

# CREAM — CREAting Metadata for the Semantic Web

Siegfried Handschuh<sup>a</sup> Steffen Staab<sup>a,b,c</sup>

<sup>a</sup>*Institute AIFB, University of Karlsruhe, 76128 Karlsruhe, Germany*  
*<http://www.aifb.uni-karlsruhe.de/WBS>*

<sup>b</sup>*Learning Lab Lower Saxony, Expo Plaza 1, 30539 Hannover, Germany*  
*<http://www.learninglab.de>*

<sup>c</sup>*Ontoprise GmbH, Haid-und-Neu Str. 7, 76131 Karlsruhe, Germany*  
*<http://www.ontoprise.de>*

---

## Abstract

Richly interlinked, machine-understandable data constitute the basis for the Semantic Web. We provide a framework, CREAM, that allows for creation of metadata. While the annotation mode of CREAM allows to create metadata for existing web pages, the authoring mode lets authors create metadata — almost for free — while putting together the content of a page.

As a particularity of our framework, CREAM allows to create *relational metadata*, i.e. metadata that instantiate interrelated definitions of classes in a domain ontology rather than a comparatively rigid template-like schema as Dublin Core. We discuss some of the requirements one has to meet when developing such an ontology-based framework, e.g. the integration of a metadata crawler, inference services, document management and a meta-ontology, and describe its implementation, *viz.* OntoMat, a component-based, ontology-driven Web page authoring and annotation tool.

*Key words:* Semantic Web, Ontology, Metadata, Authoring

---

## 1 Introduction

The Semantic Web builds on metadata describing the contents of Web pages. In particular, the Semantic Web requires *relational metadata*, i.e. metadata that describe how resource descriptions instantiate class definitions and how they are semantically interlinked by properties. We have carried through several case studies that build on this idea of the Semantic Web in order to provide intelligent applications to make knowledge about researchers, about companies and markets, and about research papers accessible by semantic means. An

important cornerstone of the case studies — and many other scenarios in the Semantic Web — is a framework and a mechanism that let user easily and comprehensively contribute relational metadata that the Semantic Web builds on. For this objective, we have developed our semantic annotation framework, CREAM — CREAting MEtadata for the Semantic Web — that we present here.<sup>1</sup> CREAM is geared to allow for the easy and comfortable creation of semantic metadata.

CREAM allows for the *a posteriori* annotation of existing resources. A *posteriori* creation of metadata involves the consideration of an item (e.g. a document) by an agent (possibly, but not necessarily, a human agent) and its description with metadata. For this purpose we distinguish two *modes of interaction*, (*i*), the process may include the identification of elements already existing within the item. For instance, one may consider an HTML page and identify its author by a footer appearing *within* the page. (*ii*), one may classify the item to belong to a category like BUSINESS though neither the term itself nor a synonym or hyponym appears in the item.

A *posteriori* creation of metadata comes with a major drawback. In order to provide metadata about the contents of a Web page, the author must *first* create the content and *second* annotate the content in the additional, a-posteriori, annotation step. To avoid the overhead work, we propose a third mode of interaction, (*iii*), an author has the possibility to easily combine authoring of a Web page *and* the creation of relational metadata describing its content.

Scrutinizing the differences between the **modes of interaction** (*i*), (*ii*), and (*iii*), we found that the major problems one must deal with for an annotation framework are identical. In fact, we found it preferable to hide the border between annotation (i.e. (*i*) and (*ii*)) and authoring (i.e. (*iii*)) as far as possible within CREAM. Some questions, however, were triggered by the need to distinguish the ontology that defines the structure of the relational metadata from its use for annotation or authoring. For this purpose, we here introduce a **meta ontology** that describes how the annotation and authoring modes of OntoMat interfere with classes and properties of the ontology proper. We modularize the ontology parts needed in the metadata creation process from the ones relevant for the targeted content description.

For the CREAM framework we have also provided a reference implementation, viz. OntoMat-Annotizer (OntoMat for short), which is freely available for download.<sup>2</sup>

---

<sup>1</sup> This paper wraps up previous contributions to conference proceedings presented in [23, 20].

<sup>2</sup> <http://annotation.semanticweb.org/ontomat>

In the following we first describe the case studies from which we took a major part of our experiences for guiding the development of CREAM (Section 2). Then, we describe some of the requirements in detail that were derived from the case studies (Section 3). We explain our terminology in more detail and give an example of the metadata we want to create in Section 4. We derive in Section 5 the design of CREAM from the requirements elaborated before. In Section 6, we specify how the meta ontology may modularize the ontology description from the way the ontology is used in CREAM. In Section 7, we explain the major modes of interaction with OntoMat, our implementation of CREAM. Before we conclude, we give a survey of related work in the areas knowledge markup on the Web, knowledge acquisition, annotation environments and authoring environments.

## 2 Case Studies for CREAM

Below, we describe three case studies that we have performed and that have guided our development of CREAM. The case studies have in common that they require the generation of metadata given a HTML document from which a human could identify relevant metadata entities.

During the studies, we repeatedly encountered several principal problems. Some of the problems were mostly syntactic, viz. people would easily make *syntactic errors*, e.g. closing XML parentheses incorrectly or not at all. In the further course, however, we found that people violated *semantic constraints*, too, e.g. they would give a string instead of an object identifier. Also, human annotators would violate *pragmatic constraints*, e.g. have wrong assumptions about the existence of object descriptions in the Semantic Web and, thus, make errors when referring to people.

### 2.1 KA2 Initiative and KA2 Portal

The origin of our work facing the challenge of creating relational metadata dates back to the start of the seminal KA2 initiative [2], *i.e.* the initiative for providing semantic markup on HTML pages for the knowledge acquisition community and its presentation in a Web portal [41]. The KA2 portal<sup>3</sup> 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

---

<sup>3</sup> <http://ka2portal.aifb.uni-karlsruhe.de>

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.

Metadata for KA2 was initially contributed by manually editing HTML pages providing metadata with an ASCII editor. The syntactic errors that arose from the difficulty of this manual task soon became obvious. However, we still had the wrong intuition that a simple tool that would take care of syntax issues would resolve the problem. Thus, OntoPad was developed in order to facilitate editing [14, 5]. However, it was soon found that the semantic and pragmatic concerns are as important as the syntactic issue and could hardly be dealt with by such simple tools that were available then.

## 2.2 *TIME2Research Portal*

In a second case study, we have used semantic annotation in order to provide knowledge about the TIME (telecommunication, IT, multimedia, e-business) markets in the TIME2Research portal [42]. The principal idea of the TIME2Research portal 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. Therefore, we have created a knowledge portal for business analysts that let the analysts provide semantic annotation for incoming HTML, Word and Excel documents. Then, they were able to browse through the metadata and the documents and they could perform semantic searches looking for individual as well as semantically aggregated information provided by annotation.

For the case study we provided OntoAnnotate<sup>4</sup>. OntoAnnotate gave the human annotator an easy-to-use interface. He was allowed to highlight some piece of HTML text, Word text or an Excel cell and then drag’n’drop it onto a corresponding — possibly instantiated — concept, attribute or relation from the ontology. By the control through ontology guidance and browsing of existing facts most of the semantic and pragmatic problems were dealt with. A substantial part of this paper is about this part of OntoAnnotate<sup>5</sup> that is also realized within CREAM.

In the TIME2Research case study we found that people would need the possi-

---

<sup>4</sup> OntoAnnotate is now a commercial tool available from Ontoprise GmbH.

<sup>5</sup> For a description of OntoAnnotate, OntoMat and other ontology tools see OntoWeb public deliverable D1.3 [http://ontoweb.semanticweb.org/About/Deliverables/D13\\_v1-0.zip](http://ontoweb.semanticweb.org/About/Deliverables/D13_v1-0.zip).

bility to easily create documents and their metadata *in one step*, rather than produce a report and create the semantic annotation *a posteriori*. The reason is that in a commercial setting people are even more resistant to overhead work than in a closed-community, academic initiative like KA2.

### 2.3 Authors' Annotations of Paper Abstracts at ISWC-2002

The most recent case study involved the mark-up of papers for the First International Semantic Web conference — ISWC-2002<sup>6</sup>. “Eating their own dog food”, authors of accepted paper were required to annotate title-pages and abstracts of their papers in order to contribute to the actual creation of the Semantic Web and to produce experience in actually providing semantic annotation. The authors could, but were not forced to, use the OntoMat annotation tool and a specifically created ISWC ontology.

