Semantic Community Web Portals

S. Staab [A,O,1,2], J. Angele [O,3], S. Decker [A,O,2], M. Erdmann [A,2], A. Hotho [A,2], A. Maedche [A,2], H.-P. Schnurr [A,O,2], R. Studer [A,O,2], Y. Sure [A,2]

[A] *Institute for Applied Informatics and Formal Description Methods (AIFB), University of Karlsruhe, 76227 Karlsruhe, Germany* [4]

[O] *Ontoprise GmbH, Hermann-Loens-Weg 19, 76275 Ettlingen, Germany* [5]

Abstract

Community web portals serve as portals for the information needs of particular communities on the web. We here discuss how a comprehensive and flexible strategy for building and maintaining a high-value community web portal has been conceived and implemented. The strategy includes collaborative information provisioning by the community members. It is based on an ontology as a semantic backbone for accessing information on the portal, for contributing information, as well as for developing and maintaining the portal. We have also implemented a set of ontology-based tools that have facilitated the construction of our show case — the community web portal of the knowledge acquisition community.

Key words: Web Portal; Collaboration; Ontology; Web Site Management; Information Integration

1 Introduction

One of the major strengths of the web is that virtually everyone who owns a computer may contribute high-value information — the real challenge is to make valuable information be found. Obviously, this challenge can not only be

[1] Corresponding author.

[2] E-Mail: {staab, decker, erdmann, hotho, maedche, schnurr, studer, sure}@aifb.uni-karlsruhe.de

[3] E-Mail: angele@ontoprise.de

[4] http://www.aifb.uni-karlsruhe.de/WBS/

[5] http://www.ontoprise.de/

achieved by centralized services, since the coverage of even the most powerful crawling and indexing machines has shrunk in the last few years in terms of percentage of the number of web pages available on the web. This means that a proper solution to this dilemma should rather be sought along a principle paradigm of the WWW, *viz.* self-organization.

Self-organization does not necessarily mean automatic, machine-driven organization. Rather, from the very beginning communities of interest have formed on the web that covered what they deemed to be of interest to their group of users in — what we here call — *community web portals*. Community web portals are similar to Yahoo and its likes by their goal of presenting a structured view onto the web, however they are dissimilar by the way knowledge is provided in a collaborative process with only few resources (manpower, money) for maintaining and editing the portal. Another major distinction is that community web portals count in the millions, since a large percentage, if not the majority, of web or intranet sites is not maintained by a central department, but rather by a community of users. Strangely enough, technology for supporting communities of interest has not quite kept up with the complexity of the tasks of managing community web portals. A few years ago such a community of interest would have comparatively few sources of information to consider. Hence, the overall complexity of managing this task was low. Now, with so many more people participating a community portal of only modest size may easily reach the point where it appears to be more of a jungle of interest rather than a convenient portal to start from.

This problem gave us reason to reconsider the techniques for managing community web portals. We observed that a successful web portal would weave loose pieces of information into a coherent presentation adequate for sharing knowledge with the user. On the conceptual, knowledge sharing, level we have found that Davenport and Prusak's maxime [6], "people can't share knowledge if they don't speak a common language", is utterly crucial for the case of community web portals. The only difference to Davenport and Prusak's thoughts derives from the fact that knowledge need not only be shared between people, but also between people and machines.

At this point, ontologies and intelligent reasoning come in as key technologies that allow knowledge sharing at a conceptually concise and elaborate level. These AI techniques support core concerns of the "Semantic Web" (*cf.* Berners & Lee [3]). In this view, information on the web is not restricted to HTML only, but information may also be formal and, thus, machine understandable. The combination may be accounted for by an explicit model of knowledge structures in an *ontology*. The ontology formally represents common knowledge and interests that people share within their community. It is used to support the major tasks of a portal, *viz. accessing* the portal through manifold, dynamic, conceptually plausible views onto the information of interest in a particular community, and *providing* information in a number of ways that reflect different types of information resources held by the individuals.

Following these principles when putting the Semantic Community Web Portal into practice, incurs a range of subtasks that must be accounted for and requires a set of tools that support the accomplishment of these tasks. In Section 2 we discuss requirements that we have derived from a particular application scenario, the KA2 portal, that also serves as our testbed for development. Section 3 describes how ontologies are created and used for structuring information and, thus, appears as the conceptual cornerstone of our community web portal. We proceed with the actual application of ontologies for the purposes of accessing the KA2 portal by navigating and querying explicit and implicit information through conceptual views on and rules in the ontology (Section 4). Section 5 covers the information provisioning part for the community web portal considering problems like information gathering and integration. Then, we describe the engineering process for our approach (Section 6) and present the overall architecture of our system (Section 7). Before we conclude with a tie-up of experiences and further work, we compare our work with related approaches (Section 8).

2 Requirements for a Community Web Portal - The KA2 Example

Examples for community web portals abound. In fact, one finds portals that very well succeed regarding some of the requirements we describe in this section. For instance, MathNet (http://www.math-net.de/) introduces knowledge sharing through a database relying on Dublin Core metadata. Another example, RiboWeb [1], offers means to navigate a knowledge base about ribosomes. However, these approaches lack an integrated concept covering all phases of a community web portal, *viz.* information accessing, information providing, and portal development and maintenance. We pursue a system that goes beyond isolated components towards a comprehensive solution for managing community web portals.

2.1 Portal Access by Users

Navigating through a community web portal that is unknown is a rather difficult task in general. Information retrieval may facilitate the finding of pieces of texts, but its usage is not sufficient in order to provide novice users with the right means for exploring unknown terrain. This turns out to be a problem particularly when the user does not know much about the domain and does not know what terms to search for. In such cases it is usually more helpful for the user to *explore* the portal by browsing — given that the portal is well and comprehensively structured. Simple tree-structured portals may be easy to maintain, but the chance is extremely high that an inexperienced user looking for information gets stuck in a dead-end road. Here, we must face the trade-off between resources used for structuring the portal (money, man-power) and the extent to which a comprehensive navigation structure may be provided. Since information in the community portal will be continually amended by users, richly interrelated presentation of information would usually require extensive editing, such as is done, *e.g.*, for Yahoo. In contrast, most community web portals require that comprehensive structuring of information for presentation comes virtually for free.

There has been interesting research (*e.g.* Fröhlich et al. [13] or Kesseler [15]) that demonstrates that authoring, as well as reading and understanding of web sites, profits from conceptual models underlying document structures "in the large", *i.e.* the interlinking between documents, as well as document structures "in the small", *i.e.*, the contents of a particular document. Rather naturally, once a common conceptual model for the community exists and is made explicit, it is easier for the individual to access a particular site. Hence, in addition to rich interlinking between document structures "in the large", comprehensive surveys and indices of contents and a large number of different views onto the contents of the portal, we require that the semantic structure of the portal is made explicit at some point. The following sections will show that this stipulation does not raise a conflict with other requirements, but that it fits well with the requirements that arise from provisioning of information and maintenance of the portal. Section 4 will elaborate on how such a conceptual level is exploited for a complex web site with extensive browsing and querying capabilities.

2.2 Information Provisioning through Community Members

An essential feature of a community web portal is the contribution of information from all (or at least many) members of the community. Though, they share some common understanding of their community, the information they want to contribute comes in many different (legacy) formats. Still, presentations of and queries for information contents must be allowed in many ways that need to be rather independent from the way by that information was provided originally. The web portal must remain adaptable to the information sources contributed by its members — and not vice versa. This requirement precludes the application of database-oriented approaches (*e.g.*, [21]), since they presume that a uniform mode of storage exists that allows for the structuring of information at a particular conceptual level, such as a relational database scheme. In real-world settings, one must neither assume that a uniform mode for information storage exists nor that only one particular conceptual level is adequate for structuring information of a particular community. In fact, even more sophisticated approaches such as XML-based techniques that separate content from layout and allow for multiple modes of presentation appear insufficient, because their underlying transformation mechanisms (*e.g.*, XSLT or XQL [24], [9]) are too inconvenient for integration and presentation of various formats at different conceptual levels. The reason is that they do not provide the semantic underpinning required for proper integration of information.

