Towards Annotation using DAML+OIL

Sean Bechhofer Carole Goble
Information Management Group
Department of Computer Science,
University of Manchester,
Kilburn Building
Oxford Road
Manchester M13 9PL
UK
{seanb,carole}@cs.man.ac.uk,
http://img.cs.man.ac.uk

ABSTRACT

Semantic metadata will play a significant role in the provision of the Semantic Web. Agents will need metadata that describes the content of resources in order to perform operations, such as retrieval, over those resources. In addition, if rich semantic metadata is supplied, those agents can then employ reasoning over the metadata, enhancing their processing power. Key to this approach is the provision of annotation, both through automatic and human means. In this paper, we describe a prototype annotation tool that we are developing during the COHSE project, allowing the description of semantic metadata using the DAML+OIL language. The development of the tool has exposed a number of issues concerning the use of a language like DAML+OIL for semantic annotation and we discuss these.

1. INTRODUCTION

The Semantic Web (SW) vision, as articulated by Tim Berners-Lee [3], is of a Web in which resources are accessible not only to humans, but also to automated processes, e.g., automated "agents" roaming the web performing useful tasks such as improved search (in terms of precision) and resource discovery, information brokering and information filtering. The automation of tasks depends on elevating the status of the web from machine-readable to something we might call machine-understandable. The key idea is to have data on the web defined and linked in such a way that its meaning is explicitly interpretable by software processes rather than just being implicitly interpretable by humans.

Within this vision of the SW, ontologies have become an increasingly important research topic. This is a result both of their usefulness in a range of application domains [27, 20, 26], and of the pivotal role that they are set to play in the development of the SW.

To realise this vision, it will be necessary to annotate web resources with *metadata* (i.e., data describing their content/functionality). In particular, we may wish to annotate resources with *semantic* metadata that provides some indication of the content of a resource. This is a further step along the way from simple textual annotations, as the intention within the SW context is that this information will be accessible not only to humans but also to software agents. In order to do this we require languages which will support the representation of semantic metadata. Standardisation proposals for metadata languages are already recommendations or candidate recommendations of the World Wide Web Consortium (W3C), in particular the Resource Description Framework (RDF) and RDF Schema (RDF(S)) – see [10] for a discussion of the roles of these languages and of XML/XML Schema. However, such annotations will be of limited value to automated processes unless they share a common understanding as to their meaning. Ontologies can help to meet this requirement by providing a "representation of a shared conceptualisation of a particular domain" and a shared, controlled vocabulary that can be communicated across people and applications [14, 15].

In this paper we discuss our initial experience of building an annotation tool based on a rich ontological language. The paper has three basic sections. First, we discuss DAML+OIL, a proposed ontology language for the web. Second, we provide an overview of a prototype annotation tool for annotation using DAML+OIL. The construction of the tool has highlighted a number of issues concerning the use of DAML+OIL for annotation, and these are discussed in the final section of the paper.

2. DAML+OIL: A LANGUAGE FOR THE SEMANTIC WEB

A prerequisite for a widespread use of ontologies is a joint standard for their description and exchange. RDF Schema (RDF(S)) itself is already recognisable as an ontology/knowledge representation language: it talks about classes and properties (binary relations), range and domain constraints (on properties), and subclass and subproperty (subsumption) relations. However, RDF(S) is a relatively primitive language (the above is an almost complete descrip-

tion of its functionality), and more expressive power would clearly be necessary/desirable in order to describe resources in sufficient detail. Moreover, such descriptions should be amenable to *automated reasoning* if they are to be used effectively by automated processes.

These considerations led to the development of the Ontology Inference Layer OIL [11] and latterly DAML+OIL [8], ontology languages that extend RDF(S) with a much richer set of modelling primitives.

OIL [11] was a language proposed as a knowledge representation language for the web and web-based applications. OIL coupled modelling primitives commonly used in frame-based ontologies, with the simple, clean and *well-defined semantics* of an expressive Description Logic (DL). The frame primitives facilitate tool building and ontology exchange, the DL facilitates the provision of automated reasoning services, in particular consistency and subsumption checking. Thus a modeller is offered the best of both worlds in both development and deployment of an ontology.

In addition, efforts were made to maximising compatibility with emerging web standards. These standards, such as RDF(S), make it easier to use ontologies consistently (consistent meaning for the elements of the ontology) across the web.

In a similar development, the DAML [8] programme was investigating the definition of an agent markup language for the web. The two initiatives have now been combined, leading to the definition of the DAML+OIL language, which has the following characteristics:

- An *underlying mapping* to an expressive Description Logic ($\mathcal{SHIQ}$) [18] provides a well defined semantics and a clear understanding of the language's formal properties. The DL gives DAML+OIL the ability and flexibility to compose classes and slots to form new expressions using boolean connectives, unlimited nesting of class elements, transitive and inverse slots, general axioms, etc. For example, we are able to define sufficiency conditions for concepts as well as necessary conditions. Moreover, we are able to use *automated reasoning support* to automatically compute subsumption (isa) relations between concepts, or to check the consistency and coherency of the classification and its concepts. This is highly useful when collaboratively building substantial ontologies or when ontologies are reused or merged [24]. The mapping also provides a mechanism for the provision of practical reasoning services by exploiting implemented DL systems, e.g., the FaCT system [17]. This means that an ontology expressed in DAML+OIL can be reasoned over by the FaCT reasoner (see Section 2.3).

