has to fulfill.

Discussion. The following points are finally important to mention with respect to our presented proposal for extending RDF with import facilities:

[10] Additionally to the statements about *hasParent*, which are not shown for sake of brevity.

[11] Marjan is the lion who survived years of conflict and ill-treatment in Afghanistan and died at Kabul zoo. The 25 year-old beast who was half-blind, lame and almost toothless died of old age only weeks after an international animal rescue mission arrived to help him. The only lion in Kabul zoo, he was a gift from Germany in more peaceful times 23 years ago, and became something of a symbol of survival against the odds. Among his reported exploits are killing and eating a Taleban fighter who climbed into his enclosure to prove his bravery. The man's brother attacked the lion with a grenade in revenge, leaving it lame and blind in one eye.

- It does not rely on XInclude, as this approach is not applicable for RDF. The proposed merging of infosets would lead to two <RDF> </RDF> elements, which is incompatible with the RDF syntax specification. Furthermore, the source identification is not possible, as inclusion is invisible to the document consumers.

- The operation rdfm:include and the rdfm:importFrom property family are left in the RDF model. As statements from modules are only inserted at runtime this is not problematic. It is also important information for consuming agents that are interested in the way the viewed information is assembled.

- Modularization operations are generally transitive. Cyclic references are allowed. The usefulness of cyclic references has been shown in [4]. Even if cyclic references suggest that modules should be merged. We still keep the ability for organizational purposes. Infinite recursion can be avoided using simple means.

4. APPLICATIONS AND GUIDELINES

In this section we introduce applications where the foundations of our conceptual framework for modularizing RDF-based models have been successfully used. Additionally, we provide engineering guidelines for modularized ontologies.

4.1 Re-engineering Existing Resources

Experiences have shown that when developing an ontology-based system, conceptual resources, e.g. in the form of thesauri, lexical-semantic nets, related domain and application ontologies are already available. Furthermore, it has been seen that for the development of ontology-based information systems typically only parts of existing resources are to be used. Therefore, in our approach, we convert existing resources onto a common representation format, namely RDF-Schema. Based on this representation we generate application specific modules by applying bottom-up ontology pruning techniques based on a given set of text relevant for a specific domain [7].

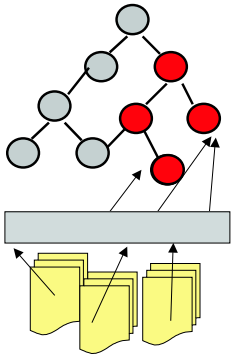

Figure 5: Applying Ontology Pruning to create modules

We take the assumption that the occurrence of specific concepts and conceptual relations in web documents are vital for the decision whether or not a given concept or relation should remain in an ontology. We take a frequency based approach determining concept frequencies in a corpus. Entities that are frequent in a given corpus are considered as a constituent of a given domain. To determine domain relevance ontological entities retrieved from a domain corpus are compared to frequencies obtained from a generic corpus. The user can select several relevance measures for frequency computation. The ontology pruning algorithm uses the computed frequencies to determine the relative relevancy of each concept contained in the ontology. All existing concepts and relations which are more frequent in the domain-specific corpus remain in the ontology. The user may also control the pruning of concepts that are neither contained in the domain-specific nor in the generic corpus. This pruning approach has been successfully applied in the following domains:

- GermaNet pruning for an insurance intranet application [7]: In this approach we used the German version of WordNet as a basis for generating an insurance-specific module, that supported an intranet-based knowledge management application.

- WordNet pruning for Reuters news document clustering: WordNet has been used as a semantic backbone for clustering Reuters news documents. The overall Wordnet lexical semantic net has been pruned on the basis of a set of selected Reuters documents, thus a news-specific module has been generated

- AGROVOC[12] pruning for the animal feed application: Finally, AGROVOC is a thesaurus provided by United Nations Food and Agricultural Organization, describing terms in the context of food and agriculture. It has been used as a basis for developing a module that exclusively describes the "animal feed" domain, for which a metadata-driven search engine will be built.

Using modularization techniques on top of the pruning results has the advantage that we do not create new ontologies. Instead we focus on the application specific part of a given ontology without defining new resources.

In general, we want to mention that there is a lack of modularity in current ontologies. Although, there exists the clear and good separation of top-level, domain, task and application ontologies (see [6]), there are no real-world ontologies and applications that are based on this principle. Most ontologies are not modular, neither by task, nor by domain. Therefore, ontology integration should modularize the namespace of a domain and separate task-oriented knowledge from the domain knowledge.

4.2 Engineering Guidelines

Much work has been described in the area of merging, mapping and integrating ontologies using very different approaches. The point of all those approaches is that they try to establish interoperability between syntactically and semantically heterogeneous conceptual models. Typically, this is done in an "ex-post" way, when the systems have already been established and are running.

Our engineering approach for modularized ontologies pursues another idea, namely the "ex-ante" establishment of interoperability. This approach allows the ontology engineer to import reusable modules from existing ontologies. The reader may note that this approach is already implicitly used in several existing commercial software products, e.g. in the area of knowledge management. Typically, these systems are divided in a standard basic conceptual model (describing basic concepts like documents, person's metadata, etc.) and a domain specific part of the conceptual model (describing domain-specific concepts like topic hierarchies, etc.). The point is that every KM application based on the basic conceptual data model can exchange data on this level, but it cannot exchange

[12] http://www.fao.org/agrovoc/

data on the domain-specific part of the conceptual model. We pick up this approach in an explicit way in the sense that we provide basic conceptual models in the form of ontology modules, e.g. for documents, persons, etc. There are many design goals within this modularization framework, e.g. to create coherent sets of semantically related modules, to support the creation of subsets and supersets of ontology modules for specific purposes, to facilitate future development by allowing modules to be upgraded or replaced independently of other modules and to encourage and facilitate the reuse of common modules by developers.

