CREAM — Creating relational metadata with a component-based, ontology-driven annotation framework

[1]Siegfried Handschuh, [1,2]Steffen Staab, [1,3]Alexander Maedche

[1]Institute AIFB, University of Karlsruhe, D-76128 Karlsruhe, Germany
http://www.aifb.uni-karlsruhe.de/WBS
{sha,sst,ama}@aifb.uni-karlsruhe.de

[2]Ontoprise GmbH, Haid-und-Neu Straße 7, 76131 Karlsruhe, Germany
http://www.ontoprise.de

[3]FZI Research Center for Information Technologies,
Haid-und-Neu Straße 10-14, 76131 Karlsruhe, Germany
http://www.fzi.de/wim

"The Web is about links; the Semantic Web is about the relationships implicit in those links."
Dan Brickley

ABSTRACT

Richly interlinked, machine-understandable data constitutes the basis for the Semantic Web. Annotating web documents is one of the major techniques for creating metadata on the Web. However, annotation tools so far are restricted in their capabilities of providing richly interlinked and truely machine-understandable data. They basically allow the user to annotate with plain text according to a template structure, such as Dublin Core. We here present CREAM (Creating RElational, Annotation-based Metadata), a framework for an annotation environment that allows to construct *relational metadata*, i.e. metadata that comprises class instances and relationship instances. These instances are not based on a fix structure, but on a domain ontology. We discuss some of the requirements one has to meet when developing such a framework, e.g. the integration of a metadata crawler, inference services, document management and information extraction, and describe its implementation, *viz.* Ont-O-Mat a component-based, ontology-driven annotation tool.

Keywords

Metadata, Markup, Annotations, Ontology, DAML+OIL, RDF, Semantic Web

1. INTRODUCTION

Research about the WWW currently strives to augment syntactic information already present in the Web by semantic metadata in order to achieve a Semantic Web that human and software agents alike can understand. RDF(S) or DAML+OIL are languages that have recently advanced the basis for extending purely syntactic information, *e.g.* HTML documents, with semantics. Based on these recent advancements one of the the most urgent challenges now is a knowledge capturing problem, *viz.* how one may turn existing syntactic resources into interlinked knowledge structures that represent relevant underlying information. This paper is about a framework for facing this challenge, called CREAM[1], and about its implementation, Ont-O-Mat.

The origin of our work facing this challenge dates back to the start of the seminal KA2 intiative [1], *i.e.* the initiative for providing semantic markup on HTML pages for the knowledge acquisition community. The basic idea then was that manual knowledge markup on web pages was too error-prone and should therefore be replaced by a *simple* tool that should help to avoid syntactic mistakes.

Developing our CREAM framework, however, we had to recognize that this knowledge capturing task exhibited some intrinsic difficulties that could not be solved by a *simple* tool. We here mention only some challenges that immediately came up in the KA2 setting:

- **Consistency**: Semantic structures should adhere to a given ontology in order to allow for better sharing of knowledge. For example, it should be avoided that people confuse complex instances with attribute types.

- **Proper Reference**: Identifiers of instances, *e.g.* of persons, institutes or companies, should be unique. For instance, in KA2 metadata there existed three different identifiers of our colleague Dieter Fensel. Thus, knowledge about him could not be grasped with a straightforward query.[2]

[1]CREAM: Creating RElational, Annotation-based Metadata.

[2]The reader may see similar effects in bibliography databases. E.g., query for James (Jim) Hendler at the — otherwise excellent — DBLP: http://www.informatik.uni-trier.de/~ley/db/.

- **Avoid Redundancy**: Decentralized knowledge provisioning should be possible. However, when annotators collaborate, it should be possible for them to identify (parts of) sources that have already been annotated and to reuse previously captured knowledge in order to avoid laborious redundant annotations.

- **Relational Metadata**: Like HTML information, which is spread on the Web, but related by HTML links, knowledge markup may be distributed, but it should be semantically related. Current annotation tools tend to generate template-like metadata, which is hardly connected, if at all. For example, annotation environments often support Dublin Core [12], providing means to state, e.g., the name of authors, but not their IDs[3].

- **Maintenance**: Knowledge markup needs to be maintained. An annotation tool should support the maintenance task.

- **Ease of use**: It is obvious for an annotation environments to be useful. However, it is not trivial, because it involves intricate navigation of semantic structures.

- **Efficiency**: The effort for the production of metadata is a large restraining threshold. The more efficiently a tool support the annotation, the more metadata will produce a user. These requirement stand in relationship with the ease of use. It depends also on the automation of the annotation process, e.g. on the preprocessing of the document.

CREAM faces these principal problems by combining advanced mechanisms for inferencing, fact crawling, document management and — in the future — information extraction. Ont-O-Mat, the implementation of CREAM, is a component-based plug-in architecture that tackles this broad set of requirements.[4]

In the following we first sketch two usage scenarios (Section 2). Then, we explain our terminology in more detail, derive requirements from our principal considerations above and explain the architecture of CREAM (Section 3). We describe our actual tool, Ont-O-Mat, in Section 4. Before we conclude, we contrast CREAM with related work, namely knowledge acquisition tools and annotation frameworks.