- A *machine-readable syntactic encoding in the languages of the web.* RDF(S) [4] is a proposed mechanism for deploying metadata. DAML+OIL is defined as an extension of RDF(S), thereby making DAML+OIL ontologies (partially) accessible to any "RDF(S)-aware" application [5]. An ontology in DAML+OIL can be used by an agent that is not DAML+OIL aware but is RDF(S) aware. Thus DAML+OIL is a potential management device for the general Semantic Web.

- A *layered architecture*, avoiding the temptation to throw everything into the core language, mixing up features that cannot be reasoned over with those that can be. Thus the limits are clear and explicit.

DAML+OIL forms a key piece of work strongly related to the W3C's Semantic Web Activity. The language has been described in more detail elsewhere [11], but we provide here a flavour of the DAML+OIL approach. DAML+OIL allows the definition and description of classes (concepts), slots (relationships), individuals (instances) and axioms within an ontology.

2.1 Class Definitions

In DAML+OIL, class definitions are provided through the use of assertions or axioms. These assert that, for example, a particular class `cat` is a subclass of the class `animal`. Alternatively, axioms can assert equality between classes, say that `animal` is equivalent to `beast`. A key aspect of DAML+OIL is that these axioms and assertions need not simply concern atomic concepts, such as `animal` or `cat`, but can also use concept operators such as the boolean operators `and`, `or` and `not`. In addition, *restriction* expressions that represent role restriction or quantification can be used which themselves may employ complex fillers. This is in contrast to conventional frame representation systems, where in general, slot constraint fillers and superclasses must be class names.

For example, we may define `carnivore` as the conjunction of `animal` and the restriction that all fillers for `eats` must be of type `animal`. As a more complex example, we could define a `fussy-carnivore` as the conjunction of `animal` with the restriction that all fillers for `eats` must be the conjunction of `animal` with the restriction that the animal must `come-from India`, i.e. a fussy carnivore only eats Indian animals.

In this way, arbitrarily complex descriptions can be built up.

2.2 Property Definitions

Property definitions gives the name of the property and allows additional properties of the slot to be asserted, i.e. inverses, or whether the slot has transitive, symmetric or functional properties. Slots also form hierarchies, so we may also specify the names of any superslots.

In keeping with RDF(S), domain and range restrictions on a slot can also be specified, and as with class descriptions, domain and range restrictions can be arbitrary class expressions such as anonymous class expressions or boolean combinations of class names and class expressions, again extending the expressivity of traditional frame languages. All assertions made about slots are used by the reasoner, and may induce hierarchical relationships between classes.

2.3 Reasoning and DAML+OIL

The reasoning services offered by a Description Logic support the development and incremental maintenance of an ontology [24]. Highly optimised implementations of sound and complete tableaux subsumption algorithms for very expressive DLs such as $\mathcal{SHIQ}$ can be used in spite of the high worst-case complexity. Thus an ontology expressed using DAML+OIL can be verified using the FaCT reasoner. The key reasoning services are:

Subsumption checking between two concept descriptions, C and D, C subsumes D, when the set of individuals that are instances of D are always a subset of the individuals that are instances of C.

Classification organises a collection of concept expressions into a partial order based on the subsumption check. This provides a lattice of definitions, ranging from the general to the specific. Composed definitions have their position implicitly determined automatically. Thus classification is a dynamic process where new compositional expressions can be added to an existing hierarchy.

Concept satisfiability checks whether a concept description can never have instances because of inconsistencies or contradictions in the model.

When classifying an ontology, a number of new subsumption relationships may be discovered (due to the class definitions in the model). This can be useful during the ontology construction phase [25].

2.4 Layers of DAML+OIL

DAML+OIL has a *layered architecture*, avoiding the temptation to throw everything into the core language, mixing up features that cannot be reasoned over with those that can be. Thus the limits are clear and explicit. The idea is that features such as rules can be included as layers, carefully preserving the bounds of what can be reasoned with but without excluding features that are desirable.

3. ANNOTATION

In this section we present the basic architecture and philosophy behind our prototype annotation tool.

3.1 COHSE

Our interest in annotation here is within the context of the COHSE (Conceptual Open Hypermedia Service) project [6]. COHSE aims to bring together an open hypermedia architecture (in particular the Distributed Links Service [7] or DLS) with ontological services in order to provide an architecture for the Semantic Web [12]. Key to the COHSE approach is the ability to annotate resources with semantic medata – this metadata is then used to link documents (in particular to provide links *out* of documents). COHSE's current prototype employs a specialist browser plugin[1] that applies links to documents.