The results from the ISWC abstracts annotation study complemented some of the problematic experiences we gained from KA2. Without a tool, even highly trained authors of highly valued papers produced mark-up that obviously was not XML (not to mention RDF) as not all parentheses matched correctly. Others who did not use a tool reused existing RDF code — copying the syntactic errors that the original code contained.

The authoring component of CREAM, would also have simplified life for the ISWC authors, since they could have created the annotated abstract directly instead of first creating the HTML title and abstract description and then adding the semantic annotation. However, the authoring component was not yet stable enough at the time of the study to be used by the authors of accepted papers.

Given the code produced by OntoMat, some third parties actually took advantage of the metadata to build applications on top. M. Frank *et al.* produced a spreadsheet-like summary of some of the core data exploiting their Web-Scripter approach [16]. C. Fillies and F. Weichhardt used their visualization methods in order to create a graphical depiction of the underlying metadata [15].<sup>7</sup>

---

<sup>6</sup> <http://annotation.semanticweb.org/iswc/documents.html>

<sup>7</sup> Cf. also <http://www.semtalk.com/pub/sardinia.htm>.

### 3 Requirements for CREAM

Given the problems with syntax, semantics and pragmatics experienced in the case studies described above, we can now list a more precise set of requirements. Thereby, the principal requirements apply for *a-posteriori annotation* as well as for the *integration of web page authoring with metadata creation* as follows:

- **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 use an attribute, where the ontology requires a concept instance.
- **Proper Reference:** Identifiers of instances, *e.g.* of persons, institutes or companies, should be unique. For instance, the metadata generated in the KA2 case study contained three different identifiers for our colleague Dieter Fensel. Thus, knowledge about him could not be grasped with a straightforward query.<sup>8</sup>
- **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 [10, 11, 30], providing means to state, *e.g.*, the name of authors of a document, but not their IDs<sup>9</sup>. Thus, the only possibility to query for all publications of a certain person requires the querying for some attribute like `FULLNAME` — which is very unsatisfying for frequent names like “John Smith”.
- **Maintenance:** Knowledge markup needs to be maintained. An annotation tool should support the maintenance task. In the remainder of the paper we will provide some infrastructure support for the task.

In particular we provide a baseline document management system to enable the handling of changing documents, facts, and ontologies in the future. However, this needs to be further explored (*e.g.*, along the lines elaborated for robust linking in [40] and Ontology Evolution[44]).

---

<sup>8</sup> 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/>.

<sup>9</sup> In the web context one typically uses the term ‘URI’ (uniform resource identifier) to speak of ‘unique identifier’.

- **Ease of Use:** It is obvious that an annotation environment should be easy to use in order to be really useful. However, this objective is not easily achieved, because metadata creation involves intricate navigation of semantic structures, e.g. taxonomies, properties and concepts.
- **Efficiency:** The effort for the production of metadata is a large restraining threshold. The more efficiently a tool supports metadata creation, the more metadata users tend to produce. This requirement is related to the ease of use. It also depends on the automation of the metadata creation process, e.g. on the preprocessing of the document.
- **Multiple Ontologies:** HTML documents in the semantic web may contain information that is related to different ontologies. Therefore the annotation framework should cater for concurrent annotation with multiple ontologies.

Our framework, CREAM that is presented here, targets a comprehensive solution for metadata creation during web page authoring and a-posteriori annotation. The objective is pursued by combining advanced mechanisms for inferencing, fact crawling, document management, meta ontology definitions, metadata re-recognition, content generation, and (as explained in a companion paper [22]) information extraction. These components — but the last — are explained in the subsequent sections.

## 4 Relational Metadata

We elaborate the terminology we use in our framework, because many of the terms that are used with regard to metadata creation tools carry several, ambiguous connotations that imply conceptually important decisions for the design rationale of CREAM:

- **Ontology:** An ontology is a formal, explicit specification of a shared conceptualization of a domain of interest [19]. In our case, an ontology is defined in RDF(S) or DAML+OIL. Hence, an ontology is constituted by statements expressing definitions of DAML+OIL classes – RDF(S) resources, respectively – and properties ([17], [3]).
- **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, e.g. like `'http://www.aifb.uni-karlsruhe.de/WBS/sst/#Steffen'`. They frequently come with attribute instances, such as a human-readable label like 'Steffen'.

- **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 modelling 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 also contain statements like ‘Siegfried cooperates with Steffen’, *i.e.* relational metadata contain relationships between class instances rather than only textual notes.

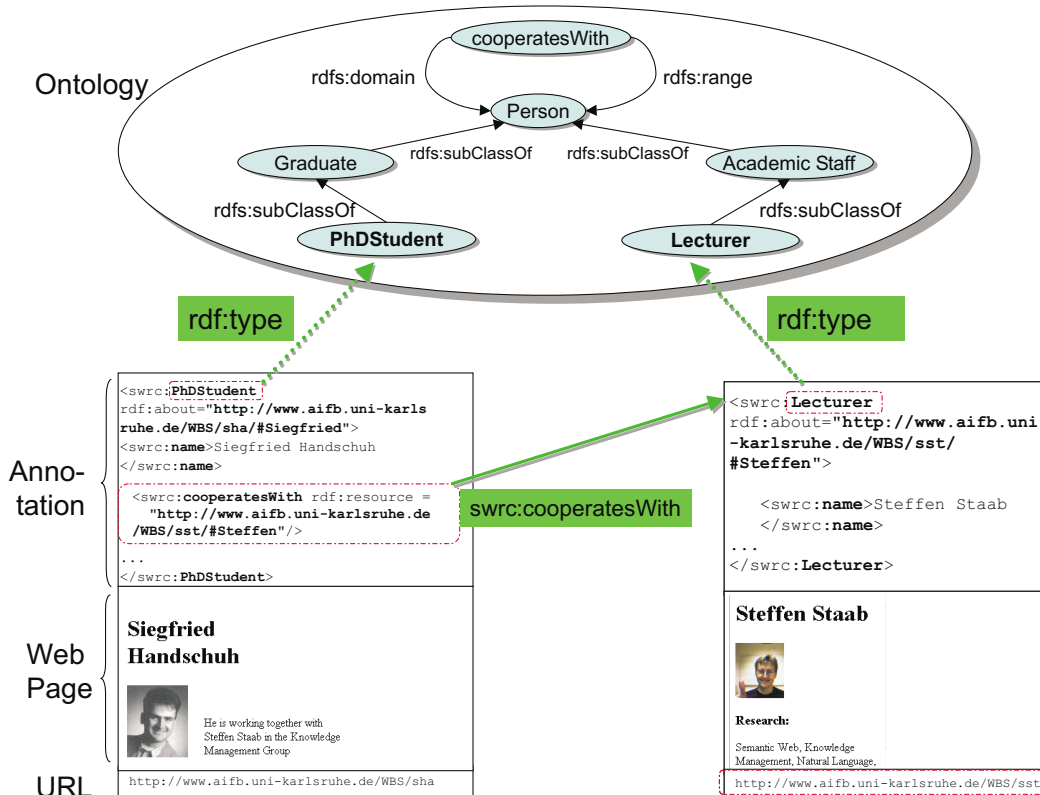


Fig. 1. Annotation example.

Figure 1 illustrates our use of the terms “ontology”, “annotation” and “relational metadata”. It depicts some part of the SWRC<sup>10</sup> (semantic web research community) ontology. Furthermore it shows two homepages, viz. pages about 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 (`http://www.aifb.uni-karlsruhe.de/`

<sup>10</sup> `http://ontobroker.semanticweb.org/ontos/swrc.html`

WBS/sst/#Steffen and <http://www.aifb.uni-karlsruhe.de/WBS/sha/#Siegfried>). The swrc:name of <http://www.aifb.uni-karlsruhe.de/WBS/sha/#Siegfried> is “Siegfried Handschuh”. In addition, there is a relationship instance between the two persons, *viz.* they cooperate. This cooperation information ‘spans’ the two pages.

The objective of CREAM is to allow for the easy generation of such a target representation irrespective of whether the major mode of interaction is a-posteriori annotation or web page authoring.