2. SCENARIOS FOR CREAM

We here only summarize two scenarios, two knowledge portals, for annotation that have been elaborated in [21]:

The first scenario extends the objectives of the seminal KA2 initiative. The KA2 portal provides a view onto knowledge of the knowledge acquisition community. Besides of semantic retrieval as provided by the original KA2 initiative, it allows comprehensive means for navigating and querying the knowledge base and also includes guidelines for building such a knowledge portal. The potential users provide knowledge, e.g. by annotating their web pages in a decentralized manner. The knowledge is collected at the portal by crawling and presented in a variety of ways.

[3]In the web context one typically uses the term 'URI' (uniform resource identifier) to speak of 'unique identifier'.
[4]The core Ont-O-Mat can be downloaded from:
`http://ontobroker.semanticweb.org/annotation`.

The second scenario is a knowledge portal for business analysts that is currently constructed at Ontoprise GmbH. The principal idea is that business analyst review news tickers, business plans and business reports. A considerable part of their work requires the comparison and aggregation of similar or related data, which may be done by semantic queries like"Which companies provide B2B solutions?", when the knowledge is semantically available. At the Time2Research portal they will handle different types of documents, annotate them and, thus, feed back into the portal to which they may ask questions.

3. DESIGN OF CREAM

In this section we explain basic design decisions of CREAM, which are founded on the general problems sketched in the introduction above. In order to provide a clear design rationale, we first provide definitions of important terms we use subsequently:

- **Ontology**: An ontology is a formal, explicit specification of a shared conceptualization of a domain of interest [8]. In our case it is constituted by statements expressing definitions of DAML+OIL classes and properties [7].

- **Annotations**: An annotation in our context is a set of instantiations attached to an HTML document. We distinguish *(i)* instantiations of DAML+OIL classes, *(ii)* instantiated properties from one class instance to a datatype instance — henceforth called attribute instance (of the class instance), and *(iii)* instantiated properties from one class instance to another class instance — henceforth called relationship instance.

  Class instances have unique URIs. Instantiations may be attached to particular markups in the HTML documents, *viz.* URIs and attribute values may appear as strings in the HTML text.

- **Metadata**: Metadata are data about data. In our context the annotations are metadata about the HTML documents.

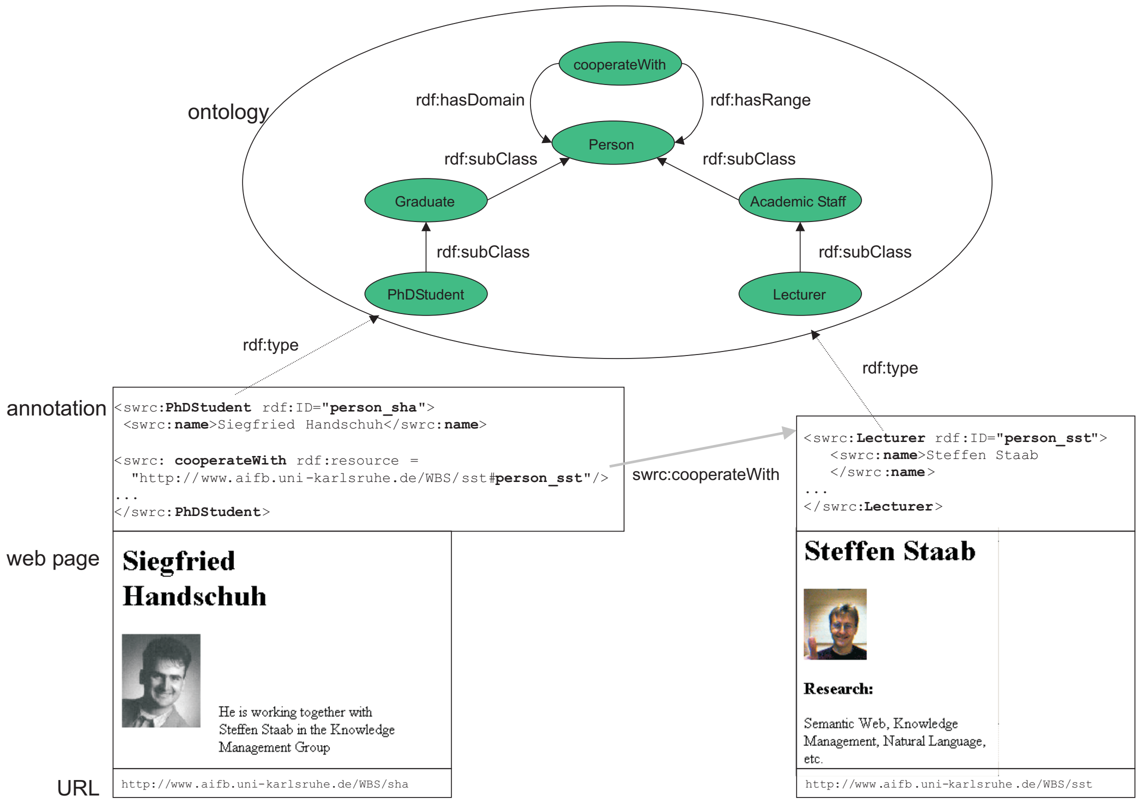
- **Relational Metadata**: We use the term relational metadata to denote the annotations that contain relationship instances.