In order to integrate diverse information, we require another layer besides the common distinction into document content and layout, *viz.* explicit knowledge structures that may structure *all* the information in different formats for a community at various levels of granularity. Different information formats need to be captured and related to the common ontology:

(1) Several types of metadata such as available on web pages (*e.g.* , HTML META-tags),
(2) manual provision of data to the knowledge base, and
(3) a range of different wrappers that encapsulate structured and semi-structured information sources (*e.g.*, databases or HTML documents).

Section 5 will address exactly these issues. The question now remains as to *how* this kind of Semantic Community Web Portal is put into practice.

2.3 Development and Maintenance

A community web portal as we have stipulated constitutes a complex system. Hence, the developers and editors will need comprehensive tool support for presenting contents through views and links, and for maintaining consistency in the system, as well as guidelines that describe the procedures for actually building such a portal. Indeed, some of our first experiences with the example portal described in Section 2.4 was that even users who were well acquainted with all the principles hated to acquire detailed, technical knowledge in order to provide information or maintain “their” information in the portal. Thus, we need a comprehensive concept that integrates tools and methods for building the portal, capturing information, and presenting its contents to the community. While some of the tools will be touched upon in subsequent sections, development and maintenance issues in general will be dealt with in Section 6.

2.4 The Example

The example that we draw from in the rest of this paper is the portal for the *"Knowledge Annotation initiative of the Knowledge Acquisition community"* (KA2; *cf.* [2]). The KA2 initiative has been conceived for semantic knowledge retrieval from the web building on knowledge created in the KA community. To structure knowledge, an ontology has been built in an international collaboration of researchers. The ontology constitutes the basis to annotate WWW documents of the knowledge acquisition community in order to enable intelligent access to these documents and to infer implicit knowledge from explicitly stated facts and rules from the ontology. Though KA2 has provided much of the background knowledge we now want to exploit, it lacked much of the ease for accessing and providing community knowledge that we aim at with the KA2 community web portal.

Given this basic scenario, which may be easily transfered towards other settings for community web portals, we have investigated the techniques and built the tools that we describe in the rest of this paper. Nevertheless the reader may note that we have not yet achieved a complete integration of all tools and neither have we exploited all our technical capabilities in our up and running demonstration portal (http://ka2portal.aifb.uni-karlsruhe.de).

3 Structuring the Community Web

Let us now summarize the principal stipulations we have found so far. We need

- a conceptual structure for presenting information to the user,
- support for integrating information from different granularities stored in various formats,
- comprehensive tool support for providing information, developing and maintaining the portal, and
- a methodology for implementing the portal.

In particular, we need an explicit structuring mechanism that pervades the portal and reaches from development and maintenance, over provisioning to presentation of information. For this purpose, we use an ontology as the conceptual backbone of our community web portal.

3.1 The Role of Ontologies

An *ontology* is an explicit specification of a shared conceptualization [14]. Their usefulness for information presentation (*e.g.* [13]), information integration (*e.g.* [30]) and system development (*e.g.* [2]) has been demonstrated recently. We introduce another application area, *viz.* the use of ontologies for intranet management or, more specific in our example, for community portal management. The role of ontologies is the capturing of domain knowledge in a generic way and the provision of a commonly agreed understanding of a domain, which may be reused and shared within communities or applications. Though only few communities have explicitly modeled their knowledge structures yet (examples are UMLS [26], Altmann et al. [1]), practically all share a common understanding of their particular domain.

Hence, our strategy is to use an ontology as a backbone of our community web portal. In fact, we even allow the usage of multiple views that reflect diverging standards of understanding and usage of terminology in different subcommunities or for different groups of users (*e.g.*, novice vs. expert). Rules may then be used to translate between different views such that one may view the information contributed from another subcommunity.

3.2 Modelling

The KA2 ontology consists of (i) concepts defining and structuring important terms, (ii) their attributes specifying properties and relations, and (iii) rules allowing inferences and the generation of new knowledge. Our representation language for defining the ontology is F-Logic [16], which provides adequate modeling primitives integrated into a logical framework.

To illustrate the structure of the ontology, the screenshot in Figure 1 depicts part of the KA2 ontology as it is seen in the ontology development environment *OntoEdit* (*cf.* Section 6.2.1). The leftmost window shows the is-a-relationship that structures the concepts of the domain in a (possibly multiple) taxonomy. Attributes and relations of concepts are inherited by subconcepts. Multiple inheritance is allowed as a concept may fit into different branches of the taxonomy. In Figure 1, attributes and relations of the concept Researcher appear in the middle window. Some of these attributes, like FIRSTNAME and LASTNAME are inherited from the superordinate concept Person. Relations refer to other concepts, like WORKSATPROJECT denoting a relation between Researcher and Project.

We use the KA2 ontology to manage and structure the community portal. The structure of concepts with its attributes and relations supports user navigation

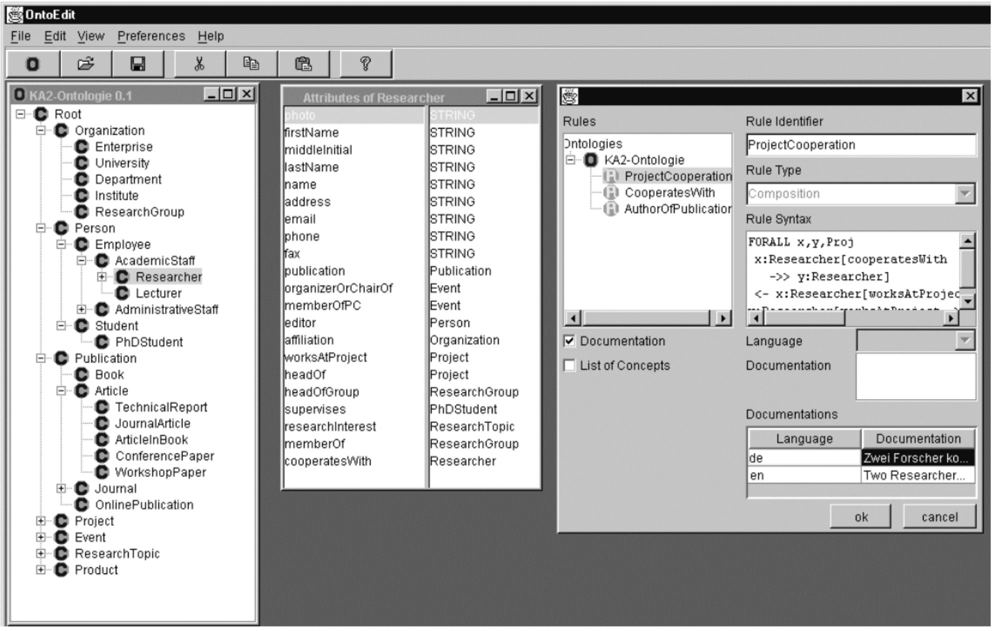


Figure 1. Part of the KA2 Ontology in an OntoEdit-Screenshot

through the domain as detailed in Section 4. Beyond simple structuring we model rules allowing inferencing and by that means the generation of new knowledge. The rightmost window in Figure 1 shows some rules of our KA2 ontology. One of them, the *ProjectCooperation* rule, describes the cooperation of two researchers working in the same project. The rule states that, two researchers cooperate, if a Researcher X works at a Project $Proj$ and if a Researcher Y works at the same Project $Proj$ and X is another person than Y. The rule is formulated in an F-Logic representation:

(1) $\texttt{FORALL}\ X,\ Y,\ Proj$
X:Researcher [$\textsc{cooperatesWith} \twoheadrightarrow Y$:Researcher]
$\leftarrow$
X:Researcher [$\textsc{worksAtProject} \twoheadrightarrow Proj$:Project]
$\texttt{AND}\ Y$:Researcher [$\textsc{worksAtProject} \twoheadrightarrow Proj$:Project]
$\texttt{AND NOT equal}(X,\ Y)$.