In the Semantic Web, modules have to be made accessible via a search engine for ontology modules. The search allows to query on a lexical (e.g. by providing a set of concept and property labels that should be contained in the module) and a conceptual layer (e.g. by providing a set of RDF statements that should be contained in the module). We are currently developing such a "ontology search engine" on the basis of our previous work for measuring the similarity between ontologies and parts of it [9].

5. RELATED WORK

Modularization is an established principle in software engineering, nevertheless, generally only acyclic inclusions are possible. Import mechanisms exist in all major programming languages and allow programmers to use classes, functions and methods defined in other modules by explicit naming.

Knowledge based systems introduced means for modularization in the early nineties. The LOOM system [8] provided an acyclic graph of inclusion relationships. Means for importing are not included, although references to symbols defined in other (non-included) ontologies are possible. Notably, the declarative semantics of ontologies are endangered by this, as the definition of those symbols is not visible.

Ontolingua [4] allows for modular organization in the ontology library system, organizes units into modules and allows cyclic inclusions. Additionally referenced ontologies can be extended by polymorphic refinement and restriction. RDF automatically supports some of those refinements (i.e. adding new domains and ranges[13]). Notably, we cannot support restrictions as RDF is not expressive enough to support the required translation axioms.

ONIONS [5] is a methodology that highlights the stratified design of ontologies. They propose different naming policies to achieve the modular organization or stratified storage of ontologies [5]. They show that disjointed partitioning of classes can facilitate modularity, assembling and integrating of ontologies.

As reported in section 2, several means for modularization are proposed for the XML world. XLink allows presentation time inclusion. Handling of inclusion in XLink is left to the application, which is not sufficient for the Semantic Web. No means for importing exist. XInclude introduces real inclusion for XML but does not allow for importing either. Furthermore the issue of conflicting XML identifiers is not recognized, which will come up in implementations.

XML Schema introduces proprietary means for inclusion as well as importing, but these operations are not made visible to agents consuming the composite schema. This raises severe digital rights problems as the original provider is not recognizable anymore. The idea of defining a set of exported elements, stemming from experience with programming languages and similar schema definition systems, does not make sense for the Semantic Web. This export set suggests that there is value in distinguishing the internal implementation of a module from the features or interfaces that it provides for reuse by others.

The recently proposed DAML+OIL ontology language recognized the need for modularity but only considers inclusion. Unfortunately, the established wording was not conceived. DAML+OIL follows the tradition of SHOE, where usage of other ontologies can be specified. Both approaches make the source of modular information opaque and present specialized approaches for ontologies only. Neither one presents means for imports.

6. CONCLUSION

Based on the general principle of modularization that has been an important design issue for the World Wide Web, and, in general the basis for powerful software systems, we have presented general principles and an approach for establishing a modularized Semantic Web.

Thus, ontologies and other RDF models can become a unit of composition which make context dependencies explicit. RDF models can be deployed independently and are subject to composition by third parties. This way, ontologies become logically cohesive, loosely coupled modules that denote single abstractions. Therefore they simplify the construction and maintenance and ensure reusability in other contexts. They also meet modularization requirement that was recognized for the upcoming Ontology Web Language[14].

In the future we plan to provide a simple implementation for a modularity processor and embed modularity management in our ontology engineering environment. We will also further our considerations of using URNs to allow replication of mission-critical RDF models.

7. REFERENCES

[1] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 2001.

[2] T. Bray, J. Paoli, and C.M. Sperberg-McQueen. Extensible markup language (XML) 1.0. Technical report, W3C, 1998. http://www.w3.org/TR/1998/REC-xml-19980210.

[3] Steve DeRose, Eve Maler, and David Orchard. XML Linking Language (XLink) Version 1.0, W3C Recommendation, 2001. http://www.w3.org/TR/xlink/.

[4] R. Fikes, A. Farquhar, and J. Rice. Tools for assembling modular ontologies in ontolingua. Technical Report KSL-97-03, Knowledge Systems Laboratory,Stanford University, 1997.

[5] Aldo Gangemi, Domenico M. Pisanelli, and Geri Steve. An overview of the ONIONS project: Applying ontologies to the integration of medical terminologies. *Data Knowledge Engineering*, 31(2):183–220, 1999.

[6] N. Guarino. Formal ontology and information systems. In *Proceedings of FOIS'98 – Formal Ontology in Information Systems, Trento, Italy, 6-8 June 1998*. IOS Press, 1998.

[7] J.-U. Kietz, R. Volz, and A. Maedche. A method for semi-automatic ontology acquisition from a corporate intranet. In *EKAW-2000 Workshop "Ontologies and Text", Juan-Les-Pins, France, October 2000.*, 2000.

[8] R. MacGregor. LOOM users manual. Technical Report ISI/EP-22, USC/ Information Sciences Institute, 1990.

[9] A. Maedche and S. Staab. Measuring similarity between ontologies. In *Technical Report, E0448, University of Karlsruhe*, 2001.

[13] Multiple ranges are demanded by the RDF Working Group

[14] See requirement 3 in the OWL requirements document, currently at http://km.aifb.uni-karlsruhe.de/owl

[10] Jonathan Marsh and David Orchard. XML Inclusions (XInclude) Version 1.0, W3C Working Draft , 2001. http://www.w3.org/TR/xinclude/.

[11] Henry S. Thompson, David Beech, Murray Maloney, and Noah Mendelsohn. XML Schema – W3C Recommendation, 2001. http://www.w3.org/TR/xmlschema-1/.

[12] Karsten Tolle. Validating rdf parser: A tool for parsing and validating rdf metadata and schemas. Master's thesis, University of Hannover, 2000.