  Often, the term "annotation" is used to mean something like "private or shared note", "comment" or "Dublin Core metadata". This alternative meaning of annotation may be emulated in our approach by modeling these notes with attribute instances. For instance, a comment note "I like this paper" would be related to the URL of the paper via an attribute instance 'hasComment'.

  In contrast, relational metadata contain statements like 'student Siegfried cooperates with lecturer Steffen', *i.e.* relational metadata contain relationships between class instances rather than only textual notes.

Figure 1 illustrates our use of the terms "ontology", "annotation" and "relational metadata". It depicts some part of the SWRC[5] (semantic web research community) ontology. Furthermore it shows two homepages, viz. pages about

[5]`http://ontobroker.semanticweb.org/ontos/swrc.html`



```
<swrc:PhDStudent rdf:ID="person_sha">
 <swrc:name>Siegfried Handschuh</swrc:name>

<swrc: cooperateWith rdf:resource =
  "http://www.aifb.uni-karlsruhe.de/WBS/sst#person_sst"/>
...
</swrc:PhDStudent>
```

```
<swrc:Lecturer rdf:ID="person_sst">
   <swrc:name>Steffen Staab
   </swrc:name>
...
</swrc:Lecturer>
```

Siegfried Handschuh

He is working together with Steffen Staab in the Knowledge Management Group

http://www.aifb.uni-karlsruhe.de/WBS/sha

Steffen Staab

Research:

Semantic Web, Knowledge Management, Natural Language, etc.

http://www.aifb.uni-karlsruhe.de/WBS/sst

Figure 1: Annotation example

Siegfried and Steffen (`http://www.aifb.uni-karlsruhe.de/WBS/sha` and `http://www.aifb.uni-karlsruhe.de/WBS/sst`, respectively) with annotations given in an XML serialization of RDF facts. For the two persons there are instances denoted by corresponding URIs (`person_sha` and `person_sst`). The `swrc:name` of `person_sha` is "Siegfried Handschuh". In Addition, there is a relationship instance between the two persons: they cooperate. This cooperation information 'spans' the two pages.

3.1 Requirements for CREAM

The difficulties sketched in the introduction directly feed into the design rationale of CREAM. The design rationale links the challenges with the requirements. This results in a N:M mapping (neither functional nor injective). An overview of the matrix is given in Table 1. It shows which modules (requirements) are mainly used to answer challenges set forth in the introduction, *viz.*:

- **Document Viewer**: The document viewer visualizes the web page contents. The annotator may easily provide annotations by highlighting text that serves as input for attribute instances or the definition of URIs. The document viewer must support various formats (HTML, PDF, XML, etc.).

- **Ontology Guidance**: The annotation framework needs guidance from the ontology. In order to allow for sharing of knowledge, newly created annotations must be consistent with a community's ontology. If annotators instantiate arbitrary classes and properties the semantics of these properties remains void. Of course the framework must be able to adapt to varying ontologies in order to reflect different foci of the annotators.

  Furthermore, the ontology is important in order to guide annotators towards creating relational metadata. We have done some preliminary experiments and found that subjects have more problems with creating relationship instances than with creating attribute instances (cf. [22]). Without the ontology they would miss even more cues for assigning relationships between class instances.

  Both ontology guidance and document viewer should be easy to use: Drag'n'drop helps to avoid syntax errors and typos and a good visualization of the ontology can help to correctly choose the most appropriate class for instances.

- **Crawler:** The creation of relational metadata must take place *within* the Semantic Web. During annotation annotaters must be aware of which entities exist in the part of the Semantic Web they annotate. This is only possible if a crawler makes relevant entities immediately available. So, annotators may look for proper reference, i.e. decide whether an entity already has a URI (e.g. whether the entity named "Dieter Fensel" or "D. Fensel" has already been identified by some other annotators) and thus only annotators may recognize whether properties have already been instantiated (e.g. whether "Dieter Fensel" has already be linked to his

Table 1: Design Rationale — Linking Challenges with Required Modules

Requirement				Storage		
			Replication			
General Problem	Document Viewer	Ontology Guidance	Crawler	Annotation Inference Server	Document Management	Information Extraction
Consistency		X		X		
Proper Reference			X	X		
Avoid Redundancy			X	X	X	
Relational Metadata		X	X	X		
Maintenance					X	X
Ease of use	X	X				X
Efficiency	X	X	X	X	X	X

publications). As a consequence of annotators' awareness relational metadata may be created, because class instances become related rather than only flat templates are filled.

- **Annotation Inference Server**: Relational metadata, proper reference and avoidance of redundant annotation require querying for instances, i.e. querying whether and which instances exist. For this purpose as well as for checking of consistency, we provide an annotation inference server in our framework. The annotation inference server reasons on crawled and newly annotated instances and on the ontology. It also serves the ontological guidance, because it allows to query for existing classes and properties.