With this rule, we may infer, that researcher X cooperates with researcher Y due to the fact that they work in the same project — even, if there is no *explicit* fact, that they cooperate.

4 Accessing the Community Web Portal

A major requirement from Section 2 has been that navigation and querying of the community web portal need to be conceptually founded, because only then a structured, richly interwoven presentation may be compiled on the fly. In fact, we elaborate in this section how a semantic underpinning, like the KA2 ontology described above, lets us define a multitude of views that dynamically arrange information. Thus, our system may provide the rich interlinking that is most adequate for the individual user and her navigation and querying of the community web portal that we have aimed at in the beginning. We start with a description of the query capabilities of our representation framework. The framework builds on the very same F-Logic mechanism for querying as it did for ontology representation and, thus, it may also exploit the ontological background knowledge. Through this semantic level we achieve the independence from the original, syntactically proprietary, information sources that we stipulated earlier. Nevertheless, F-Logic is as poorly suited for presentation to naive users as any other query language. Hence, its use is mostly disguised in various easy-to-use mechanisms that more properly serve the needs of the common user (*cf.* Section 4.2), while it still gives the editor all the power of the principal F-Logic representation and query capabilities. Finally in this section, we touch upon some very mission-critical issues of the actual inference engine that answers queries and derives new facts by combining facts with structures and rules from the ontology.

4.1 Query Capabilities

Though information may be provided in a number of different formats our underlying language for representation and querying is F-Logic. For instance, using a concrete example from our showcase the following query asks for all publications of the researcher "Studer".

(2) `FORALL` Pub
$\leftarrow$
`EXISTS` $ResID$ $ResID$:Researcher [LASTNAME $\twoheadrightarrow$ "Studer";
PUBLICATION $\twoheadrightarrow$ Pub].

The substitutions for the variable *Pub* constitute the publications queried by this expression. The expressiveness and usability of such queries is improved by the possibility to use a simple form of information retrieval using regular expressions whithin queries. For instance the following query asks for abstracts that contain the word "portal":

(3) FORALL $Abstr$
$\leftarrow$
EXISTS Pub, X
Pub:Publication [ABSTRACT $\twoheadrightarrow$ $Abstr$]
AND regexp (“[p|P]ortal”, $Abstr$, X).

The substitutions for the variable $Abstr$ are the abstracts of publications which contain the word “portal”.

In addition, the query capabilities allow to make implicit information explicit. They use the background knowledge expressed in the KA2 ontology including rules as introduced in Section 3.2. If we have a look at web pages about research projects, information about the researchers (*e.g.* their names, their affiliation, ...) involved in the projects is often explicitly stated in HTML. However, the fact that researchers who are working together in projects are cooperating is not explicitly stated. A question might be: “Which researchers are cooperating with other researchers?” Querying for cooperating researchers the implicit information about project cooperation of researchers is exploited. The query may be formulated by:

(4) FORALL $ResID1$, $ResID2$
$\leftarrow$
$ResID1$:Researcher [COOPERATESWITH $\twoheadrightarrow$ $ResID2$].

The result set includes explicit information about a researchers cooperation relationships, which are stored in the *knowledge warehouse*, and also implicit information about project cooperation between researchers derived using the project-cooperation rule modeled in the ontology.

4.2 Navigating and Querying the Portal

Usually, it is too inconvenient for users to query the portal using F-Logic. Therefore we offer a range of techniques that allow for navigating and querying the community web:

- A *hypertext link* may contain a query which is dynamically evaluated when one clicks on the link. Browsing is made possible through the definition of views onto top-level concepts of the KA2 ontology, such as Persons, Projects, Organizations, Publications, Research Topics and Events. Each of these topics can be browsed using predefined views. For example, a click on the Projects hyperlink results in a query for all projects known at the portal. The query is evaluated and the results are presented to the user in a table.
- A choice of concepts, instances, or combinations of both may be issued to the user in *HTML forms*. Choice options may be selected through check boxes,

selection lists, or radio buttons. For instance, clicking on the Projects link (*cf.* upper part of Figure 2) an F-Logic query is evaluated and all projects contained in the portal are retrieved. The results can be restricted using topic-specific attributes contained in the KA2 ontology for projects, such as topics of a project, people involved etc. The selection list (*e.g.* for all people involved in projects) is generated dynamically from the information contained in the knowledge warehouse (*cf.* Section 5.4). Using the form's contents a query may be compiled and evaluated.

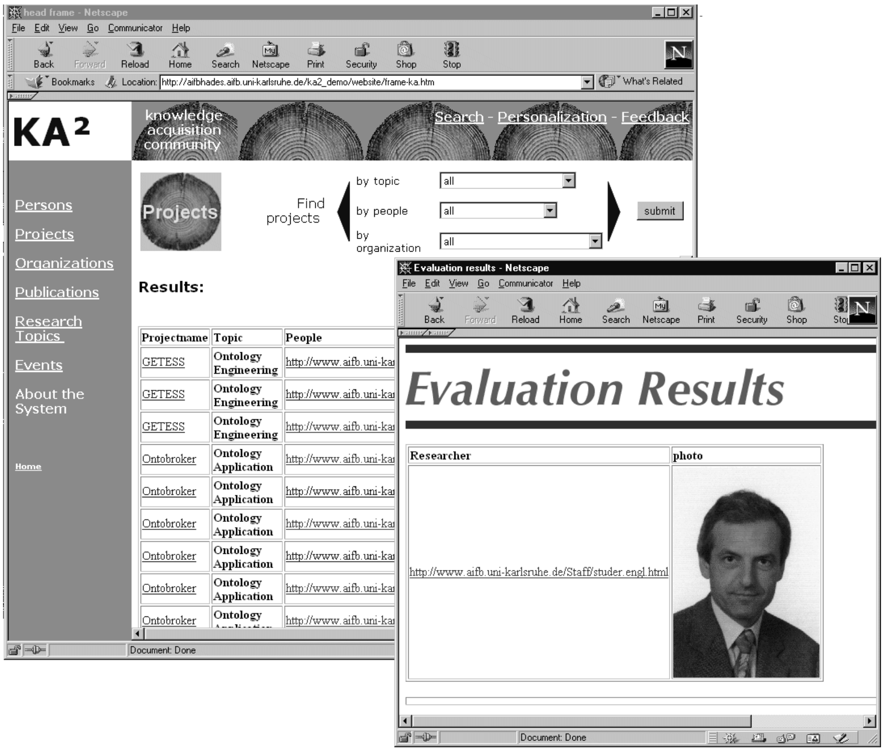


Figure 2. Accessing the Community Web Portal

- A query may also be generated by using the hyperbolic view interface (*cf.* Figure 3). The hyperbolic view visualizes the ontology as a hierarchy of concepts. The presentation is based on hyperbolic geometry (*cf.* [18]) where nodes in the center are depicted with a large circle, whereas nodes at the border of the surrounding circle are only marked with a small circle. This visualization technique allows a survey over all concepts, a quick navigation to nodes far away from the center, as well as a closer examination of nodes and their vicinity. When a user selects a node from the hyperbolic view, a form is presented which allows the user to select attributes or to insert values for the attributes. An example is shown in Figure 3. The user is searching for the community member "Studer" and his photo. Based on the selected node and the corresponding attributes, a query is compiled. The query-result is shown in the right part of Figure 2.
- Furthermore, queries created by the hyperbolic view interface may be stored

using the personalization feature. Queries are personalized for the different users and are available for the user in a selection list. The stored queries can be considered as *semantic bookmarks*. By selecting a previously created bookmark, the underlying query is evaluated and the updated results are presented to the user. By this way, every user may create a personalized view on the portal.

- Finally, we offer an expert mode. The most technical (but also most powerful and flexible) way for querying the portal requires that F-Logic is typed in by the user. This way is only appropriate for users who are very familiar with F-Logic and the KA2 ontology.