Up to now, our prototypes have used a very simple approach. Word-matching within the text of documents has been used to determine candidate link anchor opportunities, with a lexicon attached to the ontology providing the bridge between the document and the concepts. We are now at a stage where we wish to employ the use of explicit metadata associated with the documents – this will require semantic annotations and tools which facilitate production of those annotations.

3.1.1 Linking as Annotation

Koivunen et al. [19] discuss approaches to Web annotations and categorise systems as, in the main, either **proxy-based** or **browser-based**. In a proxy-based approach, the annotations and document are merged by the proxy, with the browser seeing only the merged documents. In a browser-based approach, a specialist browser application will merge the annotations with the original documents while browsing. Annotations can be stored separately and provided via some annotation service (or kept within the proxy itself). This bears many comparisons with the DLS, where instead of annotations we are adding links.

In fact the provision of dynamic linking as used by the COHSE project can be seen as a kind of annotation – in this case hypertext links are being provided rather than some textual annotation. COHSE's current architecture uses the browser-based approach, although we are investigating the use of a proxy (as was initially employed by the DLS).

Note that within COHSE, the purpose of annotation is not simply to populate a knowledge base for retrieval, as the annotations will be used to derive link anchors for outward links from resources.

3.2 The Tool

Our annotator has been implemented using a number of existing components and is provided as a browser plug-in. Providing the annotation tool as a plug-in to an existing web browser has a number of advantages. The user can continue to browse documents in a familiar environment with access to their own bookmarks and browsing preference. In addition, the browser can take care of the difficulties of dealing with badly-formed HTML, a perennial problem when dealing with resources on the web. By using a plug-in architecture, the annotator has access to the DOM object built by the browser and can base annotations on that.

Figure 1 shows the basic architecture that we are employing. The annotation tool interacts with a web browser and a collection of ontologies, providing annotations which are then stored in an annotation service or RDF repository.

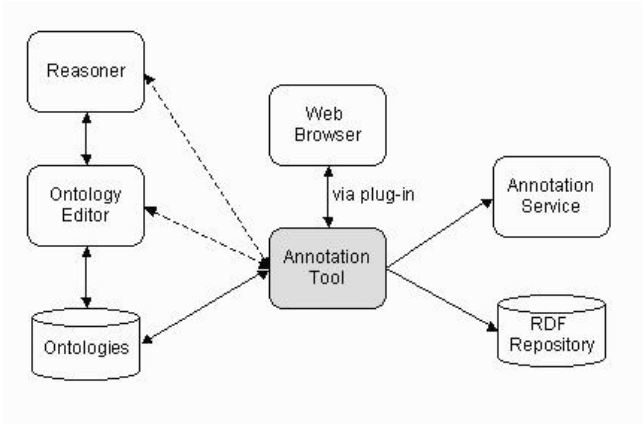

Figure 1: Basic Architecture

Those ontologies may be built with the help of a reasoner. In addition, the annotation tool may use functionality provided by an ontology editing component, or even a reasoner as discussed below.

The annotator consists of an extra toolbar which is added to the user's web browser (here we describe the Internet Explorer version, although the Mozilla client plugin is similar). The tool bar offers a number of actions:

- Annotate the currently selected text with the currently selected concept.

[1]Versions based on both Mozilla and Microsoft's Internet Explorer have been developed

- Open the concept browser. This opens up the interface which allows the user to specify a concept to be used for the annotation.
- Save the annotation session.
- Load the results from an earlier annotation session.

3.2.1 Concept Browsing

OilEd [1] is a simple editor for the OIL and DAML+OIL languages. It adopts a frame-like approach that allows users to construct ontologies using a familiar "user-friendly" interface. To date, some 700 downloads of the tool have been requested from the OilEd web site[2]. OilEd was developed in Java, and as a result the components used within the interface can be easily reused in the annotator. In particular, the OilEd interface components allow us to provide a concept browser that can construct arbitrarily complex and nested concept expressions. Figure 2 shows the concept browser with the simple concept `Researcher` selected – this will be familiar to anyone who has used the OilEd application.

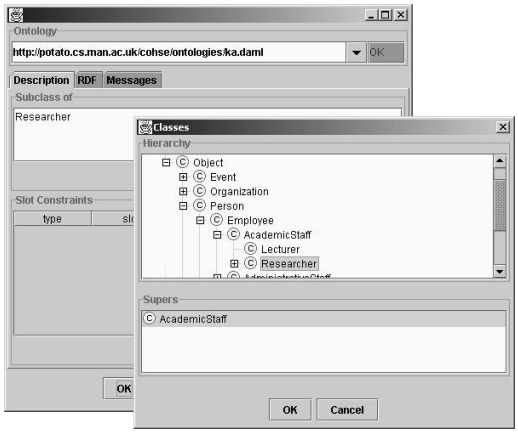

Figure 2: Concept Browser

3.2.2 An Example Annotation

Figure 3 shows the Web browser. A region of the document has been selected for annotation. The annotation toolbar can be seen below the Google toolbar on the browser. Once a resource in a region and an appropriate concept have been selected, the annotation can be made. This associates the selected region with the selected concept in an internal structure. Figure 4 shows the situation after the annotation has been made. A marker is added to the document rendering to indicate the presence of the annotation. Note, however, that the annotation has not been added to the actual document. When the mouse is moved over this marker, the concept forming the annotation is shown. Once the annotations have been made, there are a number of actions that we may take.