- **Document Management**: In order to avoid redundancy of annotation efforts, it is not sufficient to ask whether instances exist at the annotation inference server. When an annotator decides to capture knowledge from a web page, he does not want to query for all single instances that he considers relevant on this page, but he wants information, whether and how this web page has been annotated before. Considering the dynamics of HTML pages on the web, it is desirable to store annotated web pages together with their annotations. When the web page changes, the old annotations may still be valid or they may become invalid. The annotator must decide based on the old annotations and based on the changes of the web page.

  A future goal of the document management in our framework will be the semi-automatic maintenance of annotations. When only few parts of a document change, pattern matching may propose revision of old annotations.

- **Information Extraction**: Even with sophisticated tools it is laborious to provide semantic annotations. A major goal thus is semi-automatic annotation taking advantage of information extraction techniques to propose annotations to annotators and, thus, to facilitate the annotation task. Concerning our environment we envisage two major techniques: First, "wrappers" may be learned from given markup in order to automatically annotate similarly structured pages (cf., e.g., [16]). Second, message extraction like systems may be used to recognize named entities, propose coreference, and extract some relationship from texts (cf., e.g., [20]).

Besides of the requirements that constitute single modules, one may identify functions that cross module boundaries:

- Storage: CREAM supports two different ways of storage. The annotations will be stored inside the document that is in the document management component, but it is also stored in the annotation inference server.

- Replication: We provide a simple replication mechanism by crawling annotations into our annotation inference server.

3.2 Architecture of CREAM

The architecture of CREAM is depicted in Figure 2. The complete design of CREAM comprises a plug-in structure, which is flexible with regard to adding or replacing modules. Document viewer and ontology guidance module together constitute the major part of the graphical user interface. Via plug-ins the core annotation tool, Ont-O-Mat, is extended to include the capabilities outlined above. For instance, a plug-in for a connection to a document management system provides document management and retrieval capabilities that show the user annotations of a document he loads into his browser. This feature even becomes active when the user does not actively search for already existing annotations. Similarly, Ont-O-Mat provides extremely simple means for navigating the taxonomy, which means that the user can work without an inference server. However, he only gets the full-fledged semantics when the corresponding plug-in connection to the annotation inference server is installed.

4. IMPLEMENTATION: ONT-O-MAT

This section describes Ont-O-Mat, the implementation of our CREAM framework. Ont-O-Mat is a component-based, ontology-driven markup tool. The architectural idea behind CREAM is a component-based framework, thus, being open, flexible and easily extensible.

In the following subsection we refer to the concrete realization and the particular technical requirements of the components. In subsection 4.2 we describe the functionality of Ont-O-Mat based on an example ontology for annotation that is freely available on the web.

4.1 Ont-O-Mat services and components

The architecture of Ont-O-Mat provides a plug-in and service mechanism. The components are dynamically plug-able to the core Ont-O-Mat. The plug-in mechanism notifies each installed component, when a new component is registered.

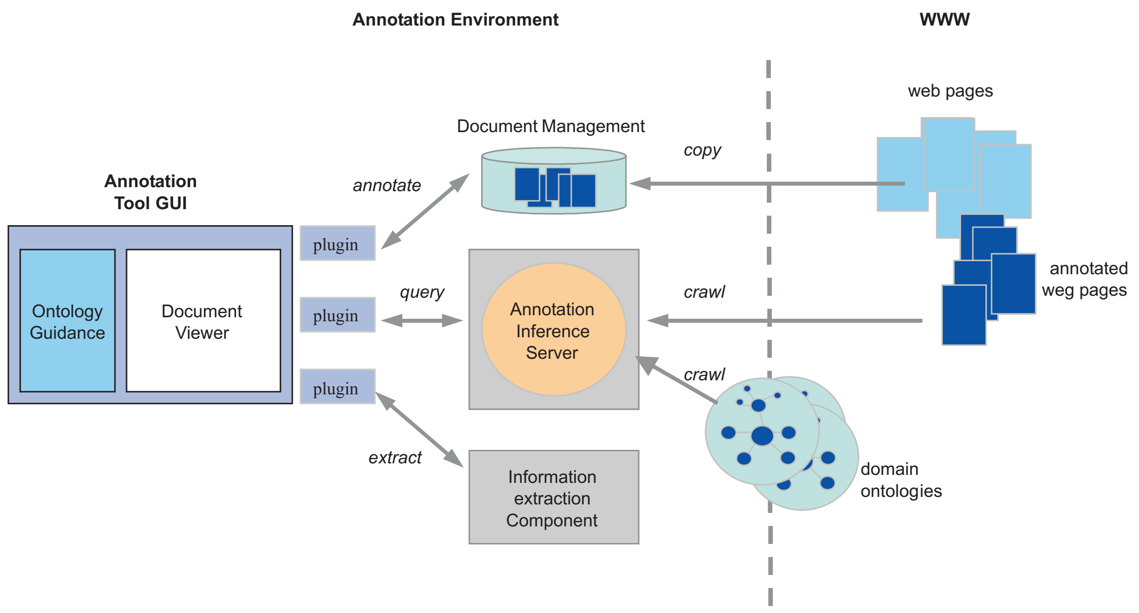


Figure 2: Architecture of CREAM.