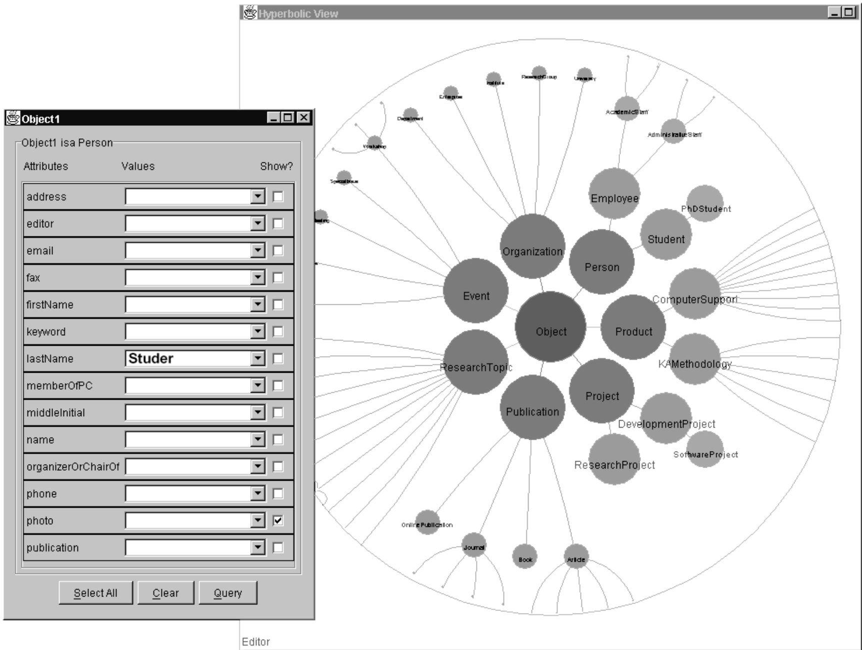


Figure 3. Hyperbolic View Interface

4.3 The Inference Engine

The inference engine answers queries and it performs derivations of new knowledge by an intelligent combination of facts with an ontology denoted in F-Logic like the examples described above. While the expressiveness of F-Logic and its Java-powered realization in our inference engine constitute two major arguments for using it in a semantic community web portal, wide acceptance of a service like this also depends on *prima facie* unexciting features like speed of service. The principal problem we encounter here is that there exist worst case situations (not always recognizable as such by the user) where a very large set of facts must be derived by the inference engine in order to solve a particular query. While we cannot guarantee for extremely fast response times, unless we drastically cut back on the expressiveness of our representation formalism, we provide several strategies to cope with performance problems:

- The inference engine may be configured to subsequently deliver answers to the query instead of waiting for the entire set of answers before these answers are presented to the user. Thus, answers that are directly available as facts may be presented immediately while other anwers that have to be derived using rules are presented later.
- The inference engine caches all facts and intermediate facts derived from earlier queries. Thus, similar queries or queries that build on previously derived facts may be answered fast.
- Finally, we allow the inference engine to be split into several inference engines that execute in parallel. Every engine may run on a different processor or even a different computer. Every inference engine administers a subset of the rules and facts. A master engine coordinates user queries and distributes subqueries to the slave engines. These slave engines either answer these subqueries directly or distribute incoming subqueries to other inference engines.

The reader may note that though we have provided all the technical means to pursue one or several of these strategies, our showcase has not reached yet the amount of facts that really necessitates any performance enhancing strategies.

5 Providing Information

"One method fits all" does not meet the requirements we have sketched above for the information provisioning part of community web portals. What one rather needs is a set of methods and tools that may account for the diversity of information sources of potential interest to the community portal. While these methods and tools need to obey different syntactic mechanisms, coherent integration of information is only possible with a conceptual basis that may sort loose pieces of information into a well-defined knowledge warehouse. In our setting, the conceptual basis is given through the ontology that provides the background knowledge and that supports the presentation of information by semantic, *i.e.* rule-enhanced, F-Logic queries. Talking about the syntactic and/or interface side, we support three major, different, modes of information provisioning: First, we handle *metadata-based information sources* that explicitly describe contents of documents on a semantic basis. Second, we align regularities found in documents or data structures with the corresponding semantic background knowledge in *wrapper-based* approaches. Thus, we may create a common conceptual denominator for previously unrelated pieces of information. Finally, we allow the direct provisioning of facts through our *fact editor*. All the information is brought together in a knowledge warehouse that stores data and metadata alike. Thus, it mediates between the original information sources and the navigating and querying needs discussed in the previous section.

5.1 Metadata-based Information

Metadata-based information enriches documents with semantic information by explicitly adding metadata to the information sources. Over the last years several metadata languages have been proposed which can be used to annotate information sources. In our approach the specified ontology constitutes the conceptual backbone for the different syntactic mechanisms.

Current web standards for representing metadata like RDF [28] or XML [27] can be handled within our semantic web portal approach. On the one hand, RDF facts serve as direct input for the knowledge warehouse, on the other hand, RDF facts can be generated from information contained in the portal knowledge warehouse. We have developed *SiLRI* (Simple Logic-based RDF Interpreter), a logic-based inference engine implemented in Java that can draw inferences based on the RDF data model (*cf.* Decker et al. [7]).

XML provides the chance to get metadata for free, *i.e.* as a side product of defining the document structure. For this reason, we have developed a method and a tool called *DTDMaker* for generating DTDs out of ontologies [10]. DTDMaker derives an XML document type definition from a given ontology in F-Logic, so that XML instances can be linked to an ontology. The linkage has the advantage that the document structure is grounded on a true semantic basis and, thus, facts from XML documents may be directly integrated into the knowledge warehouse.

HTML-A, early proposed by Fensel et al. [11], is an HTML extension which adds annotations to HTML documents using an ontology as a metadata schema. HTML-A has the advantage to smoothly integrate semantic annotations into HTML and prevents the duplication of information. An example is an HTML page that states that the text string “Rudi Studer” is the name of a researcher where the URL of his homepage is used as his object identifier. Using HTML-A this could be realized by:

```
<HTML>
  <BODY>
    <A onto="page:Researcher"/>
    <H1> HomePage of
      <A onto="page[firstName=body]">
       Rudi</A>
      <A onto="page[lastName=body]">
```

```
        Studer</A>
      </H1>
      ...
    </BODY>
  </HTML>
```

The keyword page refers to the webpage that contains the ontological markup. The first annotation denotes an object of type Reseacher that represents the homepage of Rudi Studer. Subsequent annotations define the firstName and the lastName attributes of this object by relating to the values from the body of the corresponding anchor-tags. To facilitate the annotation of HTML, we have developed an HTML-A annotation tool called *OntoPad*. An example annotation of an email of the Researcher Rudi Studer using OntoPad is illustrated in Figure 4. Similarly to HTML-A, it is possible to enrich documents generated with Microsoft Office applications with metadata by using our plug-ins *Word-A* and *Excel-A*.