## 5 Design of CREAM

### 5.1 CREAM Modules

The requirements and considerations from Sections 1 to 4 feed into the design rationale of CREAM. The design rationale links the requirements with the CREAM modules. This results in a N:M mapping (neither functional nor injective). An overview of the matrix is given in Table 1, page 13.

- **Document Editor:** The document editor may be conceptually — though not practically — distinguished into a viewing component and the component for generating content:
  - **Viewer:** The document viewer visualizes the document contents. The metadata creator may easily provide new metadata by selecting pieces of text and aligning it with parts of the ontology. The document viewer should support various formats<sup>11</sup> (HTML, PDF, XML, etc.). For some formats the following component for content generation may not be available.
  - **Content Generation** The editor also allows the conventional authoring of documents. In addition, instances already available may be dragged from a visualization of the content of the annotation inference server and dropped into the document. Thereby, some piece of text and/or a link is produced taking into account the information from the meta ontology (cf. Section 6). The newly generated content is already annotated and the meta ontology guides the construction of further information, e.g. further XPointers are attached to instances.
- **Ontology Guidance and Fact Browser:** The 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 metadata

---

<sup>11</sup> The current prototypical implementation of CREAM focus on HTML/XHTML. A support for PDF is in development.

creators instantiate arbitrary classes and properties the semantics of these properties remains void. Of course the framework must be able to adapt to multiple ontologies in order to reflect different foci of the metadata creators. In the case of concurrent annotation with multiple ontologies there is an ontology guidance/fact browser for each ontology.

Furthermore, the ontology guidance and the browser for already given facts are important in order to guide metadata creators 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. [43]). Without the ontology they would miss even more cues for assigning relationships between class instances.

Both ontology guidance/fact browser and document editor/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 metadata creation, subjects must be aware of which entities exist already in their part of the Semantic Web. This is only possible if a crawler makes relevant entities immediately available. So, metadata creators 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 metadata creators) and only by this way metadata creators may recognize whether properties have already been instantiated (e.g. whether “Dieter Fensel” has already been linked to his publications). As a consequence of metadata creators’ awareness, relational metadata may be created, because class instances become related rather than only flat templates are filled.

A crawler answers the requirements of proper reference and the avoidance of redundancy. For example, one might crawl the homepages of his community before he starts to annotate his homepage in order to reuse and reference existing metadata. However, a crawler reduces and don’t solve the problems since it is obviously not possible to crawl the whole web.

We have built a RDF Crawler<sup>12</sup>, a basic tool that gathers interconnected fragments of RDF from the Web and builds a local knowledge base from this data (cf. [23] for a more detailed description).

- **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 created instances and on the ontology. It also serves the ontological guidance and fact browser, because it allows to query for existing classes, instances and

---

<sup>12</sup> RDF Crawler is freely available for download at:  
<http://ontobroker.semanticweb.org/rdfcrawler>.

properties.

The annotation inference server supports multiple ontologies. It distinguishes them into different namespaces. CREAM is based on the model, view, controller paradigm. The annotation inference server constitutes the model, whereas the ontology/fact browser constitutes the view. Every ontology in the annotation server may have a corresponding ontology/fact browser as a view.

We use Ontobroker's [7] underlying F-Logic [29] based inference engine SilRI [6] as annotation inference server. The F-Logic inference engine combines ordering-independent reasoning in a high-level logical language with a well-founded semantics including multiple namespaces [39]. However, other schemes like the DAML+OIL iFACT reasoner [26, 4] or a fact serving peer [36] might be exploited, too.

- **Document Management:** We distinguish two scenarios. First, when annotating one's own Web pages, one knows when they change and one can install organizational routines that keep track of annotation and changes of annotation (e.g. on the annotation inference server). In the second scenario, one may want to annotate external resources (cf., e.g., [37]). In this case one must 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 foreign web pages one has annotated together with their annotations. Foreign documents for which modification is not possible may be remotely annotated by using XPointer (cf. [9], [18]) as a addressing mechanism. When the foreign 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 on foreign web pages. When only few parts of a document change, pattern matching may propose revisions of old annotations. In our current implementation we use a straightforward file-system based document management approach. OntoMat 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, OntoMat 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.

- **Metadata Re-recognition & Information Extraction:** Even with sophisticated tools it is laborious to provide semantic annotations. A major goal thus is semi-automatic metadata creation taking advantage of information extraction techniques to propose annotations to metadata creators and, thus, to facilitate the metadata creation task. Concerning our environment we envisage three major techniques:

- (1) First, metadata re-recognition compares existing metadata literals with newly typed or existing text. Thus, the mentioning of the name “Siegfried Handschuh” in the document triggers the proposal that URI, `http://www.aifb.uni-karlsruhe.de/WBS/sha/#Siegfried` is co-referenced at this point.
- (2) “Wrappers” may be learned from given markup in order to automatically annotate similarly structured pages (cf., e.g., [31]).
- (3) Message extraction like systems may be used to recognize named entities, propose co-reference, and extract some relationship from texts (cf., e.g., [35, 48]).

This component has been realized by using the Amilcare information extraction system (cf. [22])<sup>13</sup>, but it is not yet available in the download version of OntoMat.

- **Meta Ontology:** The purpose of the meta ontology is the separation of ontology design and use. It is specifically explained in Section 6.

Besides 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. Alternatively or simultaneously it is also possible to store them in the annotation inference server.
- **Replication:** We provide a simple replication mechanism by crawling annotations into our annotation inference server. Then inferencing can be used to rule out formal inconsistencies.

---

<sup>13</sup> <http://www.dcs.shef.ac.uk/~fabio/Amilcare.html>

Table 1. Design Rationale — Linking Requirements with CREAM Modules.

Requirement	Document Editor		Replication		Storage			Metadata Re-cognition	General Information Extraction	Meta Ontology
	Viewer	Content Generation	Ontology Guidance	Crawler	Annotation Inference Server	Document Management				
Consistency		X	X		X					X
Proper Reference		X		X	X			X		X
Avoid Redundancy		X		X	X	X				
Relational Metadata			X	X	X		X	X		
Maintenance		X					X	X	X	
Ease of use	X							X	X	X
Efficiency	X	X	X	X		X		X	X	X
Multiple Ontologies			X			X				

## 5.2 Architecture of CREAM

The architecture of CREAM is depicted in Figure 2. The Design of the CREAM framework pursues the idea to be flexible and open. Therefore, OntoMat, the implementation of the framework, comprises a plug-in structure, which is flexible with regard to adding or replacing modules. Document viewer/editor and ontology guidance/fact browser together constitute the major part of the graphical user interface. Because of the plug-in structure they can be replaced by alternative viewers. For instance, we are currently working on a PDF viewer plugin capable of writing RDF into PDF documents exploiting Adobe's XMP framework.

Further capabilities are provided through plugins that establish connections, e.g. one might provide a plug-in for a connection to a commercial document management system.

The core OntoMat already comes with some basic functionalities. For instance, one may work without a plug-in for an annotation inference server, because the core OntoMat provides simple means to navigate the taxonomy *per se*. The core OntoMat, which is downloadable, consist an Ontology Guidance and Fact browser, a document viewer/editor, and a internal memory data-structure for the ontology and metadata. However, one only gets the full-fledged semantic capabilities (e.g. datalog reasoning or subsumption reasoning) when one uses a plug-in connection to a corresponding annotation inference server.