- Save the annotation, using RDF or a similar representation. Figure 5 shows the RDF corresponding to the annotation made above. We are using an extension of the W3C annotation schema[3] that provides a specialization of the `http://www.w3.org/2000/10/annotation-ns#body` property. The provision of an RDF Schema based representation for DAML+OIL allows us to use a uniform representation for the annotation – the annotations will be accessible to any agent that understands the annotation schema and basic RDF(S) (although without knowledge of DAML+OIL or reasoning the agent may not be able to infer any extra information).
- Alternatively, we can send the annotation to an Annotation Service such as Annotea[4].
- Finally, if the original document is within our control, we can insert the annotation into the document. This is, of course, a less satisfactory solution, but is the approach that has been adopted in some early experiments in annotation (such as the DAML+OIL homework[5]).

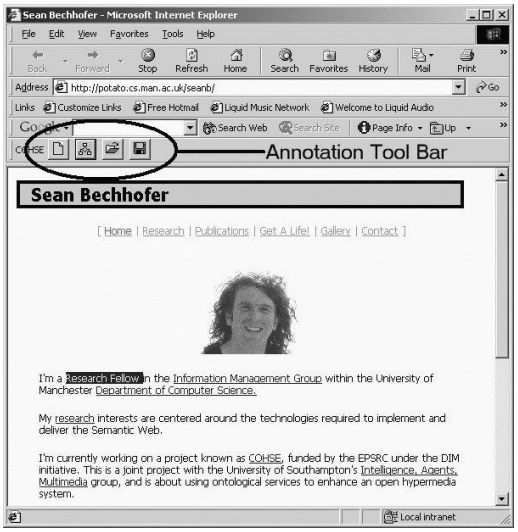

Figure 3: Browser before annotation

Our current implementation offers only the first of these options – a facility for saving annotations as RDF.

In this example (see Figure 5), a single concept has been used for the annotation (shown in italics). Note also the XPointer-like expressions used to identify the region of the document which has been selected. Work is underway to bring this into line with the W3C XPointer candidate recommendation[6]. To look at a more complex example, we may annotate a page about Indian animals using the concept shown in Figure 6 This will then result in the RDF shown in Figure 7 which again has the concept shown in italics.

4. DISCUSSION

In this section, we focus on three areas of interest which have arisen during our experience of building our prototype:

[2] `http://img.cs.man.ac.uk/oil`
[3] `http://www.w3.org/2000/10/annotation-ns`
[4] `http://www.w3.org/2001/Annotea`
[5] `http://www.daml.org/homework/`
[6] `http://www.w3.org/TR/xptr`