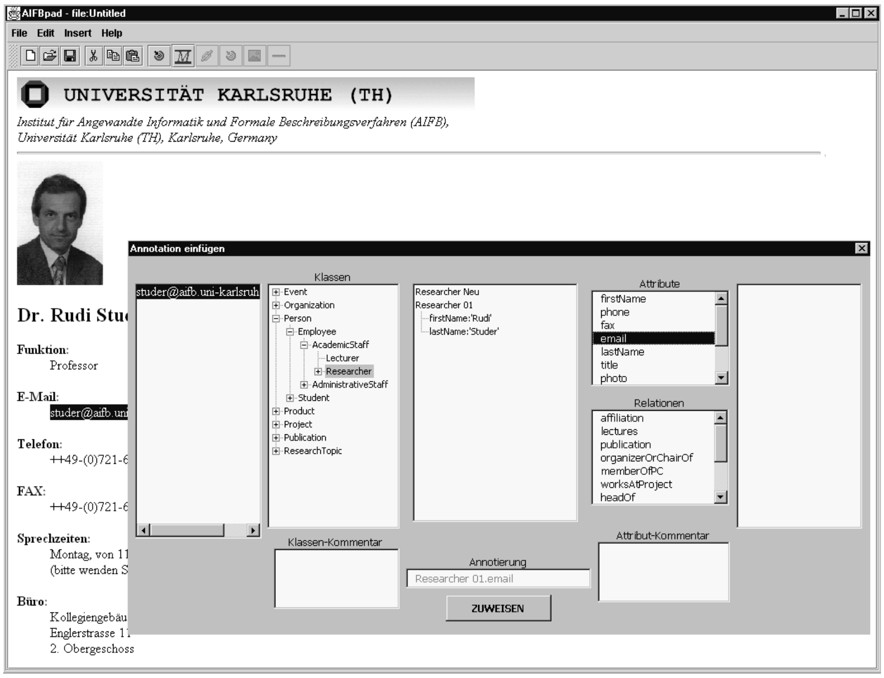


Figure 4. OntoPad - Providing Semantics in HTML Documents using HTML-A

5.2 Wrapper-based Information

In general, annotating information sources by hand is a time consuming task. Often, however, annotation may be automated when one findes regularities in a larger number of documents. The principle idea behind wrapper-based information is that there are large information collections that have a similar structure. We here distinguish between semi-structured information sources (*e.g.* HTML) and structured information sources (*e.g.* relational databases).

5.2.1 Semi-structured Sources

In recent years several approaches have been proposed for wrapping semi-structured documents, such as HTML documents. Wrapper factories (*cf.* Sahuguet et al. [25]) and wrapper induction (*cf.* Kushmerick [17]) have considerably facilitated the task of wrapper construction. In order to wrap directly into our knowledge warehouse we are currently developing our own wrapper approach that directly alignes regularities in semi-structured documents with their corresponding ontological meaning.

5.2.2 Structured Sources

Though, in the KA2 community web there are no existing information systems, we would like to emphasize that existing databases and other legacy-systems may contain valuable information for building a community web portal. Ontologies have shown their usefulness in the area of intelligent database integration. They act as information mediators (*cf.* Wiederhold & Genesereth [30]) between distributed and heterogeneous information sources and the applications that use these information sources. Existing entities in legacy systems are mapped onto concepts and attributes defined in the ontology. Thus, existing information may be pumped into the knowledge warehouse by a batch process or it may be accessed on the fly.

5.3 Fact Editor

The process of providing new facts into the knowledge warehouse should be as easy as possible. For this reason we offer the hyperbolic interface tool (*cf.* Figure 3) which may be used as a *Fact Editor*. In this mode its forms are not used to ask for values, but to insert values for attributes of instances of corresponding concepts from the ontology. The Fact Editor is also used for maintaining the portal, *viz.* to add, modify, or delete facts.

5.4 Knowledge Warehouse

The different methods and tools we have just described feed directly into the knowledge warehouse or indirectly when they are triggered by a web crawl. The warehouse itself hosts the ontology, *i.e.* the metadata level, as well as the data proper. The knowledge warehouse is indirectly accessed, through a user query or a query by an inference engine such as described in Section 4. Hence, one may take full advantage of the distribution capabilities of the inference engine and, likewise, separate the knowledge warehouse into several knowledge bases or knowledge marts. Facts and concepts are stored in a relational database, however, they are stored in a *reified* format that treats relations and concepts as first-order objects and that is therefore very flexible with regard to changes and amendments of the ontology.

6 Development of Web Portals

6.1 The Development and Maintenance Process

Even with the methodological and tool support we have described so far, developing a web portal for a community of non-trivial size remains a complex task. Strictly ad-hoc rapid prototyping approaches easily doom the construction to fail or they easily lead up to unsatisfactory results. Hence, we have thought about a more principled approach towards the development process that serves as means for documenting development, as well as for communicating principal structures to co-developers and editors of the web portal. We distinguish different phases in the development process that are illustrated in Figure 5. For the main part this model is a sequential one. Nevertheless, at each stage there is an evaluation as to whether and as to how easily further development may proceed with the design decisions that have been accomplished before. The results feed back into the results of earlier stages of development. In fact, experiences gained by running the operational system often find their way back into the system.

The main stages of the development process and their key characteristics are given in the following:

- The process starts with the *elicitation* of user requirements in the requirements elicitation phase. In this phase, requirements about important and interesting topics in the domain are collected, the information goals of potential users of the portal are elicited, and preferences or expectations concerning the structure and layout of presented information is documented. Results of this very first phase constitute the input for the design of the web site and for preliminary HTML pages and affect the formal domain model embodied in the ontology.
- The requirements determine, *e.g.*, which views and queries are useful for users of the portal, which navigation paths they expect, how different web pages are linked, or which functionality is provided in different areas of the

portal. Requirements like these are realized in the *web site design*. This design phase may be performed independently to a very large extent from the underlying formal structuring, *i.e.* the ontology. Since a mock-up version of the web site is developed early in the development phase, one may check early whether the system to be developed really meets the users' needs.

- In parallel to the development of the structure and layout of the web site an *ontology engineering* process is started. The first phase elicits relevant *domain terms* that need to be refined and amended in the ontology engineering phase. First, the static ontology parts, *i.e.* the concept hierarchy, the attributes, and relations between concepts are formally defined. Thereafter, *rules* and constraints are developed. Rule development may necessitate a major revision of the concept hierarchy. For instance, new sub-concepts may have to be introduced, attributes may have to turn into relations or into other concepts, or relations may have to become concepts. This (intra-ontology) engineering cycle must be performed until the resulting ontology remains sufficiently stable.
- In the *query formulation* step the views and queries described in one of the earlier phases are formalized. At first, their functionality is tested independently from the web site design. To express the information needs formally, the developer has to access the ontology, whereby additional rules or relations that define new views or ease the definition of queries may become necessary. In order to test ontology and queries, a set of test facts has to be prepared. During this process of testing, inconsistencies in the ontology may be detected, which may lead to a feed back-loop back into the ontology engineering phase.
- Finally, web pages are *populated*, *i.e.* the queries and views developed during website design, and formalized and tested during query formalization are integrated into the operational portal. Information may be accessed via the portal as soon as a sufficient amount has been made available as outlined in Section 5.

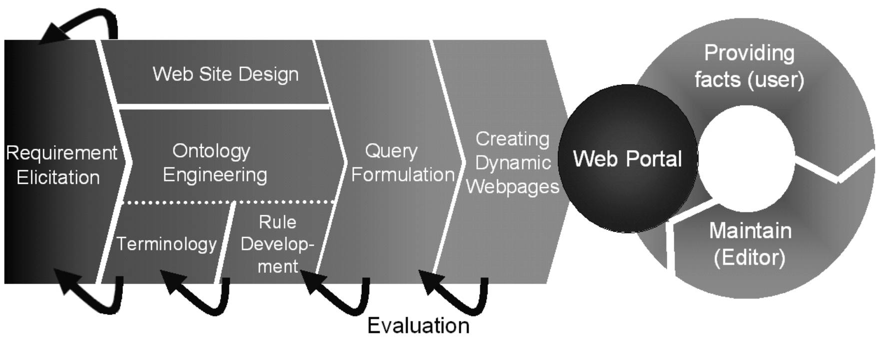


Figure 5. The Development Process of the Community Web Portal

During operation of the community portal it must be *fed* and maintained:

- The user community provides facts via numerous input channels (*cf.* Section 5).
- These facts may contain errors or undesired contents, or the integration of different sources may lead to inconsistencies. To counter problems like these, an editor is responsible to detect these cases and act appropriately. The detection of inconsistencies is supported by the inference engine via constraints formulated in F-Logic. The editor then has to decide how to proceed. He may contact responsible authors, simply ignore conflicting information, or he may manually edit the contents.
- Changing requirements of the community must be reflected in the portal, *e.g.* popularity increasing in new fields of interests or technologies or viewpoints that shift may incur changes to the ontology, new queries, or even a new web site structure. In order to meet such new requirements, the above mentioned development process may have to be partially restarted.