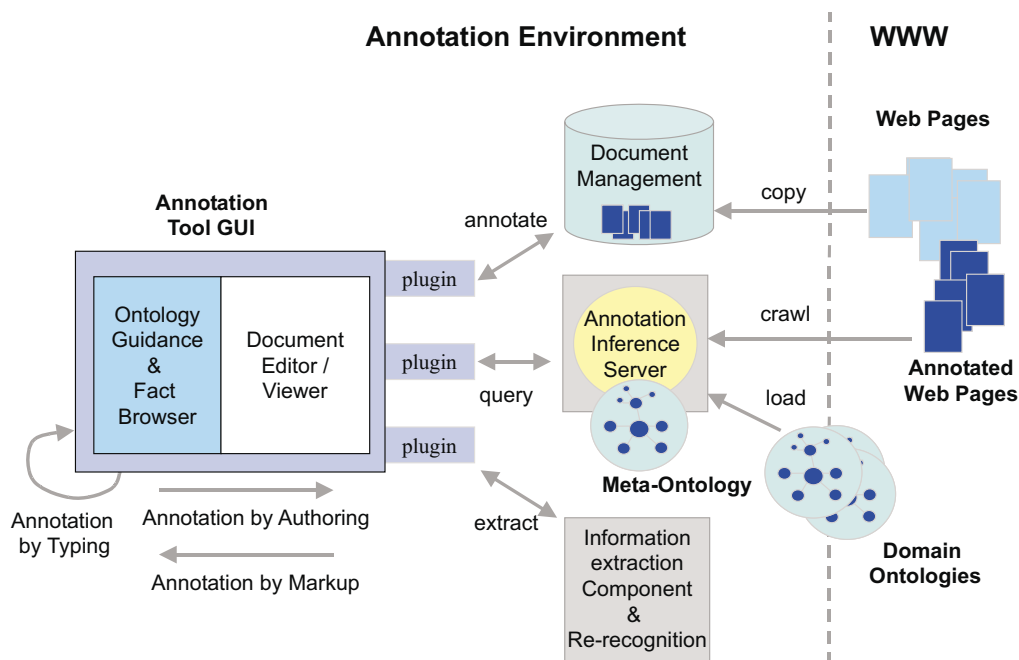


Fig. 2. Architecture of CREAM.

## 6 Meta Ontology

A meta ontology is needed to describe how classes, attributes and relationships from the domain ontology should be used by the CREAM environment. Hence, it describes what role the properties in the domain ontology have in the annotation system. In particular, we have recognized the urgent need for the meta ontology characterizations elaborated in Sections 6.1 to 6.3.

The meta ontology is given with the annotation system. The annotator must establish the connection of a ontology with the meta ontology. The effort for the annotator is reduced by default settings for this connection, such as the name attribute is by default the label of the instance. The task to connect the ontology with the meta-ontology is supported by the user interface, e.g. the annotator can change the role of the properties in the Ontology and Fact Browser.

Thus, the ontology describes how the semantic data should look like and the meta ontology connected to the ontology describes how the ontology is used by the annotation environment to actually create semantic data.

This modularization into ontology and meta ontology fulfills the requirement that it should be possible to define a new ontology or to reuse an existing one rather independently from the way it is used to create metadata by web page authoring and annotation.

The reader may note that the descriptions of how the meta ontology influences the interaction with the ontology, which are given in this section, depend to some extent on the modes of interaction described in Section 7 — and vice versa.

### 6.1 Label

The specification of RDFS [3] provides a `RDFS:LABEL` as a human-readable version of a resource name. Analogously, one wants to assign an instance with a human-readable name even if it instantiates a class from a given ontology that does not use the property `RDFS:LABEL` *per se*.

For instance, assume that (part of) the RDF(S) ontology definition is as follows:

```
<rdf:Property ID="ssn">
  <rdfs:comment> Social Security Number</rdfs:comment>
  <rdfs:range rdf:resource=
```

```

    "http://www.w3.org/2000/03/example/classes#Integer"/>
    <rdfs:domain rdf:resource="#Person"/>
</rdf:Property>
<rdf:Property ID="fullname">
  <rdfs:comment> Last Name, First Name, Middle Initial
</rdfs:comment>
  <rdfs:range rdf:resource=
    "http://www.w3.org/2000/01/rdf-schema#Literal"/>
  <rdfs:domain rdf:resource="#Person"/>
</rdf:Property>

```

Then, one might want to state that the property FULLNAME rather than the property SSN takes the role of RDFS:LABEL for class Person. We may link the meta ontology (the relevant piece here is RDFS:LABEL) with the ontology proper by:

```

<meta_onto:Label about="label1">
  <meta_onto:about_concept rdf:resource:=#Person>
  <meta_onto:about_attribut rdf:resource:=#fullname>
</meta_onto>

```

Now, the authoring and annotation environment may exploit this additional piece of information at (at least) two points of interaction:

- (1) Instance Generation: When a *new instance* is about to be created and some piece of text has been chosen to represent the name, CREAM creates a new URI and automatically assigns this piece of text to the attribute recorded as RDFS:LABEL, i.e. here FULLNAME.
- (2) Content Generation: When an instance is selected for generating content of a Web page, the generated text is produced by the RDFS:LABEL attribute. By this way we may produce text that is meaningful to humans with little interaction, because the author need not specify the attribute that should be used but he may just refer to the instance.

One may note that another person, e.g. from administration, could rather choose:

```

<meta_onto:Label about="label1">
  <meta_onto:about_concept rdf:resource:=#Person>
  <meta_onto:about_attribut rdf:resource:=#ssn>
</meta_onto>

```

Thus, this person would create web page content referring to social security numbers when authoring with existing instances and she would create new instances of PERSON from given social security numbers and not from names.

The reader may note from this scriptsize example that

- The difference between more or less metadata creation effort is often just one click.<sup>14</sup> For example, without the meta ontology description the following steps would have been necessary for instance creation: a instance creation with an appropriate label, filling the attribute "fullname", filling of the attribute "ssn".
- The connection between ontology and meta ontology is not an objective one. Rather their linkage depends on the way an ontology is used in a particular metadata creation scenario.

Statements equivalent to the last text passage hold for the following meta ontology descriptions.

## 6.2 Default Pointing

For many instances that are to be created it is desirable to point to the Web page from which they "originated". Analogously to the way that `RDFS:LABEL` is used, we use three types of definitions in order to specify the default pointing behavior for class instances, exploiting the XPointer working draft [9].

Consider the meta ontology properties `CREAM:UNIQUEDP POINTER`, `CREAM:AUTOD POINTER`, and `CREAM:AUTOUNIQUEDP POINTER`. If a property is `RDFS:SUBPROPERTYOF` one of the following interactions take place during annotation or authoring:

- (1) Instance Generation: When a new instance is generated and a property of that instance is of type `CREAM:AUTOD POINTER` or `CREAM:AUTOUNIQUEDP POINTER`, an XPointer to the current (part of the) web page will be automatically added into the corresponding slot of the instance.
- (2) Content Generation: When an instance is used for generating web page content, the attribute containing the XPointer is offered for link generation. When the attribute is of type `CREAM:UNIQUEDP POINTER` or `CREAM:AUTOUNIQUEDP POINTER` indicating uniqueness, a link with text corresponding to `RDFS:LABEL` and HRef corresponding to the XPointer will be automatically generated .

For instance, one may model in the ontology that a `PERSON` comes with properties `HASHHOMEPAGE` and `FULLNAME` and in the instantiation of the meta ontology that `HASHHOMEPAGE` is a subproperty of `CREAM:UNIQUEDP POINTER` and `FULLNAME` a subproperty of `RDFS:LABEL`. During annotation of people homepages, the label and pointer mechanisms automate, (i), the generation of unique IDs with reasonable labels, (ii), the creation of pointers to people's

---

<sup>14</sup> We conjecture that the one click difference may distinguish between success and failure of a tool.

homepages, and, *(iii)*, the correct linking between people mentioned on the different homepages. Like with RDFS:LABEL, the linkage between meta ontology and ontology proper may depend on the actual usage scenario.

### 6.3 Property Mode

The property mode distinguishes between different roles, which correspond to different ways the property should be treated by the metadata creation environment:

- (1) **Reference:** In order to describe an object, metadata may simply point to a particular place in a resource, e.g. a piece of text or a piece of multimedia. For instance, one may point to a particular place at `http://www.whitehouse.gov` in order to refer to the current U.S. president. Even when the presidency changes the metadata may remain up-to-date.

References are particularly apt to point to parts of multimedia, e.g. to a part of a scalable vector graphics.

The reader may note that every default pointer is a reference, but not vice versa. For example, assume a property HASNAME, which is a reference – it points to a particular place – to a name on the web page but not modeled as a default pointer, therefore the pointer is not automatically created or considered as unique.

- (2) **Quotation:** In order to describe an object, metadata may copy an excerpt out of a resource. In contrast to the mode “reference”, a quotation does not change when the corresponding resource changes. A copy of the string “Bill Clinton” as president of U.S. in 1999 remains unchanged even if its original source at `http://www.whitehouse.gov` changes or is abandoned.
- (3) **Unlinked Fact:** An unlinked fact describes an object, but is not in any way stemming or depending on a resource. Unlinked facts are typical for comments. They are also very apt to be combined with references in order to elucidate the meaning or name of a graphics or piece of multimedia.