Through the service mechanism each component can discover and utilize the services offered by another component [9]. A service represented by a component is typically a reference to an interface. This provides among other things a de-coupling of the service from the implementation and allows therefore alternative implementations.

The Ont-O-Mat services have been realized by components according to the requirements listed in subsection 3.1. So far we have realized the following components: a comprehensive user-interface, component for document-management, an annotation inference-server and a crawler:

- **Document Viewer and Ontology Guidance**: There are various ways how the gained knowledge database can be visualized and thus experienced. On the one hand, the system can be used as a browser. In the annotated web pages, the extracted text fragments are then highlighted and an icon after each fragment is visible. By clicking on the icon, the name of the assigned class or attribute will be shown. On the other hand, the user can browse the ontology and retrieve for one class all instances or for one instance all attributes.

  The underlying data model used for Ont-O-Mat has been taken from the comprehensive ontology engineering and learning system OntoEdit / Text-To-Onto (see [18]).

  Ont-O-Mat works currently in "read-only–mode" with respect to the ontology and only operates on the relational metadata defined on top of the given ontology.

- **Document Management**: A component for document management is required in order to avoid duplicate annotations and existing semantic annotations of documents should be recognized. In our current implementation we use a straight forward file-system based document management approach.

  Ont-O-Mat uses the URI to detect the re-encounter of previously annotated documents and highlights annotations in the old document for the user. Then the user may decide to ignore or even delete the old annotations and create new metadata, he may augment existing data, or he may just be satisfied with what has been previously annotated. In order to recognize that a document has been annotated before, but now appears under a different URI, Ont-O-Mat computes similarity with existing documents by simple information retrieval methods, e.g. comparison of the word vector of a page. If thereby a similarity is discovered, this is indicated to the user, so that he can check for congruency.

- **Annotation Inference Server**: The annotation inference server reasons on crawled and newly annotated instances and on the ontology. It also serves the ontological guidance, because it allows to query for existing classes and properties. We use Ontobroker's [3] underlying F-Logic [14] based inference engine SilRI [2] as annotation inference server. The F-Logic inference engine combines ordering-independent reasoning in a high-level logical language with a well-founded semantics.

- **RDF Crawler**: As already mentioned above, the annotation must take place right within the Semantic Web and not isolated. Therefore, we have built a RDF Crawler[6], a basic tool that gathers interconnected fragments of RDF from the Web and builds a local knowledge base from this data.

  In general, RDF data may appear in Web documents in several ways. We distinguish between *(i)* pure RDF

[6]RDF Crawler is freely available for download at: `http://ontobroker.semanticweb.org/rdfcrawler`.

(files that have an extension like "*.rdf"), *(ii)* RDF embedded in HTML and *(iii)* RDF embedded in XML. Our RDF Crawler relys on Melnik's RDF-API[7] that can deal with the different embeddings of RDF described above. One general problem of crawling is the applied filtering mechanism: Baseline document crawlers are typically restricted by a predefined depth value. Assuming that there is an unlimited amount of interrelated information on the Web (hopefully this will soon hold about RDF data as well), at some point RDF fact gathering by the RDF Crawler should stop. We have implemented a baseline approach for filtering: At the very start of the crawling process and at every subsequent step we maintain a queue of all the URIs we want to analyze. We process them in the breadth-first-search fashion, keeping track of those we have already visited. When the search goes too deep, or we have received sufficient quantity of data (measured as number of links visited or the total web traffic or the amount of RDF data obtained) we may quit.

- **Information Extraction**: This component has not yet been integrated in our Ont-O-Mat tool. Actually, we are near finishing an integration of a simple wrapper approach [15], but we have not yet the message extraction approach for Ont-O-Mat that suggests relevant part of the texts for annotation.

4.2 Using Ont-O-Mat — An Example

Our example is based on the freely available SWRC (**S**emantic **W**eb **R**esearch **C**ommunity)[8] ontology , the successor of the KA2 ontology. The SWRC ontology models the semantic web research community, its researchers, topics, publications, tools, etc. and properties between them. It is available in the form of DAML+OIL classes and properties, in pure RDF-Schema and in F-Logic. The general idea behind SWRC is that the SW research community creates relational metadata according to the SWRC ontology to enable semantic access to their web pages. In the following we shortly explain how Ont-O-Mat may be used for creating relational metadata based on the SWRC ontology.

The annotation process is started either with an annotation inference server or the server process is fed with metadata crawled from the web and the document server. Figure 3 shows the screen for navigating the ontology and creating annotations in Ont-O-Mat. The right pane displays the document and the left panes show the ontological structures contained in the ontology, namely classes, attributes and relations. In addition, the left pane shows the current semantic annotation knowledge base, i.e. existing class instances, attribute instances and relationship instances created during the semantic annotation.

1. First of all, the user browses a document by entering the URL of the web document that he would like to annotate. This step is quite familiar from existing browsers.

2. Then the user selects a text fragment by highlighting it and takes a look on the ontology which fits in the topic and is therefore loaded and visible in ontology browser.