6.2 Tools for Development and Maintenance

The previous subsection has described the principal steps for developing a community web portal. For efficient development of a community web portal, however, the process must be supported by tools. In the following, we describe the most important tools that allow us to facilitate and speed up the development and maintenance process. The tools cover the whole range from ontology engineering (OntoEdit), query formulation (Query Builder), up to the creation of dynamic web pages with the help of HTML/JavaScript templates. We just want to note here that we also rely on common HTML editing tools for web site design.

6.2.1 OntoEdit.

The OntoEdit toolset has already been mentioned in Section 3.2 and a screenshot is shown in Figure 1. OntoEdit is used during terminology engineering and rule development. It delivers a wide range of functionalities for the engineering of ontologies. As introduced in Section 3, ontology modeling — from our point of view — includes the creation of concepts, attributes, relations, rules, and general metadata. To reduce complexity and to simplify the difficult task of ontology modeling, OntoEdit offers different views on the main modeling elements, thus facilitating the complex process of ontology modeling. The modeling task is usally started with introducing new concepts and organizing them into a hierarchy (*cf.* left part of Figure 1). The next step of the modeling task uses the concepts to model attributes of and relations between concepts. On the basis of these knowledge structures, OntoEdit includes a rule view enabling the user to model rules which state common legalities (*e.g.* the symmetry of the cooperation relationship between two persons).

6.2.2 Query Builder

While queries with low complexity can be expressed in F-Logic using the rule debugger alone, in other cases it is more convenient to create queries using our Query Builder tool. The hyperbolic view (*cf.* Section 4) interface may be configured for this tool to allow the user to generate queries instead of posing queries. Such queries may then be integrated as links into a web page with the help of a common web page editor by copying and pasting it into the web editors form.

6.2.3 HTML/JavaScript Templates

Another time consuming activity is the development of the web pages that assemble queries from parts and that display the query results. For that purpose we have developed a *library of template pages*:

- Templates with check boxes, radio boxes, and selection lists are available. These HTML forms produce data that is used to generate queries. These queries can then be sent to the inference engine using submit buttons.
- The results of a query are fed into a template page as Javascript arrays. From these data different presentation forms may be generated [6]:
  - · A general purpose template contains a table that presents answer tuples returned by the inference engine in a HTML table. The template provides functions to sort the table in ascending or descending order on different columns. Substitutions of certain variables may be used as URLs for other entries in the table. Different data formats are recognized and processed according to their suffixes, *i.e.* a ".gif" or ".jpg" sufix is interpreted as a picture and rendered as such (*cf.* Figure 2 for an example).
  - · Results may also be fed into selection lists, radio boxes, or check lists. Thus, query results can provide the initial setting of further HTML form fields. These forms can be used to create new queries based on the results of previous ones.
- A user can create queries using the hyperbolic view interface (*cf.* Section 4). As a personalization feature of our web portal he can store these queries by assigning a personal label. All stored queries can be accessed through a selection list, to restart the query and retrieve the most up to date answers. This list of stored queries provides individual short cuts to often needed information.

[6] For the future, we envision that XML is returned by the inference engine and outlayed by XSL style sheets.

The template pages are compatible with some standard web editors, *i.e.* the web designer is able to rearrange or redesign the elements without destroying their functionality.

7 The System Architecture

This section summarizes the major components of our system. An overall view of our system is depicted in Figure 6, which includes the core modules for accessing and maintaining a community web portal:

- *Providing Information* in our community web portal has already been introduced in Section 5. In our approach we distinguish between metadata-based, wrapper-based and fact-based information. Metadata-based information (such as HTML-A, Word-A, Excel-A, RDF, XML) is collected from the web using a fact crawler. Wrapper-based information means integrating semi-structured and structured information semi-automatically into the knowledge warehouse. Using the *Fact Editor* factual information can be directly added to the knowledge warehouse.
- The *Knowledge Warehouse* is the knowledge base of the community web portal. It is structured according to the ontology. The facts contained in the knowledge warehouse and the ontology itself serve as the input for the inference engine.
- The *Inference Engine* uses information from the knowledge warehouse to answer queries. In addition, it uses ontological structures and rules to derive additional factual knowledge that is only implicitly provided. This inference mechanism may also be used to reduce the maintenance efforts, because facts that may be automatically derived from other facts need not be provided by some member of the community.
- *Accessing* the community web portal means navigating and querying for information as described in Section 4. Queries embedded in the portal or formulated using the hyperbolic view interface (*cf.* Figure 3) are posted to the inference engine. The results may be delivered in different forms like HTML, XML, or RDF.

8 Related Work

This section positions our work in the context of existing web portals like Yahoo and Netscape and also relates our work to other technologies that are or could be deployed for the construction of community web portals.

One of the well-established web portals on the web is Yahoo , a manually main-

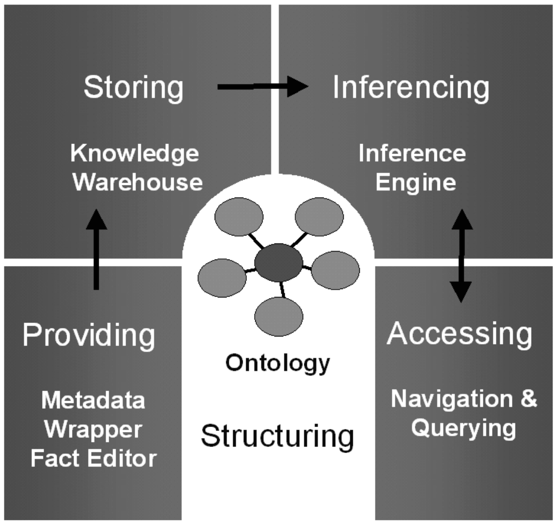


Figure 6. System Architecture

tained web index (http://www.yahoo.com). Yahoo allows information seekers to retrieve web documents by navigating a tree-like taxonomy of topics. Each web document indexed by Yahoo is classified manually according to a topic in the taxonomy. In contrast to our approach Yahoo only utilizes a very light-weight ontology that solely consists of categories arranged in a hierarchical manner. Yahoo offers keyword search (local to a selected topic or global) in addition to hierarchical navigation but is only able to retrieve complete documents, *i.e.* it is not able to answer queries concerning the contents of documents, not to mention to present or combine facts being found in different documents. Due to its weak ontology Yahoo cannot extend information to include facts that could be derived through ontological axioms. The mentioned points are realized in our portal that builds upon a rich ontology enabling the portal to give detailed and integrated answers to queries. Furthermore, our portal supports the active provision of information by the user community. Thus we get rid of the manual and centralized classification of documents. The portal is made by the community for the community.

A portal that is specialized for a scientific community has been built by the Math-Net project, an initiative for "setting up the technical and organizational infrastructure for efficient, inexpensive and user driven information services for mathematics" [5]. At http://www.math-net.de/ the portal for the (german) mathematics community is installed that makes distributed information from several mathematical departments available. The scope of offered information ranges from publications such as preprints and reports to information about research projects, organizations, faculty etc. All this data is accompanied by meta-data according to the Dublin Core (http://www.purl.org/dc)

Standard [29] that makes it comparatively easy to provide structured views onto the stored information. The 15 elements of Dublin Core primarily describe meta-data about a resource, *e.g.* its title, its author, or its format. The Dublin Core element "Subject" is used to classify resources as students, as conferences, as research groups, as preprints etc. A finer classification (*e.g.* via attributes) is not possible except for instances of the publication category. Here the common MSC-Classification (Mathematical Subject Classification; http://www.ams.org/msc/) is used that resembles an ontology of the field of mathematics. The pros of Math-Net lie in the "important organizational task" of "establishing a network of persons [...] dedicating their time and work to the electronic information system" [5] and in the technical task of collecting various resources and integrating them to make them uniformly accessible. The cons of Math-Net are the lack of a rich ontology that could enhance the quality of search results (esp. via inferencing), and the restriction to information in the DC format.