For instance, there may be a reference pointing to the picture “Guernica” (`http://www.grnica.swinternet.co.uk/guernica.jpg`) and attributes that are specified to be unlinked facts. The unlinked fact-attributes may be filled by someone who knows Picasso’s paintings, e.g. with specifications like “Guernica” or “Spanish Civil War”.

The meaning of the property mode may slightly overlap with the definition of the range of a property, e.g. a unlinked fact is typically only used with an attribute that has a literal as its range. The reason is that a pointer may be used as a URI (e.g. [18]) and a URI should typically not appear in a literal (though this is not forbidden). We separate the two aspects, because not every

URI is a pointer and it sometimes makes sense to specify the value of a literal by a pointer. Thus, the definition of the range of a property as reference, quotation or unlinked fact may be considered orthogonal to the range of a property being a literal or a resource.

#### 6.4 Further Meta Ontology Descriptions

Concluding this section, we want the reader to note that the list of possibly useful meta ontology descriptions sketched here is not closed by far. Rather, we envision (and partially support) the use of meta ontology descriptions for purposes such as

- Knowledge acquisition from templates: For example we describe in SWO-BIS (<http://tools.semanticweb.org/>) software tools with metadata (cf. Figure 3). For each instance, there are a number of attributes required to specify a software tool. The meta ontology allows the definition of attribute instances being required attribute instances. This information is used to automatically generate a template like interface for OntoMat — one that is similar in its structure to a Dublin Core template. This approach is akin to the way that Protege allows to construct knowledge acquisition interfaces [38].
- Authoring of dynamic ontology and metadata-based Web pages (also cf. OntoWebber [27]).
- Provisioning of metametadata, e.g. author, date, time, and location of an annotation. Though this may appear trivial at first sight, this objective easily clashes with several other requirements, *e.g.* ease of use of metadata generation *and* usage. Eventually, it needs a rather elaborate meta ontology, containing not only static, but also dynamic definitions, *i.e.* rules. For example, to describe that a person A created an instance X, a second person B created an instance Y and a third person C created a relationships instance between X and Y, it is necessary to reify the relationship instance. In order to directly use the relationship instance, it should be translated by a rule into an unreified relationship instance.

## 7 Modes of Interaction with OntoMat

The metadata creation process in OntoMat is actually supported by three types of interaction with the tool (also cf. Figure 2):

- (1) Annotation by Typing Statements: This involves working almost exclusively within the ontology guidance/fact browser.

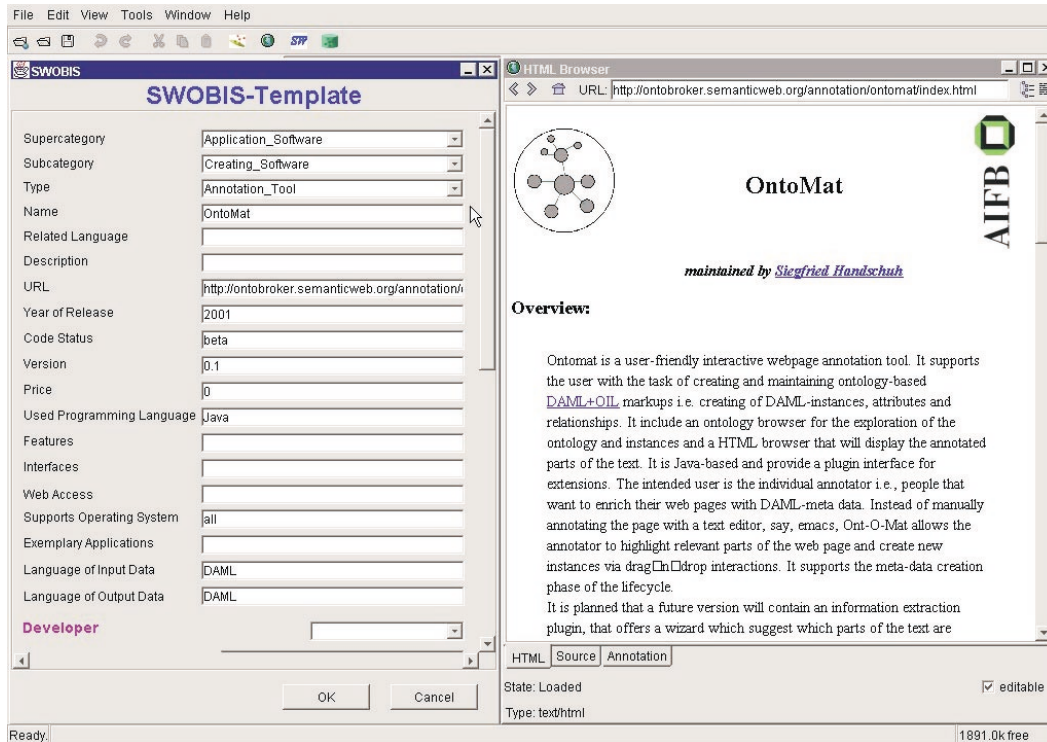


Fig. 3. SWOBIS template.

- (2) Annotation by Markup: This mostly involves the reuse of data from the document editor/viewer in the ontology guidance/fact browser.
- (3) Annotation by Authoring Web Pages: This mostly involves the reuse of data from the fact browser in the document editor.

In order to clarify the different role of the three types of interaction, we here describe how they differ for generating three types of metadata:

- (1) Generating instances of classes
- (2) Generating attribute instances
- (3) Generation relationship instances

### 7.1 Annotation by Typing

Annotation by typing is almost purely based on the ontology guidance/fact browser (cf. Section 5) and the generated templates (cf. Section 6.4). Basically, the more experienced user navigates the ontology and browses the facts, the less experienced user should rather use templates. The user generates metadata (class instances, attribute instances, relationship instances) that are completely independent from the Web page currently viewed.

The specification of the RDFS:LABEL property allows to create (or re-discover)

instances by typing where the URI is given and the `RDFS:LABEL` property is filled with the text. The specification of a default pointer by the meta ontology may associate newly created instances with the currently marked passage in the text.

In addition, the user may drag-and-drop around instances that are already in the knowledge base in order to create new relationship instances (cf. arrow #0 in Figure 4).

## 7.2 Annotation by Markup

The basic idea of annotation by markup is the usage of marked-up content in the document editor/viewer for instance generation.

- (1) Generating class instances: When the user drags a marked up piece of content onto a particular concept from the ontology, a new class instance is generated. If the class definition comes with a meta ontology description of a `RDFS:LABEL` a new URI is generated and the corresponding property is assigned the marked up text (cf. arrow #1 in Figure 4).

For instance, marking “Siegfried Handschuh” and dropping this piece of text on the concept `PHDSTUDENT` creates a new URI, instantiates this URI as belonging to `PHDSTUDENT` and assigns “Siegfried Handschuh” to the `SWRC:NAME` slot of the new URI. In addition, default pointers may be provided.

- (2) Generating attribute instance: In order to generate an attribute instance the user simply drops the marked up content into the corresponding table entry (cf. arrow #2 in Figure 4). Depending on whether the attribute is specified as reference or quotation the corresponding `XPointer` or the content itself is filled into the attribute.
- (3) Generating relationship instance: In order to generate a relationship instance the user simply drops the marked up content onto the relation of a pre-selected instance (cf. arrow #3 in Figure 4). Like in “class instance generation” a new instance is generated and connected with the pre-selected instance.