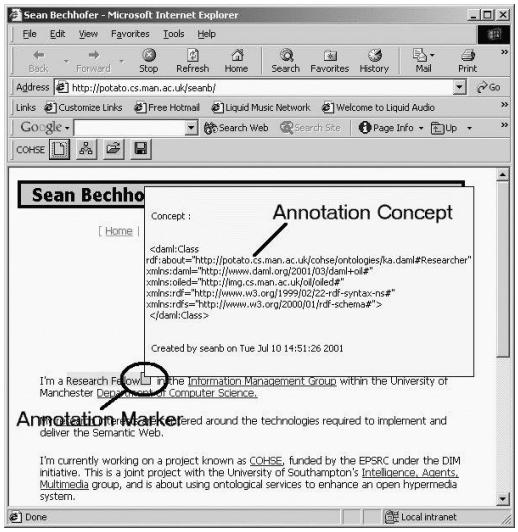

Figure 4: Browser after annotation

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-nx#"
         xmlns:dc="http://purl.org/dc/elements/1.1"
         xmlns:anno="http://www.w3.org/2000/10/annotation-ns#"
         xmlns:cohse="http://cohse.semanticweb.org/annotation-ns#">
  <rdf:Description>
    <rdf:type resource="http://www.w3.org/2000/10/annotation-ns#Annotation"/>
    <rdf:type resource="http://cohse.semanticweb.org/annotation-ns#ConceptAnnotation"/>
    <anno:annotates
       rdf:resource=
      "http://potato.cs.man.ac.uk/seanb/#P(2)/TEXT/Research(0),0::P(2)/TEXT/Fellow(0),6"/>
    <anno:context/>
    <dc:creator>seanb</dc:creator>
    <anno:created>Tue Jul 10 14:51:26 2001</anno:created>
    <dc:date>Tue Jul 10 14:51:26 2001</dc:date>
    <cohse:concept>
      <daml:Class
        rdf:about="http://potato.cs.man.ac.uk/cohse/ontologies/ka.daml#Researcher"
        xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
        xmlns:oiled="http://img.cs.man.ac.uk/oil/oiled#"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
      </daml:Class>
    </cohse:concept>
  </rdf:Description>
</rdf:RDF>
```

Figure 5: Simple RDF Annotation

- Anonymous Descriptions;
- Ontology Containment;
- Instance vs. Aboutness.

4.1 Anonymous descriptions

Perhaps the key aspect of DAML+OIL within this context is the ability to describe arbitrary concept expressions – i.e. not just simple class expression such as `cat` and `dog`, but complex expressions such as `animal` with `4 legs` and `long-fur`. This means that any annotation tool must be able to support the ability to construct such arbitrary expressions, and any framework for the representation and handling of the annotations must be capable of dealing with such expressions.

We can then use **classification**, as described above, to then position these anonymous descriptions within the concept hierarchy provided by the ontology. This classification comes into play when the COHSE client system uses the annotations to determine linking opportunities as described in more detail in [6, 12].

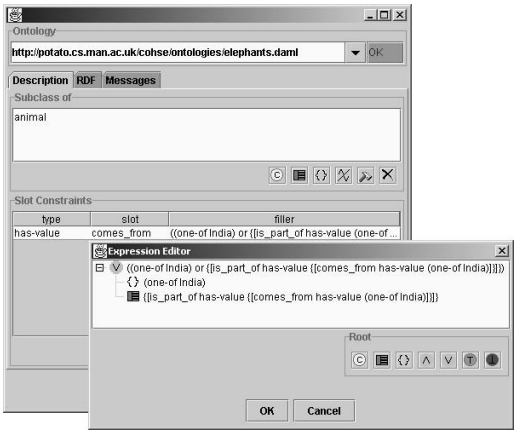

Figure 6: Concept Browser with complex expression

A key point here is that descriptions need not be introduced until they are needed, which means that provided the components are present in the ontology, annotations can be as detailed as required, without the original ontologist or modeller having to provide conceptual descriptions matching all eventualities.

As has been discussed elsewhere [25, 22], reasoning can play a part during the construction phase of the ontology. Our current prototype simply uses functionality taken from the OilEd application in order to allow the user to form complex concept descriptions. The current prototype makes no use of any reasoning during annotation. A possible use of reasoning would be to ensure that the conceptual descriptions which are being constructed are consistent and satisfiable w.r.t. the axioms in the ontology. Alternatively, we could provide an ontology-driven interface [2] as is used within the TAMBIS [13] project (see Figure 8). The TAMBIS interface allows the user to construct conceptual expression which are then translated to queries over bioinformatics data sources. A similar approach would allow users to incrementally construct concept expressions for annotation. The key aspect of the TAMBIS interface here is the use of a reasoner to determing the appropriate specialisation options that are presented to the user.

4.2 Ontology Containment

The question "what constitutes an ontology?" becomes important when we consider annotation within the Semantic Web. The context within which annotations are made is vital for the correct and consistent interpretation of those annotations. With an approach like DAML+OIL, the classes and properties in an ontology are defined by a number of statements. So far, examples of DAML+OIL ontologies tend to be given as URIs, with an XML serialization of those RDF statements forming the body of the resource. As an example, take the ontology shown in Figure 9.

This tells us something about cats (as represented by the class `http://ontos-r-us.com/moggy#cat`), in particular that they are animals and are furry. To be more explicit, within the context of this particular collection of statements (or collection of RDF triples), we can determine that a cat is a furry animal. However, in order to know this, we have to know about those statements, and we thus have to be able

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-nx#"
         xmlns:dc="http://purl.org/dc/elements/1.1"
         xmlns:anno="http://www.w3.org/2000/10/annotation-ns#"
         xmlns:cohse="http://cohse.semanticweb.org/annotation-ns#">
  <rdf:Description>
    <rdf:type resource="http://www.w3.org/2000/10/annotation-ns#Annotation"/>
    <rdf:type resource="http://cohse.semanticweb.org/annotation-ns#ConceptAnnotation"/>
    <anno:annotates
       rdf:resource=
       "http://www.bnhs.org/services/books_animals.htm#P(2)/FONT(0)/TEXT/Indian(0),
                                                    0::P(2)/FONT(0)/TEXT/Animals(0),7"/>
    <anno:context/>
    <dc:creator>seanb</dc:creator>
    <anno:created>Tue Jul 10 16:20:51 2001</anno:created>
    <dc:date>Tue Jul 10 16:20:51 2001</dc:date>
    <cohse:concept>
     <daml:Class xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
            xmlns:oiled="http://img.cs.man.ac.uk/oil/oiled#"
            xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
      <daml:intersectionOf parseType="daml:collection">
       <daml:Class
         rdf:about="http://potato.cs.man.ac.uk/cohse/ontologies/elephants.daml#animal">
       </daml:Class>
       <daml:Restriction>
        <daml:onProperty
          rdf:resource="http://potato.cs.man.ac.uk/cohse/ontologies/elephants.daml#comes_from">
        </daml:onProperty>
        <daml:hasClass>
         <daml:Class>
          <daml:unionOf parseType="daml:collection">
           <daml:Class>
            <daml:oneOf parseType="daml:collection">
             <daml:Thing
               rdf:about="http://potato.cs.man.ac.uk/cohse/ontologies/elephants.daml#India">
             </daml:Thing>
            </daml:oneOf>
           </daml:Class>
           <daml:Restriction>
            <daml:onProperty
              rdf:resource="http://potato.cs.man.ac.uk/cohse/ontologies/elephants.daml#is_part_of">
            </daml:onProperty>
            <daml:hasClass>
             <daml:Class>
              <daml:oneOf parseType="daml:collection">
               <daml:Thing
                 rdf:about="http://potato.cs.man.ac.uk/cohse/ontologies/elephants.daml#India">
               </daml:Thing>
              </daml:oneOf>
             </daml:Class>
            </daml:hasClass>
           </daml:Restriction>
          </daml:unionOf>
         </daml:Class>
        </daml:hasClass>
       </daml:Restriction>
      </daml:intersectionOf>
     </daml:Class>
    </cohse:concept>
  </rdf:Description>
</rdf:RDF>
```