Parts of Math-Net are implemented on the basis of the Hyperwave system [21], an elaborated web server that is based on databases providing information in a structured manner. Hyperwave has a lot of fancy and useful features such as external links, automatic handling of outdated pages, avoiding of dangling links, presenting different views for different users, etc. The system is a useful basis for the development of a portal, but since it does not have the notion of an ontology, the portal is restricted to the power of the underlying database. Its capabilities clearly stay behind the inferential properties of Ontobroker's inference engine.

Another community-focused portal is RiboWeb [1], an ontology-based data resource of published ribosome data and computational models that can process this data. RiboWeb exploits several types of ontologies to provide semantics for all the data and computational models that are offered. These ontologies are specified in the OKBC (Open Knowledge Base Connectivity) representation language [4]. The primary source of data is given by published scientific literature which is manually linked to the different ontologies. Both systems, RiboWeb and our community portal, rely on ontologies for offering a semantic-based access to the stored data. However, the OKBC knowledge base component of RiboWeb does not support the kind of automatic deduction that is offered by the inference engine of Ontobroker. Furthermore, RiboWeb does not include wrappers for automatically extracting information from the given published articles. On the other hand, the computational modules of RiboWeb offer processing functionalities that are not part of (but also not intended for) our community web portal.

The Ontobroker project [8] lays the technological foundations for the KA2 portal. On top of Ontobroker the portal has been built and organizational structures for developing and maintaining it have been established. Therefore, we compare our system against approaches that are similar to Ontobroker.

The approach closest to Ontobroker is SHOE [19]. In SHOE, HTML pages are annotated via ontologies to support information retrieval based on semantic information. Besides the use of ontologies and the annotation of web pages the underlying philosophy of both systems differs significantly: In SHOE, arbitrary extensions to ontologies can be introduced on web pages and no central provider index is maintained. As a consequence, when specifying a query, users can not know all the ontological terms which have been used and the web crawler will miss annotated web pages. In contrast, Ontobroker relies on the notion of a community defining a group of web users who share a common understanding and, thus, can agree on an ontology for a given field. Therefore, both the information providers and the clients have complete knowledge of the available ontological terms, a prerequisite for building a community web portal. SHOE and Ontobroker also differ with respect to their inferencing capabilities. SHOE uses description logic as its basic representation formalism, but it offers only very limited inferencing capabilities. Ontobroker relies on Frame-Logic and supports complex inferencing for query answering.

WebKB [20] aims at providing intelligent access to Web documents. WebKB uses conceptual graphs for representing the semantic content of Web documents. It embeds conceptual graph statements into HTML tags to provide meta data about the contents of HTML documents. These statements are based on an ontology defining the concepts and relations which may be used for annotating the HTML documents. WebKB pursues a rather similar approach when compared to our backbone system Ontobroker: both systems use a general representation language for representing meta data (conceptual graphs and Frame logic, respectively) and embed meta data within the HTML source code. However, Ontobroker (and thus our portal) provides additional means for accessing non-HTML resources, *e.g.* exploiting XML, RDF metadata, or using ontology-based wrappers. Furthermore, the tool environment of WebKB does not offer the methods and tools that are needed to build a community portal on top of WebKB and, thus, to make an application out of a core technology.

The STRUDEL system [12] applies concepts from database management systems to the process of building Web sites. STRUDEL uses a mediator architecture to generate such Web sites. Wrappers transform the external data sources, being either HTML pages, structured files or relational databases, into the semi-structured data model used within STRUDEL's data repository. STRUDEL then uses so-called "site-definition queries" to create multiple views to the same Web site data. These queries are defined in STRUQL, a query language for manipulating semi-structured data. When compared to our approach, the STRUDEL system lacks the semantic level that is provided in our approach by the domain ontology and the associated inference engine.

The Observer-System [23] uses a network of ontologies to provide access to distributed and heterogeneous information. Each ontology can describe the information contained in one or more information repositories. Since these ontologies are linked explicitly by so called inter-ontology relationships (synonym, hypernym, and hyponym), the information stored in the different resources are linked as well. A user selects an ontology (a vocabulary) to express his query. The Observer system accesses the resources described by the selected ontology to answer the query. If the answers are not satisfactory other ontologies and thus, other resources can be accessed. Observer provides means to state queries but does not offer predefined or customizable views.

From our point of view, our community portal system is rather unique with respect to the collection of methods used and the functionality provided. Our approach for accessing information, providing information and maintaining the portal are more comprehensive than those found in other portals. We are able to offer this functionality since our backbone system Ontobroker and its add-ons provide more powerful techniques for *e.g.* inferencing or extracting information from various sources than those offered by comparable systems. Moreover, all these methods are integrated into one uniform system environment.

9 Conclusion

We have demonstrated in this paper how a community may build a community web portal. The portal is centered around an ontology that structures information for the purposes of presentation and provisioning of information, as well as for the development and maintenance of the portal. We have described a small scale example, the KA2 portal, that illustrates some of our techniques and methods. In particular, we have developed a set of ontology-based tools that allow to present multiple views onto the same information appropriate for browsing, querying, and personalizing web pages. Adding information in the up and running portal is possible for members of the community or the editors of the portal through a set of methods that support the integration of metadata and (semi-)structured information sources, as well as the manual manipulation of data. The tools that support development of the portal and maintenance by the editors are also ontology-based and, thus, fit together smoothly with the overall process of running the community portal.

The concepts that we have explained are general enough to apply to many other domains. As a second application, we have constructed an intranet management application for an IT service company, the rationale being that the people in a company just form a community with common interests.

For the future, we will need to tackle some very practical issues, like improving and integrating the interfaces of our tools, adding a component for session management, or versioning of views and ontologies. A major experience so far has been that community members are only willing to contribute information if this is very easy to accomplish. We could not provide this quality of service from the start. Hence our knowledge warehouse is still rather small, but large enough to show the viability of our approach. Besides fancy interfaces, the most urgent needs will be the improvement of provisioning. There are some machine learning approaches (*e.g.* [22]) that may help with dedicated subtasks of the information provisioning process. Our general experience, however, is that there is no single tool or technique that fits all needs, but there is a conceptual strategy that ties up all the loose ends, *viz.* ontologies.

References

[1] R.B. Altmann, M. Bada, X.J. Chai, M. Whirl Carillo, R.O. Chen, and N.F. Abernethy. RiboWeb: An Ontology-based System for Collaborative Molecular Biology. *IEEE Intelligent Systems*, 14(5):68–76, September/October 1999.

[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, 1999.

[3] T. Berners-Lee. *Weaving the Web*. Harper, 1999.

[4] V. Chaudri, A. Farquhar, R. Fikes, P. Karp, and J. Rice. OKBC: A Programmatic Foundation for Knowledge Base Interoperability. In *Proceedings 15th National Conference on Artificial Intelligence (AAAI-98)*, pages 600–607, 1998.

[5] W. Dalitz, M. Grötschel, and J. Lügger. Information Services for Mathematics in the Internet (Math-Net). In A. Sydow, editor, *Proceedings of the 15th IMACS World Congress on Scientific Computation: Modelling and Applied Mathematics*, volume 4 of *Artificial Intelligence and Computer Science*, pages 773–778. Wissenschaft und Technik Verlag, 1997.

[6] T. Davenport and L. Prusak. *Working Knowledge: How organizations manage what they know*. Harvard Business School Press, 1998.

[7] 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.

[8] 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.

[9] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, and D. Suciu. A Query Language for XML. In *Proceedings of the 8th Int. World Wide Web Conf. (WWW'8), Toronto, May 1999*, pages 1155–1169. Elsevier Science B.V., 1999.