3. There are two possibilities for the text fragment to be annotated: as an instance or as an property. In the case of an instance, the user selects in the ontology the class where the text fragment fits in, e.g. if he has the text fragment "Siegfried Handschuh", he would select the class "PhD Student". By clicking on the class, the annotation gets created and thus the text fragment will be shown as an instance of the selected class in the ontology at the ontology browser.

4. To each created instance, literal attributes can be assigned. The choice of the predefined attributes depends on the class the instance belongs to, e.g. the class "PhD Student" has the attributes name, address, email, and telephone number. The attributes can be assigned to the instance by highlighting the appropriate text fragment of the web document and dragging it to the related property field.

5. Furthermore, the relationships between the created instances can be set, e.g. the PhD Student Siegfried Handschuh "works at" the OntoAgent project and "is supervised" by Rudi Studer. Ont-O-Mat preselects class instances according to the range restrictions of the chosen relation, e.g. the "works at" of a PhD Student must be an Project. Therefore only Projects are offered as potential fillers to the "works at" relation of Siegfried.

[7]http://www-db.stanford.edu/~melnik/rdf/api.html
[8]http://www.semanticweb.org/ontologies/

5. COMPARISON WITH RELATED WORK

CREAM can be compared along three dimensions: First, it is a framework for mark-up in the Semantic Web. Second, it can be considered as a particular knowledge acquisition framework vaguely similar like Protégé-2000[6]. Third, it is certainly an annotation framework, though with a different focus than ones like Annotea [13].

5.1 Knowledge Markup in the Semantic Web

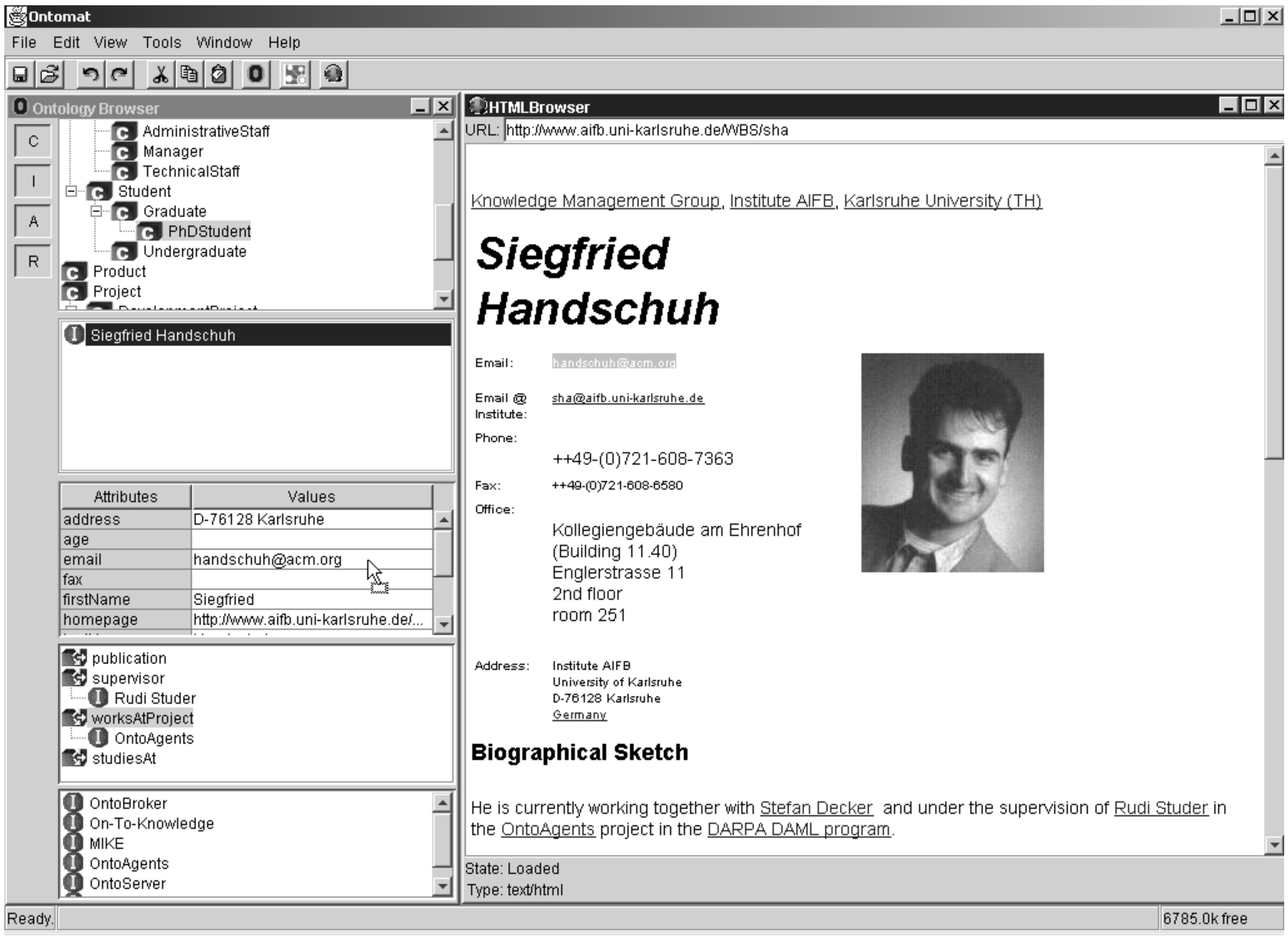
We know of three major systems that intensively use knowledge markup in the Semantic Web, viz. SHOE [10], Ontobroker [3] and WebKB [19]. All three of them rely on knowledge in HTML pages.