Figure 7: Complex RDF Annotation

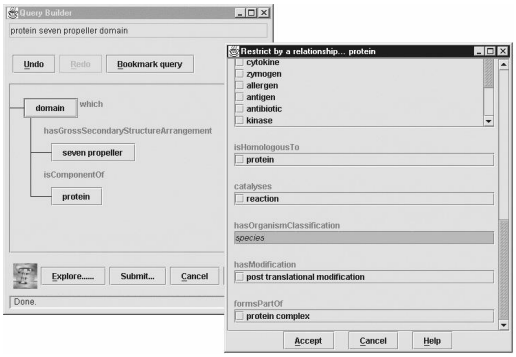

Figure 8: TAMBIS Query Interface

```
<?xml version="1.0" ?>
<rdf:RDF xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns="http://ontos-r-us.com/moggy">
  <daml:Ontology rdf:about="">
    <rdfs:comment>An Ontology</rdfs:comment>
    <daml:versionInfo>1.0</daml:versionInfo>
  </daml:Ontology>
  <daml:Class rdf:ID="http://ontos-r-us.com/moggy#animal"/>
  <daml:Class rdf:ID="http://ontos-r-us.com/moggy#furry thing"/>
  <daml:Class rdf:ID="http://ontos-r-us.com/moggy#cat">
    <rdfs:subClassOf>
      <daml:Class rdf:about="http://ontos-r-us.com/moggy#furry thing"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <daml:Class rdf:about="http://ontos-r-us.com/moggy#animal"/>
    </rdfs:subClassOf>
  </daml:Class>
</rdf:RDF>
```

Figure 9: Cats are furry animals

to talk about the use of the class `http://ontos-r-us.com/moggy#cat` within the context of these statements. This then requires us to be able to specify the context within which an annotation that uses the concept was made.

Having the class itself is not enough. Just because the class is given by the url `http://ontos-r-us.com/moggy#cat`, we can make no assumptions about where the statements that make up the ontology reside. We might *expect* that the URI `http://ontos-r-us.com/moggy` will direct us to the RDF schema that provides the statements defining cats, but we cannot guarantee this. There is no requirement within the DAML+OIL approach that ontologies must be represented as XML-serializations of RDF statements sitting at a single URI, and alternative representations or storage mechanisms may be employed.

This is not a question of namespaces – the namespace is a simple mechanism for disambiguating terms – but is instead about how we might specify exactly the context or terms of reference within which an annotation was made. A key motivation for the use of ontologies is the ability to share information *unambiguously*. Due to the mechanics of the RDF/DAML+OIL representations, there is thus a requirement for *explicitness* here, ensuring that the ontology within which the classes are being used is known.

Topic Maps [21] are an alternative representation for semantic metadata influenced by subject indices as found in books. From a formal reasoning perspective, topic maps suffer in some ways in comparison to DAML+OIL – the DAML+OIL language provides a rich collection of concept forming operators (such as conjunction, negation and explicit quantification) with well defined semantics, facilitating the use of reasoners such as the FaCT system to organize and classify hierarchies. Topic Maps do, however, recognise the containment issue with the inclusion of a scoping mechanism.

Two projects concerned with semantic annotation were SHOE and Ontobroker. In the SHOE approach [16], ontologies are marked up in an extension to HTML and are stored and referred to using a URL. The annotation (which occurs within the document) contains an explicit reference to the ontology. Ontobroker [9] again uses in-document markup (relying on an extension of the `<A>` tag). In the Ontobroker examples provided the ontology is referred to implicitly– presumably it is the responsibility of the crawler agent to determine the ontology that is in use. Both of these systems adopt an approach whereby a collection of pages are registere with a service. An agent then crawls those pages and uses the metadata to provide retrieval. This could be considered in some way a "closed" or "centralised" approach. With the COHSE approach, pages are annotated, and at browsing or reading time, the annotations are then used to augment the documents. Of course within COHSE we will still require a number of known documents (or access to a

service) in order to provide the linking targets.