[10] M. Erdmann and R. Studer. Ontologies as Conceptual Models for XML Documents. In *Proceedings of the 12th International Workshop on Knowledge Acquisition, Modelling and Mangement (KAW'99), Banff, Canada, October*, 1999.

[11] D. Fensel, S. Decker, M. Erdmann, and R. Studer. Ontobroker: The Very High Idea. In *Proceedings of the 11th International Flairs Conference (FLAIRS-98), Sanibel Island, Florida, May*, 1998.

[12] M. Fernandez, D. Florescu, J. Kang, and A. Levy. Catching the Boat with Strudel: Experiences with a Web-Site Management System. In *Proceedings of the 1998 ACM Int. Conf. on Management of Data (SIGMOD'98) , Seattle, WA*, pages 414–425, 1998.

[13] P. Fröhlich, W. Neijdl, and M. Wolpers. KBS-Hyperbook - An Open Hyperbook System for Education. In *10th World Conf. on Educational Media and Hypermedia (EDMEDIA'98), Freiburg, Germany*, 1998.

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

[15] M. Kesseler. A Schema Based Approach to HTML Authoring. In *Proceedings of the 4th Int. World Wide Web Conf. (WWW'4). Boston, December*, 1995.

[16] M. Kifer, G. Lausen, and J. Wu. Logical Foundations of Object-Oriented and Frame-Based Languages. *Journal of the ACM*, 42:741–843, 1995.

[17] N. Kushmerick. Wrapper Induction: Efficiency and Expressiveness. *Artificial Intelligence*. in press.

[18] L. Lamping, R. Rao, and P. Pirolli. A Focus+Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 401–408, 1995.

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

[20] 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.

[21] H. Maurer. *Hyperwave. The Next Generation Web Solution.* Addison Wesley, 1996.

[22] A. McCallum, K. Nigam, J. Rennie, and K. Seymore. A Machine Learning Approach to Building Domain-Specific Search Engines. In *Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 662–667, 1999.

[23] E. Mena, V. Kashyap, A. Illarramendi, and A. Sheth. Domain Specific Ontologies for Semantic Information Brokering on the Global Information Infrastructure. In N. Guarino, editor, *Formal Ontology in Information Systems*. IOS Press, 1998.

[24] J. Robie, J. Lapp, and D. Schach. XML Query Language (XQL). In *Proceedings of the W3C Query Language Workshop (QL-98) , Boston, MA, December 3-4*, 1998.

[25] A. Sahuguet and F. Azavant. Wysiwyg Web Wrapper Factory (W4F). Technical report, 1999. http://db.cis.upenn.edu/DL/WWW8/index.html.

[26] UMLS. Unified Medical Language System . http://www.nlm.nih.gov/research/umls/.

[27] W3C. XML Specification. http://www.w3.org/XML/, 1997.

[28] W3C. RDF Schema Specification. http://www.w3.org/TR/PR-rdf-schema/, 1999.

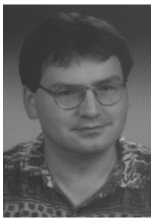
[29] S. Weibel, J. Kunze, C. Lagoze, and M. Wolf. *Dublin Core Metadata for Resource Discovery*. Number 2413 in IETF. The Internet Society, September 1998.

[30] G. Wiederhold and M. Genesereth. The Conceptual Basis for Mediation Services. *IEEE Expert / Intelligent Systems*, 12(5):38–47, September/October 1997.

Vitae

Steffen Staab is Assistant Professor for Applied Computer Science at Karlsruhe University. He has published in the fields of computational linguistics, information extraction, knowledge representation and reasoning, knowledge management, knowledge discovery, and intelligent systems for the web. Steffen studied computer science and computational linguistics between 1990 and 1998, earning a M.S.E. from the Univ. of Pennsylvania during a Fulbright scholarship and a Dr. rer. nat. from Freiburg University during a scholarship with Freiburg's graduate program in cognitive science. Since then, he has also been working as a consultant for knowledge management at Fraunhofer IAO and at the start-up company Ontoprise.

Juergen Angele received the diploma degree in computer science in 1985 from the University of Karlsruhe. From 1985 to 1989 he worked for the companies AEG, Konstanz, and SEMA GROUP, Ulm, Germany. From 1989 to 1994 he was a research and teaching assistant at the University of Karlsruhe. He did research on the operationalization of the knowledge acquisition language KARL, which led to a Ph.D. from the University of Karlsruhe in 1993. In 1994 he became a full professor in applied computer science at the University of Applied Sciences, Braunschweig, Germany. In 1999 he cofounded the company Ontoprise together with S. Decker, H.-P. Schnurr, S. Staab, and R. Studer and has been CEO of Ontoprise since then. His interests lie in the development of knowledge management tools and systems, including innovative applications of knowledge-based systems to the WWW.

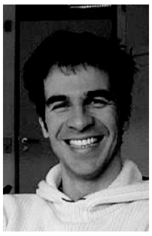


Stefan Decker is working as a PostDoc at Stanfords Infolab together with Prof. Gio Wiederhold in the Scalable Knowledge Composition project on ontology articulations. He has published in the fields of ontologies, information extraction, knowledge representation and reasoning, knowledge management, problem solving methods and intelligent systems for the web. He is one of the designers and implementers of the Ontobroker-System. Stefan Decker studied Computer Science and Mathematics at the University of Kaiserslautern and finished his studies with the best possible result in 1995. From 1995-1999 he did his PhD-studies at the University of Karlsruhe, where he worked on the Ontobroker project.

Michael Erdmann gained his Master Degree in Computer Science from the University of Koblenz (Germany) in 1995. Since October 1995 he works as a junior researcher at the University of Karlsruhe (Germany). He is a member of the Ontobroker-Project-Team and currently engaged in finishing his PhD. about the relationship between semantic knowledge modelling with ontologies and XML.

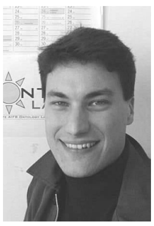
Andreas Hotho is a Ph.D. student at the Institute of Applied Computer Science and Formal Description Methods at Karlsruhe University. He earned his Master's Degree in information systems from the University of Braunschweig (Germany) in 1998. His research interests include the application of data mining techniques on very large databases and intelligent Web applications.

Alexander Maedche is a Ph.D. candidate at the Institute of Applied Computer Science and Formal Description Methods at Karlsruhe University. He received his diploma in industrial engineering (computer science, operations research) in 1999 from Karlsruhe University. His research interests include ontology engineering, machine learning, data and text mining and ontology-based applications.



Hans-Peter Schnurr is a Ph.D. candidate at the Institut of Applied Computer Science and Formal Description methods at Karlsruhe University. He received his diploma in industrial engineering in 1995 from Karlsruhe University. Between 1995 and 1998, Hans-Peter was working as a researcher and practice analyst at McKinsey & Company and is co-founder of the start-up company Ontoprise, a knowledge management solutions provider. His current research interests include knowledge management methodologies and applications, ontology engineering and ontology-based applications.

Rudi Studer obtained a Diploma in Computer Science at the University of Stuttgart in 1975. In 1982 he was awarded a Doctors degree in Mathematics and Computer Science at the University of Stuttgart, and in 1985 he obtained his Habilitation in Computer Science at the University of Stuttgart. From January 1977 to June 1985 he worked as a research scientist at the University of Stuttgart. From July 1985 to October 1989 he was project leader and manager at the Scientific Center of IBM Germany. Since November 1989 he has been Full Professor in Applied Computer Science at the University of Karlsruhe. His research interests include knowledge management, intelligent Web applications, knowledge engineering and knowledge discovery. He is co-founder and member of the scientific advisory board of the knowledge management start-up company Ontoprise.



York Sure is a Ph.D. candidate at the Institut of Applied Computer Science and Formal Description Methods at Karlsruhe University. He received his diploma in industrial engineering in 1999 from Karlsruhe University. His current research interests include knowledge management, ontology merging and mapping, ontology engineering and ontology-based applications.