They all started with providing manual mark-up by editors. However, our experiences (cf. [5]) have shown that text-editing knowledge mark-up yields extremely poor results, viz. syntactic mistakes, improper references, and all the problems sketched in the introduction.

The approaches from this line of research that are closest to *CREAM* is the *SHOE Knowledge Annotator*[9].

The SHOE Knowledge Annotator is a Java program that allows users to mark-up webpages with the SHOE ontology. The SHOE system [17] defines additional tags that can be embedded in the body of HTML pages. The Knowledge Annotater is less user friendly compared with our implementation Ont-O-Mat. It shows the ontology in some textual lists, whereas Ont-O-Mat gives a graphical visualization of the ontologies. Furthermore, in SHOE there is no direct relationship between the new tags and the original text of the page, i.e. SHOE tags are not annotations in a strict sense.

[9]http://www.cs.umd.edu/projects/plus/SHOE/KnowledgeAnnotator.html



Attributes	Values
address	D-76128 Karlsruhe
age	
email	handschuh@acm.org
fax	
firstName	Siegfried
homepage	http://www.aifb.uni-karlsruhe.de/...

http://www.aifb.uni-karlsruhe.de/WBS/sha

Knowledge Management Group, Institute AIFB, Karlsruhe University (TH)

Siegfried Handschuh

Email: handschuh@acm.org

Email @ Institute: sha@aifb.uni-karlsruhe.de

Phone: ++49-(0)721-608-7363

Fax: ++49-(0)721-608-6580

Office: Kollegiengebäude am Ehrenhof (Building 11.40) Englerstrasse 11 2nd floor room 251

Address: Institute AIFB University of Karlsruhe D-76128 Karlsruhe Germany

Biographical Sketch

He is currently working together with Stefan Decker and under the supervision of Rudi Studer in the OntoAgents project in the DARPA DAML program.

Figure 3: Ont-O-Mat Screenshot.

5.2 Comparison with Knowledge Acquisition Frameworks

The CREAM framework is specialized for creating class and property instances and for populating HTML pages with them. Thus, it does not function as an ontology editor, but rather like the instance acquisition phase in the Protégé-2000 framework [6]. The obvious difference of CREAM to the latter is that Protege does not (and does not intend to) support the particular web setting, *viz.* managing and displaying web pages.

5.3 Comparison with Annotation Frameworks

There are a lot of — even commercial — annotation tools like ThirdVoice[10], Yawas [4], CritLink [23] and Annotea (Amaya) [13].

These tools all share the idea of creating a kind of user comment on the web pages. The term “annotation” in these frameworks is understood as a remark to an existing document. As mentioned before, we would model such remarks as attribute instances only in our framework. For instance, a user of these tools might attach a note like ”A really nice professor!” to the name “Studer” on a web page.

[10]http://www.thirdvoice.com

Annotea actually goes one step further. It allows to rely on an RDF schema as a kind of template that is filled by the annotator. For instance, Annotea users may use a schema for Dublin Core and fill the author-slot of a particular document with a name. This annotation, however, is again restricted to attribute instances. The user may also decide to use complex RDF descriptions instead of simple strings for filling such a template. However, he then has no further support from Amaya that helps him providing syntactically correct statements with proper references.

To summarize, CREAM is used to generate really machine-understandable data and addresses all the problems that come from this objective: relational metadata, proper reference and consistency.

6. CONCLUSION AND FUTURE PLANS

CREAM is a comprehensive framework for creating annotations, relational metadata in particular — the foundation of the future Semantic Web. The framework comprises inference services, crawler, document management system, ontology guidance, and document viewers.

Ont-O-Mat is the reference implementation of CREAM framework. The implementation supports so far the user with the task of creating and maintaining ontology-based DAML+OIL markups, i.e. creating of class, attribute and relationship instances. Ont-O-Mat include an ontology browser for the exploration of the ontology and instances and a HTML browser that will display the annotated parts of the text. Ont-O-Mat is Java-based and provide a plugin interface for extensions for further advancement.

Our goal is a constant advancement of Ont-O-Mat and the CREAM framework in order to answer basic problems that come with semantic annotation.

We are already dealing with many different issues and through our practical experiences we could identify problems that are most relevant in our scenario/settings, KA2 and Time2Research. Nevertheless our analysis of the general problem is far from being complete. Some further important issues we want to mention here are:

- **Information Extraction**: We have done some first steps to incorporate information extraction. However, our future experiences will have to show how and how well information extraction integrates with semantic annotation.

- **Multimedia Annotation**: This requires considerations about time, space and synchronization.

- **Changing Ontologies**: Ontologies on the web have characteristics that influence the annotation process. Heflin & Hendler [11] have elaborated on changes that affect annotation. Future annotation tools will have to incorporate solutions for the difficulties they consider.