4.3 Instance vs. Aboutness

A further key question to address when we consider annotation is that of *instance-of* vs. *aboutness*. RDF(S) has a built in property `rdf:type` that allows us to make assertions about individual resources. For example, take the RDF statement shown in Figure 10.

```
<rdf:Description rdf:about="http://widgets.com/staff/bob.html">
  <rdf:type rdf:resource="http://ontos-r-us.com/work#Manager"/>
</rdf:Description>
```

Figure 10: Bob is a Manager

This says that the resource `http://widgets.com/staff/bob.html` is an instance of the class `http://ontos-r-us.com/work#Manager`. RDF is well set up to deal with such assertions. However, there may often be situations where we want to make an assertion that a particular resource is *about* a particular concept (in terms of its content), rather than saying it is an instance of it. As an example, the page `http://www.nczooeletrack.org/` is about elephants, but is not itself an instance of an elephant – we thus need some mechanism that allows us to refer to the content of resources without having to explicitly type the resources themselves.

Note that we should not confuse RDF's `rdf:about` attribute with "aboutness" as discussed here. Within RDF, `rdf:about` is a mechanism that relates a resource to RDF statements concerning it, rather than describing the content of some resource.

In our current prototype, we are effectively side-stepping the issue. All that the annotation asserts is that the selected resource has an annotation which consists of the selected concept. There is no instance-of assertion here.

One possible solution to this problem is to introduce a new relation, say `cohse:hasSubject` and use this to related the resources to the classes that represent their subjects as in Figure 11.

```
<rdf:Description rdf:about="http://www.nczooeletrack.org/">
  <cohse:hasSubject rdf:resource="http://ontos-r-us.com/beasts#elephant"/>
</rdf:Description>
```

Figure 11: This site is about Elephants

This solves this problem, but relies on a shared understanding of the semantics of our extension to the basic RDF model (e.g. the `<cohse:hasSubject>` property) if such an annotation is to interpreted consistently.

Returning to Topic Maps, the approach there is to divide the world up in terms of Topics, Associations and Occurrences. Broadly speaking, Topics are related via Associations and Occurrences represent concrete manifestations of the Topics. Topics can include both classes and instances (in the DL sense), while Occurrences are *not* instances of the Topics. Thus concepts like `Opera Singer` and `Pavarotti` will both occur as topics, with a relationship between `Pavarotti` and `Opera Singer` representing the fact that he's a singer. A web page containing, say a `JPG` image of Pavarotti, or a `WAV` of him singing *Nessun Dorma* can then be considered to be occurrences of Pavarotti. Occurence roles are used to identify the kind of role that the occurrence is playing and its relationships with the topic – for example the `JPG` is playing the role of a picture of Pavarotti, but the key factor here is that it is not an *instance* of Pavarotti.

It may be that a similar approach will help in dealing with the instance/aboutness problem.

4.4 Related Work

As discussed above, Ontobroker [9] and SHOE [16] were projects concerned with semantic annotation using ontologies.

OntoAnnotate [23] is a tool for building semantic annotations for web pages. However, it is not clear whether the tool is supporting the construction of a relationship between the web resource and the ontology, or is intended to allow the population of a knowledge base with information taken from the page.

4.5 Concluding Remarks

Semantic metadata is set to play a major part in the implementation of the Semantic Web and annotation will be a key mechanism for supplying the metadata which will then be used by agents as they retrieve information. We have presented some ideas on an annotation tool which is intended to support the use of DAML+OIL in resource annotation. This is, the authors freely admit, still preliminary work and much is to be done, in particular an evaluation of the tool. However, we have already encountered some issues relating to the use of representations such as DAML+OIL and RDF in this context. In particular, we must support complex, anonymous expressions. There is a need for explicit definitions of context which will allow agents to interpret annotations in a consistent manner. The issue of what exactly constitutes an ontology, and how we might refer to this is also in need of clarification. Finally the distinction between instance and aboutness must be supported.

5. ACKNOWLEDGEMENTS

This work was supported by EPSRC Grant GR/M75426. The prototype annotator was implemented by Tim Miles-Board of the University of Southampton.

6. REFERENCES

[1] S. Bechhofer, I. Horrocks, C. Goble, and R. Stevens. OilEd: a Reason-able Ontology Editor for the Semantic Web. In *To appear in KI2001, Joint German/Austrian conference on Artificial Intelligence*, Vienna, September 2001.

[2] S. Bechhofer, R. Stevens, G. Ng, A. Jacoby, and C.A. Goble. Guiding the User: An Ontology Driven Interface. In *UIDIS, Workshop on User Interfaces to Data Intensive Systems*, pages 158–161, Edinburgh, 1999. IEEE Computer Society.