## 7.3 Annotation by Authoring

The third major process is authoring Web pages and metadata together. There are two modi for authoring: *(i)*, authoring by using ontology guidance and fact browser for content generation and, *(ii)*, authoring with the help of metadata re-recognition or — more general — information extraction. As far as authoring is concerned, we have only implemented *(i)* so far. However, we want to

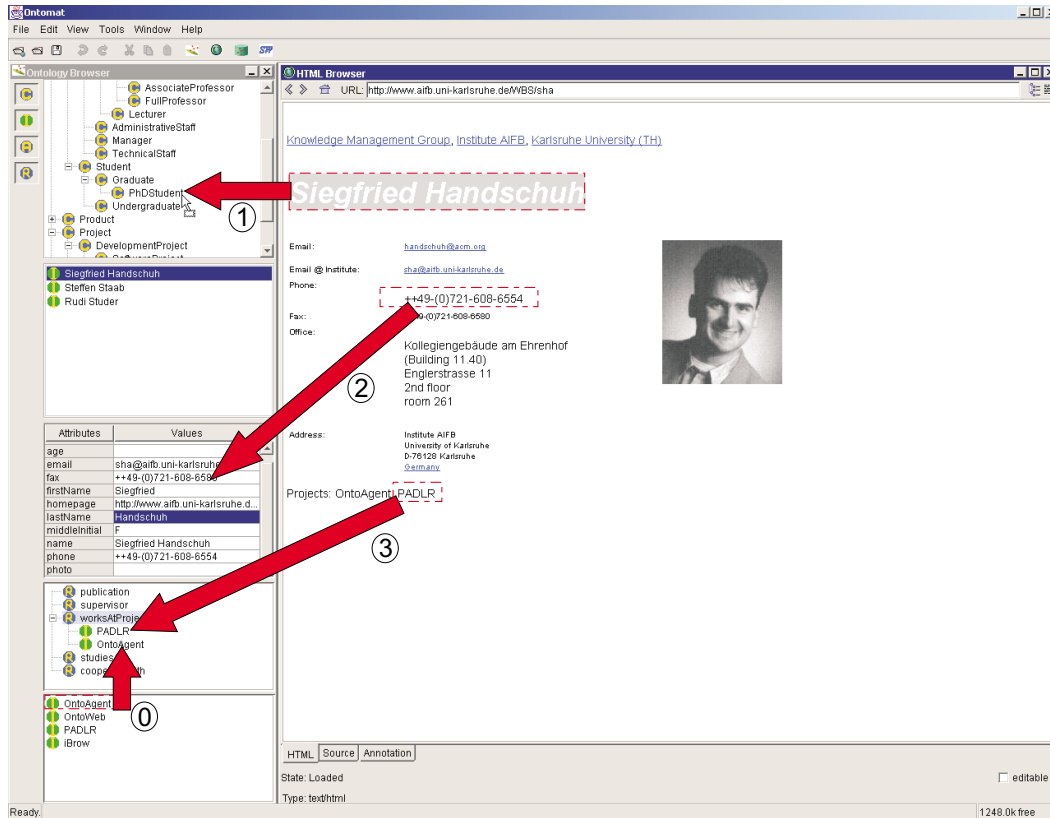


Fig. 4. Annotation example.

point out that already very simple information extraction mechanisms, i.e. metadata re-recognition (cf. Section 5) may help the author to produce consistent metadata.

**Authoring with Content Generation** By inverting the process of markup (cf. Figure 2), we may reuse existing instance description, like labels or other attributes:

- (1) Class instances: Dropping class instances from the fact browser into the document creates text according to their labels and — if possible — links (cf. arrow #1 in Figure 5).
- (2) Attribute instances: Dropping attribute instances from the fact browser in the document (cf. arrow #2 in Figure 5) generates the corresponding text (for quotations or unlinked facts) or even linked text (for references).
- (3) Relationship instances: Dropping relationship instances from the fact browser in the document generates simple “sentences”. For instance, the dropping of the relationship `COOPERATESWITH` between the instances corresponding to Rudi and Steffen triggers the creation of a small piece of text (cf. arrow #3 in Figure 5). The text corresponds to the instance labels plus the label of the relationship (if available), e.g. “Rudi Studer

cooperates with Steffen Staab”. Typically, this piece of text will require further editing.

Further mechanisms, like the creation of lists or tables from selected concepts (e.g. all PERSONS), still need to be explored.

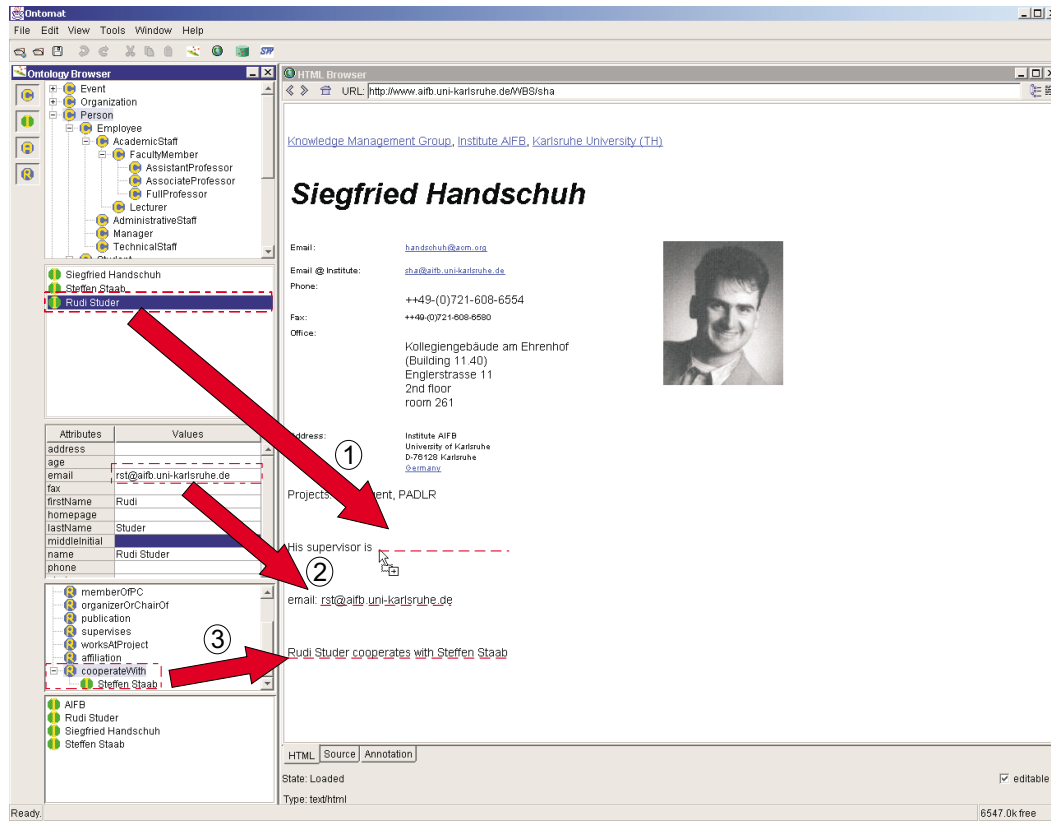


Fig. 5. Annotation by Authoring example.

## 8 Comparison with Related Work

CREAM can be compared along four dimensions: First, it is a framework for markup in the Semantic Web. Second, it may be considered as a particular knowledge acquisition framework that is to some extent similar to Protégé-2000 [13]. Third, it is certainly an annotation framework, though with a different focus than ones like Annotea [28]. And fourth it is an authoring framework with emphasis on metadata creation.

### 8.1 Knowledge Markup in the Semantic Web

We know of three major systems that intensively use knowledge markup in the Semantic Web, viz. SHOE [25], Ontobroker [7], and WebKB [34]. All three of them rely on markup in HTML pages. They all started with providing manual markup by editors. However, our experiences (cf. [12]) have shown that text-editing knowledge markup yields extremely poor results, viz. syntactic mistakes, improper references, and all the problems sketched in the scenario section.

The approaches from this line of research that are closest to *CREAM* is the SHOE Knowledge Annotator<sup>15</sup>, WebKB, and the MnM annotation tool.

The SHOE Knowledge Annotator is a Java program that allows users to mark-up web pages with the SHOE ontology. The SHOE system [33] defines additional tags that can be embedded in the body of HTML pages. The SHOE Knowledge Annotator is rather a little helper (like our earlier OntoPad [14], [7]) than a full fledged annotation environment.

WebKB [34] uses conceptual graphs for representing the semantic content of Web documents. It embeds conceptual graph statements into HTML pages. Essentially they offer a Web-based template like interface as knowledge acquisition frameworks described next.

The more recent development was the system from The Open University [32] and the corresponding MnM [47] annotation tool. MnM [47] also uses the Amilcare information extraction system. It allows the semi-automatic population of an ontology with metadata. So far, they have not dealt with relational metadata or authoring concerns.