- **Active Ontology Evolvement**: Annotation should feed back into the actual ontologies, because annotators may find that they should consider new knowledge, but need revised ontologies for this purpose. Thus, annotation affects ontology engineering and ontology learning.

Our general conclusion is that providing semantic annotation, relational metadata in particular, is an important complex task that needs comprehensive support. Our framework CREAM and our tool Ont-O-Mat have already proved very successful in leveraging the annotation process. They still need further refinement, but they are unique in their design and implementation.

7. ACKNOWLEDGEMENTS.

The research presented in this paper would not have been possible without our colleagues and students at the Institute AIFB, University of Karlsruhe, and Ontoprise GmbH. We thank Kalvis Apsitis (now: RITI Riga Information Technology Institute), Stefan Decker (now: Stanford University), Michael Erdmann, Mika Maier-Collin, Leo Meyer and Tanja Sollazzo. Research for this paper was partially financed by US Air Force in the DARPA DAML project "OntoAgents" (01IN901C0).

8. REFERENCES

[1] R. Benjamins, D. Fensel, and S. Decker. KA2: Building Ontologies for the Internet: A Midterm Report. *International Journal of Human Computer Studies*, 51(3):687, 1999.

[2] S. Decker, D. Brickley, J. Saarela, and J. Angele. A Query and Inference Service for RDF. In *Proceedings of the W3C Query Language Workshop (QL-98), http://www.w3.org/TandS/QL/QL98/, Boston, MA, December 3-4*, 1998.

[3] S. Decker, M. Erdmann, D. Fensel, and R. Studer. Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information. In R. Meersman et al., editors, *Database Semantics: Semantic Issues in Multimedia Systems*, pages 351–369. Kluwer Academic Publisher, 1999.

[4] L. Denoue and L. Vignollet. An annotation tool for web browsers and its applications to information retrieval. In *In Proceedings of RIAO2000*, Paris, April 2000. http://www.univ-savoie.fr/labos/syscom/Laurent.Denoue/riao2000.doc.

[5] M. Erdmann, A. Maedche, H.-P. Schnurr, and Steffen Staab. From manual to semi-automatic semantic annotation: About ontology-based text annotation tools. In *P. Buitelaar & K. Hasida (eds). Proceedings of the COLING 2000 Workshop on Semantic Annotation and Intelligent Content*, Luxembourg, August 2000.

[6] H. Eriksson, R. Fergerson, Y. Shahar, and M. Musen. Automatic generation of ontology editors. In *Proceedings of the 12th Banff Knowledge Acquisition Workshop, Banff, Alberta, Canada*, 1999.

[7] Reference description of the daml+oil (march 2001) ontology markup language. http://www.daml.org/2001/03/reference.html, March 2001.

[8] T. R. Gruber. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 6(2):199–221, 1993.

[9] Siegfried Handschuh. Ontoplugins – a flexible component framework. Technical report, University of Karlsruhe, May 2001.

[10] J. Heflin and J. Hendler. Searching the web with shoe. In *Artificial Intelligence for Web Search. Papers from the AAAI Workshop. WS-00-01*, pages 35–40. AAAI Press, 2000.

[11] J. Heflin, J. Hendler, and S. Luke. Applying Ontology to the Web: A Case Study. In *Proceedings of the International Work-Conference on Artificial and Natural Neural Networks, IWANN'99*, 1999.

[12] Dublin Core Metadata Initiative. http://purl.oclc.org/dc/, April 2001.

[13] J. Kahan, M. Koivunen, E. Prud'Hommeaux, and R. Swick. Annotea: An Open RDF Infrastructure for Shared Web Annotations. In *Proc. of the WWW10 International Conference. Hong Kong*, 2001.

[14] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42, 1995.

[15] J. Klotzbuecher. Ontowrapper. Master's thesis, University of Karlsruhe, to appear 2001.

[16] N. Kushmerick. Wrapper Induction: Efficiency and Expressiveness. *Artificial Intelligence*, 118(1), 2000.

[17] S. Luke, L. Spector, D. Rager, and J. Hendler. Ontology-based Web Agents. In *Proceedings of First International Conference on Autonomous Agents*, 1997.

[18] A. Maedche and S. Staab. Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16(2), 2001.

[19] P. Martin and P. Eklund. Embedding Knowledge in Web Documents. In *Proceedings of the 8th Int. World Wide Web Conf. (WWW'8), Toronto, May 1999*, pages 1403–1419. Elsevier Science B.V., 1999.

[20] *MUC-7 — Proceedings of the 7th Message Understanding Conference*. http://www.muc.saic.com/, 1998.

[21] S. Staab and A. Maedche. Knowledge portals — ontologies at work. *AI Magazine*, 21(2), Summer 2001.

[22] S. Staab, A. Maedche, and S. Handschuh. Creating metadata for the semantic web: An annotation framework and the human factor. Technical Report 412, Institute AIFB, University of Karlsruhe, 2001.

[23] Ka-Ping Yee. CritLink: Better Hyperlinks for the WWW, 1998. http://crit.org/ ping/ht98.html.