[3] T. Berners-Lee. *Weaving the Web*. Orion Business Books, 1999.

[4] D. Brickley and V.R. Guha. Resource description framework schema specification 1.0. W3C Candidate Recommendation, 2000. `http://www.w3.org/TR/rdf-schema`.

[5] J. Broekstra, M. Klein, S. Decker, D. Fensel, F. van Harmelen, and I. Horrocks. Enabling knowledge representation on the Web by Extending RDF Schema. In *Proceedings of WWW10, the 10th World Wide Web conference, Hong Kong*, May 2001.

[6] L. Carr, S. Bechhofer, C. A. Goble, and W. Hall. Conceptual Linking: Ontology-based Open Hypermedia. In *WWW10, Tenth World Wide Web Conference*, Hong Kong, May 2001.

[7] L. Carr, D. De Roure, W. Hall, and G. Hill. The Distributed Link Service: A Tool for Publishers, Authors and Readers. *World Wide Web Journal*, 1(1):647–656, 1995.

[8] DAML. Darpa Agent Markup Language Program. `http://www.daml.org`.

[9] S. Decker, M. Erdmann, D. Fensel, and R. Studer. Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information. In R. Meersman, Z. Tari, and S. Stevens, editors, *Semantic Issues in Multimedia Systems. Proceedings of DS-8*, pages 351–369. Kluwer Academic Publishers, 1999.

[10] S. Decker, F. van Harmelen, J. Broekstra, M. Erdmann, D. Fensel, I. Horrocks, M. Klein, and S. Melnik. The semantic web — on the respective roles of XML and RDF. *IEEE Internet Computing*, 2000.

[11] D. Fensel, I. Horrocks, F. Van Harmelen, S. Decker, M. Erdmann, and M. Klein. OIL in a nutshell. In *Proc. of EKAW-2000*, LNAI, 2000.

[12] C. A. Goble, S. Bechhofer, L. Carr, D. De Roure, and W. Hall. Conceptual Open Hypermedia = The Semantic Web? In *SemWeb2001 The Second International Workshop on the Semantic Web*, Hong Kong, May 2001.

[13] C.A. Goble, R. Stevens, G. Ng, S. Bechhofer, N.W. Paton, P.G. Baker, M. Peim, and A. Brass. Transparent access to multiple bioinformatics information sources. *IBM Systems Journal*, 40(2):532–551, 2001.

[14] T. R. Gruber. Towards principles for the design of ontologies used for knowledge sharing. In *Proc. of Int. Workshop on Formal Ontology*, 1993.

[15] N. Guarino. Formal ontology and information systems. In *Proc. of FOIS-98*. IOS Press, 1998.

[16] J. Heflin, J. Hendler, and S. Luke. SHOE: A Knowledge Representation Language for Internet Applications. Technical Report CS-TR-4078 (UMIACS TR-99-71), Department of Computer Science, University of Maryland, 1999.

[17] I. Horrocks. Benchmark analysis with fact. In *Proc. TABLEAUX 2000*, pages 62–66, 2000.

[18] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proc. of LPAR'99*, pages 161–180, 1999.

[19] M.-R. Koivunen, D. Brickley, J. Kahan, E. Prud'Hommeaux, and R. R. Swick. The W3C Collaborative Web Annotation Project ... or how to have fun while building an RDF infrastructure. `http://www.w3.org/2000/02/collaboration/annotation/papers/annotationinf%rastructure`, May 2000.

[20] D. L. McGuinness. Ontological issues for knowledge-enhanced search. In *Proc. of FOIS-98*, 1998.

[21] S. Pepper. The TAO of Topic Maps. In *Proceedings of XML Europe*, Paris, France, June 2000.

[22] R.Stevens, I.Horrocks, C. A. Goble, and S. Bechhofer. Building a Reason-able Bioinformatics Ontology Using OIL. In *IJCAI'01 Workshop on Ontologies and Information Sharing*, Seattle, August 2001.

[23] S. Staab, A. Maedche, and S. Handschuh. An Annotation Framework for the Semantic Web. In *The First International Workshop on MultiMedia Annotation*, Tokyo, Japan, January 2001.

[24] R. Stevens, C. A. Goble, and S. Bechhofer. Ontology-based knowledge representation for bioinformatics. *Briefings in Bioinformatics*, 2001.

[25] R. Stevens, C.A. Goble, I. Horrocks, and S. Bechhofer. Building a Bioinformatics Ontology Using OIL. To appear in IEEE Information Technology in Biomedicine special issue on Bioinformatics, 2001.

[26] M. Uschold and M. Grüninger. Ontologies: Principles, methods and applications. *K. Eng. Review*, 11(2):93–136, 1996.

[27] G. van Heijst, A. Schreiber, and B. Wielinga. Using explicit ontologies in KBS development. *Int. J. of Human-Computer Studies*, 46(2/3):183–292, 1997.