### 8.2 Comparison with Knowledge Acquisition Frameworks

The CREAM framework allows for creating class and property instances and for populating HTML pages with them. Thus, it targets a roughly similar target like the instance acquisition phase in the Protégé-2000 framework [13] (the latter needs to be distinguished from the ontology editing capabilities of Protégé). The obvious difference between CREAM and Protégé is that the latter does not (and has not intended to) support the particular Web setting, viz. managing and displaying Web pages — not to mention Web page authoring. From Protégé we have adopted the principle of a meta ontology

---

<sup>15</sup> <http://www.cs.umd.edu/projects/plus/SHOE/KnowledgeAnnotator.html>

that allows to distinguish between different ways that classes and properties are treated.

We just named Protégé-2000 as one example for other existing knowledge acquisition frameworks.

### 8.3 Comparison with Annotation Frameworks

#### *Types of Annotations*

There are certain types of annotation. For example (i) the idea of creating a kind of user comment about web pages (cf. [28]), (ii) to annotate a document with further links (cf. section 2.2 in [1]) or (iii) semantic annotation. Also, we can distinguish the semantic annotation into instance creation or aboutness (cf. section 5 in [1]). Our viewpoint of annotation lies on instance creation (see section 4), but we can emulate most of the other annotation types with our framework.

- User comment: For instance, a user attach a note like "A interesting research topic!" to the term "semantic web" on a Web page. In CREAM we can design a corresponding ontology that allow to type the comment (an unlinked fact) "A interesting research topic" into an attribute instance belonging to an instance of the class COMMENT with a unique XPointer at "semantic web".
- Link annotation: A user attach a URL to a web page. In CREAM we can design a corresponding ontology that allow to insert the URL (as reference) into an attribute instance belonging to an instance of the class URL.
- Aboutness: A user express that a particular resource (i.e. a web page) is about a certain concept, but not an instance of an concept. In CREAM we can introduce a meta-concept into the corresponding ontology. The instances of this meta concept are all concepts of the ontology. Therefore the user interface supports the creation of an aboutness annotation. For instance, a user insert the URL of a page about students (e.g. <http://www.student.de>) (as a reference) to an attribute instance belonging to the instance of the class ABOUTSTATEMENT. Then he creates a relationship instance between the AboutStatement and the concept "student" of the ontology.

#### *Annotation Tools*

There are a lot of — even commercial — annotation tools like ThirdVoice<sup>16</sup>, Yawas [8], CritLink [49] and Annotea (Amaya) [28]. These tools all share

---

<sup>16</sup> <http://www.thirdvoice.com>

the idea of creating a kind of user comment about Web pages. The term “annotation” in these frameworks is understood as a remark to an existing document.

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.

#### *8.4 Comparison with Authoring Frameworks*

An approach related to the CREAM authoring is the Briefing Associate of Teknowledge [46]. The tool is an extension of Microsoft PowerPoint. It pursues the idea to produce PowerPoint documents with the metadata coding as a by-product of the document composition. For each concept and relation in the ontology, an instantiation button is added to the PowerPoint toolbar. Clicking on one of these buttons allows the author to insert an annotated graphical element into his briefing. Thus, a graphic element in the briefing corresponds to an instance of a concept and arrows between the elements correspond to relationship instances. In order to be able to use an ontology in PowerPoint one must have assigned graphic symbols to the concepts and relations, which is done in the beginning by the visual-annotation ontology editor (again a kind of meta ontology assignment). The Briefing Associate is available for PowerPoint documents. In contrast, CREAM does not provide graphic support like the Briefing Associate, but it supports both parts of the annotation process, i.e. it permits the simultaneous creation of documents and metadata and, in addition, the annotation of already existing documents. However, the Briefing Associate has shown very interesting ideas that may be of future value to CREAM.

The authoring of hypertexts and the authoring with concepts are topics in the COHSE project [18]. They allow for the automatic generation of metadata descriptions by analysing the content of a Web page and comparing the tokens with concept names described in a lexicon. They support ontology reasoning, but they do not support the creation of relational metadata. It is unclear to what extent COHSE considers the synchronous production of document and metadata by the author.

Somewhat similar to COHSE concerning the metadata generation, Klarity [30]

automatically fills Dublin Core fields taking advantage of statistic methods to allocate values based on the current document.

## 9 Conclusion

CREAM is a comprehensive framework for creating semantic metadata, relational metadata in particular — the foundation of the future Semantic Web. CREAM builds on comprehensive experience we have collected in several case studies on creating metadata for the Semantic Web. CREAM supports three modes of interaction for *a posteriori* annotation and for creating metadata while authoring a web page. In order to avoid problems with syntax, semantics and pragmatics, CREAM employs a rich set of modules including inference services, crawler, document management system, ontology guidance/fact browser, and document editors/viewers. Process issues of the annotation/authoring task are modularized from content descriptions by a meta ontology.

OntoMat is the reference implementation of the CREAM framework. It is Java-based and provides a plug-in interface for extensions for further applications. For instance, there exists a plug-in for information extraction that is coordinated with the OntoMat mode “annotation by markup” (cf. [22]). Because the plug-in structure is identical with the one for the collaborative Ontology Engineering environment, OntoEdit (cf. [45]), we expect that synergies like collaborative metadata creation or ontology editing and evolution integrated into the metadata creation process are rather easy to achieve.

We have also recently dealt with the issues of the annotation of dynamic web documents [24]. Other work at our institute deals with Ontology Evolution [44], which is also an aspect in the maintenance of the annotation.

Nevertheless, the general problem of metadata creation remains interesting.

Questions like “how do you annotate multimedia items?” or “what happens if there are 100,000 people known in your annotation inference server?” affect the scalability to more and larger dimensions, respectively. Some of them are currently approached by other research — as will be seen, e.g., in [21] — others such as “what is the appropriate methodology for annotation” wait for more research.

## 10 Acknowledgements

The research presented in this paper has profited from discussions with our colleagues at University of Karlsruhe, Stanford University and Ontoprise GmbH. In particular, we want to thank Stefan Decker (now: Information Science Institute, USC), Alexander Maedche (now: FZI Research Center for Information Technologies), Mika Maier-Collin (Ontoprise), Tanja Sollazzo (now: Siemens) and Sichun Xu (Stanford University). We also thank the reviewers for their comments, which helped to improve this paper.

Research for this paper was partially financed by US Air Force in the DARPA DAML project “OntoAgents” (01IN901C0), the BMBF founded project PADLR (Personalized Access to Distributed Learning Resources) and in the EU IST project BIZON IST-2001-33506.

## References

- [1] Sean Bechhofer, Leslie Carr, Carole Goble, Simon Kampa, and Timothy Miles-Board. The Semantics of Semantic Annotation. In *ODBASE: First International Conference on Ontologies, DataBases, and Applications of Semantics for Large Scale Information Systems*, Irvine, California, October 2002.
- [2] R. Benjamins, D. Fensel, and S. Decker. KA2: Building Ontologies for the Internet: A Midterm Report. *International Journal of Human Computer Studies*, 51(3):687–713, 1999.
- [3] D. Brickley and R.V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. Technical report, W3C, 2002. W3C Working Draft. <http://www.w3.org/TR/rdf-schema/>.
- [4] 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 WWW 2001*, pages 467–478. ACM Press, 2001.
- [5] S. Decker. *Semantic Web Methods for Knowledge Management*. PhD thesis, University of Karlsruhe, 2002.
- [6] 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), Boston, MA, December 3-4, 1998*. <http://www.w3.org/TandS/QL/QL98/>.
- [7] 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.
- [8] L. Denoue and L. Vignollet. An annotation tool for Web browsers and its applications to information retrieval. In *Pro-*

- ceedings of RIAO2000*, Paris, April 2000. <http://www.univ-savoie.fr/labos/syscom/Laurent.Denoue/riao2000.doc>.
- [9] S. DeRose, E. Maler, and R. Daniel. XML Pointer Language (XPointer) Version 1.0. Technical report, W3C, 2001. Candidate Recommendation 11 September 2001.
- [10] Dublin core metadata initiative, April 2001. <http://purl.oclc.org/dc/>.
- [11] Dublin Core Metadata Template, 2001. [http://www.ub2.lu.se/metadata/DC\\_creator.html](http://www.ub2.lu.se/metadata/DC_creator.html).
- [12] M. Erdmann, A. Maedche, H.-P. Schnurr, and S. 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.
- [13] H. Eriksson, R. Ferguson, Y. Shahar, and M. Musen. Automatic Generation of Ontology Editors. In *Proceedings of the 12th International Workshop on Knowledge Acquisition, Modelling and Mangement (KAW'99), Banff, Canada, October, 1999*.
- [14] D. Fensel, J. Angele, S. Decker, M. Erdmann, H.-P. Schnurr, S. Staab, R. Studer, and Andreas Witt. On2broker: Semantic-based access to information sources at the WWW. In *Proceedings of the World Conference on the WWW and Internet (WebNet 99), Honolulu, Hawaii, USA*, pages 366–371, 1999.
- [15] C. Fillies and F. Weichhardt. Graphische Entwicklung und Nutzung von Ontologien mit SemTalk in MS Office. In Robert Tolksdorf and Rainer Eckstein, editors, *XML Technologien für das Semantic Web — XSW 2002*, GI-Edition — Lecture Notes in Informatics (LNI), P-14. Bonner Köllen Verlag, 2002. Workshop 24. — 25. Juni 2002 in Berlin.
- [16] M. Frank, P. Szekely, and R. Neches a. Baoshi Yan a. Juan Lopez. Web-scripter: World-wide grass-roots ontology translation via implicit end-user alignment. In M. Frank, N. Noy, and S. Staab, editors, *Proceedings of the Semantic Web Workshop 2002 (at WWW-2002)*, CEUR Proceedings, Volume 55, <http://www.ceur-ws.org>, pages 22–28, 2002.
- [17] Reference description of the DAML+OIL (March 2001) ontology markup language, March 2001. <http://www.daml.org/2001/03/reference.html>.
- [18] C. Goble, S. Bechhofer, L. Carr, D. De Roure, and W. Hall. Conceptual Open Hypermedia = The Semantic Web? In S. Staab, S. Decker, D. Fensel, and A. Sheth, editors, *The Second International Workshop on the Semantic Web*, CEUR Proceedings, Volume 40, <http://www.ceur-ws.org>, pages 44–50, Hong Kong, May 2001.
- [19] T. R. Gruber. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 6(2):199–221, 1993.
- [20] S. Handschuh and S. Staab. Authoring and annotation of web pages in cream. In *Proceedings of the 11th International World Wide Web Conference, WWW 2002, Honolulu, Hawaii, May 7-11, 2002*, pages 462–473. ACM Press, 2002.

- [21] S. Handschuh and S. Staab, editors. *Annotation in the Semantic Web*. IOS Press, 2003.
- [22] S. Handschuh, S. Staab, and F. Ciravegna. S-CREAM — Semi-automatic CREAtion of Metadata. In *EKAW02, 13th International Conference on Knowledge Engineering and Knowledge Management*, LNCS/LNAI 2473, pages 358–372, Sigüenza, Spain, October 2002. Springer.
- [23] S. Handschuh, S. Staab, and A. Maedche. CREAM — Creating relational metadata with a component-based, ontology-driven annotation framework. In *Proceedings of K-Cap 2001*, pages 76–83. ACM Press, 2001.
- [24] S. Handschuh, S. Staab, and R. Volz. On Deep Annotation. Technical report, Institute AIFB, 2002.
- [25] 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.
- [26] I. Horrocks. Using an Expressive Description Logic: FaCT or Fiction? In *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98), Trento, Italy, June 2-5, 1998*, pages 636–649. Morgan Kaufmann, 1998.
- [27] Y. Jin, S. Decker, and G. Wiederhold. OntoWebber: Model-Driven Ontology-Based Web Site Management. In *Semantic Web Working Symposium (SWWS)*, Stanford, California, USA, August 2001.
- [28] J. Kahan, M. Koivunen, E. Prud'Hommeaux, and R. Swick. Annotea: An Open RDF Infrastructure for Shared Web Annotations. In *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001*, pages 623–632. ACM Press, 2001.
- [29] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42:741–843, 1995.
- [30] Klarity – automatic generation of metadata based on concepts within the document, 2001. Klarity white paper. <http://www.klarity.com.au>.
- [31] N. Kushmerick. Wrapper Induction: Efficiency and Expressiveness. *Artificial Intelligence*, 118(1-2):15–68, 2000.
- [32] Y. Lei, E. Motta, and J. Domingue. An Ontology-Driven Approach to Web Site Generation and Maintenance. In *EKAW02, 13th International Conference on Knowledge Engineering and Knowledge Management*, LNCS/LNAI 2473, Sigüenza, Spain, October 2002. Springer.
- [33] S. Luke, L. Spector, D. Rager, and J. Hendler. Ontology-based Web Agents. In *Proceedings of the First International Conference on Autonomous Agents, Marina del Rey, CA, USA, February 5-8, 1997*, pages 59–66, 1997.
- [34] 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.
- [35] *MUC-7 — Proceedings of the 7th Message Understanding Conference*, 1998. <http://www.muc.saic.com/>.
- [36] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson,

- M. Palmer, and T. Risch. EDUTELLA: A P2P Networking Infrastructure Based on RDF. In *Proceedings of WWW 2002*, pages 604–615. ACM Press, 2002.
- [37] W. Nejdl, B. Wolf, S. Staab, and J. Tane. EDUTELLA: Searching and Annotating Resources within an RDF-based P2P Network. Technical report, Learning Lab Lower Saxony / Institute AIFB, 2001.
- [38] N. Fridman Noy, W. E. Grosso, and M. A. Musen. Knowledge-Acquisition Interfaces for Domain Experts: An Empirical Evaluation of Protege-2000. In *Proceedings of the 12th Internal Conference on Software and Knowledge Engineering. Chicago, USA, July, 5-7, 2000*, 2000.
- [39] Ontoprise GmbH. How to write F-Logic programs. A tutorial for the language f-logic, tutorial version 1.7 (covers OntoBroker version 3.1). Technical report, 2002. [http://www.ontoprise.de/downloads/f\\_logic\\_tutorial.pdf](http://www.ontoprise.de/downloads/f_logic_tutorial.pdf).
- [40] T. Phelps and R. Wilensky. Robust intra-document locations. *Proceedings of WWW9 / Computer Networks*, 33(1-6):105–118, 2000.
- [41] S. Staab, J. Angele, S. Decker, M. Erdmann, A. Hotho, A. Maedche, H.-P. Schnurr, R. Studer, and Y. Sure. Semantic Community Web Portals. *Proceedings of WWW9 / Computer Networks*, 33(1-6):473–491, 2000.
- [42] S. Staab and A. Maedche. Knowledge Portals — Ontologies at work. *AI Magazine*, 22(2):63–75, Summer 2001.
- [43] 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.
- [44] L. Stojanovic, A. Maedche, B. Motik, and N. Stojanovic. User-driven Ontology Evolution Management. In *EKAW02, 13th International Conference on Knowledge Engineering and Knowledge Management*, LNCS/LNAI 2473, pages 285–300, Sigüenza, Spain, October 2002.
- [45] Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, and D. Wenke. Ontoedit: Collaborative ontology development for the semantic web. In *Proceedings of the 1st International Semantic Web Conference (ISWC2002), June 9-12th, 2002, Sardinia, Italia*, LNCS, pages 221–235. Springer, 2002.
- [46] M. Tallis, N. Goldman, and R. Balzer. The Briefing Associate: A Role for COTS applications in the Semantic Web. In *Semantic Web Working Symposium (SWWS)*, Stanford, California, USA, August 2001.
- [47] M. Vargas-Vera, E. Motta, J. Domingue, M. Lanzoni, A. Stutt, and F. Ciravegna. MnM: Ontology Driven Semi-Automatic and Automatic Support for Semantic Markup. In *EKAW02, 13th International Conference on Knowledge Engineering and Knowledge Management*, LNCS/LNAI 2473, pages 379–391, Sigüenza, Spain, October 2002. Springer.
- [48] M. Vargas-Vera, E. Motta, J. Domingue, S. Buckingham Shum, and M. Lanzoni. Knowledge Extraction by using an Ontology-based Annotation Tool. In *Proceedings of the Knowledge Markup and Semantic Annotation Workshop 2001 (at K-CAP 2001)*, pages 5–12, Victoria, BC,

Canada, October 2001.

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