GeS 2007 German e-Science

Available online at http://www.ges2007.de

Collaborative Ontology Development — Distributed Architecture and Visualization

N. Malzahn[1], S. Weinbrenner[1], P. Hüsken[1], J. Ziegler[1] and H.U. Hoppe[1]

Department of Computer Science, Faculty of Engineering, University Duisburg-Essen,
Forsthausweg 2, 47048, Duisburg, Germany
email: {malzahn, weinbrenner, hoppe}@collide.info,
{huesken, ziegler}@interactivesystems.info
phone: (+49 203) 379 1450, *fax:* (+49 203) 379 3557

Abstract

In this paper we present the architecture of the browser-based community-driven ontology engineering platform *Ontoverse*. We will present the architectural needs and designs for an extensible collaborative ontology platform as well as the current implementation based on tuplespaces. In this context we briefly introduce the SQLSpaces and the Semantic Web Application Toolkit (SWAT). To provide interactive collaborative means for editing, merging, and discussing about ontologies adequate visualization techniques are needed to support the ontology designers and ontology users. Therefore we introduce a visualization method called *SmartTree* that implements focus and context techniques.

1 Introduction

Domains such as information technology or life sciences are characterized by rapid changes and increments in concepts and terminology. Thus, it is important for domain experts to be able to share their perspectives to avoid divergency. To facilitate a shared conceptualization (cf. [1]) the ontology development should be done collaboratively. A platform for collaborative ontology development not only has to provide the means for editing an ontology like Protégé [2] does, but also for discussions, annotations and assignment of resources to ontology objects (viz. concepts, instances and relations). Besides there should be a methodology underlying the workflow support of the platform to ease the ontology building process.

Currently there are several approaches to supporting collaborative ontology creation, trying either to enhance existing tools with collaborative means like Co-Protégé [3] or to develop complete collaborative environments such as OntoEdit [4]. Most of them are not browser-based. That means, they are not easily accessible from different places without having an additional application installed to maintain and edit the ontology or they lack adequate graphical interfaces for ontology design, evolution and maintenance. The approaches mentioned do not provide copyright protection and security management. To represent the state

of the art the targeted platform has to take into account copyright needs of the ontology designers and the domain experts.

To combine the different aspects and needs of the platform, a loosely coupled (agent-driven) architecture is an adequate way to re-use many existing components and to be free to choose the most appropriate (programming) languages for each of the services to be provided for the user.

In the following sections we will present the architecture and visualization means of a browser based collaborative ontology platform with a graphical interface that is currently being developed in the Ontoverse project[1].

2 Collaboration Architecture

The back-end of the system has to persistently store large amounts of data and has to grant concurrent access to it. Moreover it has to support group and community awareness features, the management of different branches of an ontology, conflict resolution means during merge processes, and a user management concept including groups and roles. Last but not least it needs a way to store additional data like timestamps and other copyright information in an easily extendable manner.

Since a main requirement of the system is flexibility, it should consist of independent modules, so they can be adapted and assembled corresponding to the current needs. To reach this goal we decided to use a blackboard architecture. The main advantage of this idea is called loose coupling, which means that clients do not need to know much about each other. All clients act on their own and interact indirectly with each other.

A concrete specification of this blackboard architecture is the tuple space concept (cf. [5]). A tuple space server provides access to several spaces. Each space represents an independent data storage. We developed a tuple space implementation called SQLSpaces, which works on a relational database and translates Linda operations into SQL statements. There are several advantages of SQLSpaces versus other implementations like JavaSpaces [6] or TSpaces [7]: persistency provided by the underlying database, a versioning system, awareness features, extendability towards other systems. Those are the ingredients needed for a collaborative platform.

The flexible agent architecture allows to implement the intelligent system components in Prolog offering an elegant and efficient solution for some problems related to ontologies like e. g. consistency checking. The connection to an SQLSpaces server is quite easy to establish, because the protocol used to communicate between client and server is XML based.

There is already a prominent framework to handle ontological data called Jena [8], but there are several reasons why Jena does not meet all our criteria. First of all Jena was mainly developed to have all the data in the working memory and does not provide sufficient means for incremental changes in combination

[1]The Ontoverse project is funded by the Federal Ministry of Education and Research. Project no. 01C5975

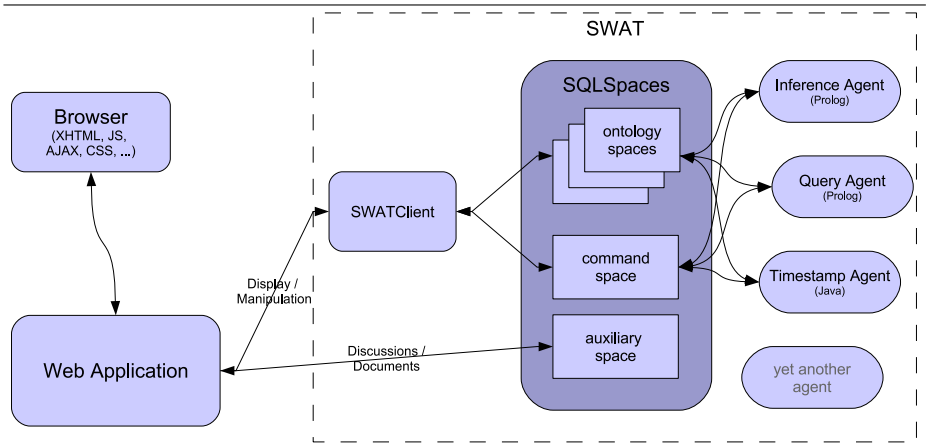

Figure 1: SQLSpaces / SWAT architecture

with a persistent data storage. This might be quite efficient, but it limits the amount of data, that can be stored, and it bears the danger of data loss in case of technical problems. Though Jena does support relational databases as data storages, its support is only rudimentary. Changes to the ontological data are not stored immediately into the database, but are maintained in the working memory until explicitly written back. However, if the data is stored in working memory, the changes of the entailed facts should be calculated incrementally, i. e. the difference to the previous state should be determined. If facts are deleted, Jena simply drops the whole entailment and calculates anew. Moreover Jena neither offers a version control system, nor does it have a copyright management facility, which are both crucial for a public collaborative ontology editing platform.

Versioning and awareness facilities are core features of SQLSpaces whereas the needed copyright management is easily provided as an agent-based service on top of it.

SWAT Since the blackboard architecture is only a basic concept of an underlying structure it needs a concrete framework with an interface for the upper layers. In this case the framework the higher levels are using is called SWAT, the *semantic web application toolkit*, and consists of a set of tools to store, manage and work with ontological data as seen in fig. 1. These components are written in Java or Prolog and are handled by a global SWAT client that is written in Java.

SWAT uses several spaces (i. e. blackboards) to separate the data of the ontologies from messages exchanged by the agents. Furthermore each ontology lies in its own space, so there are no interferences between the ontologies. However, it is also possible to import ontologies into each other by connecting to several spaces at once. These multi-connections allow to build logically unified spaces by adjusting the corresponding SQL statements in the SQLSpaces server.

To support collaborative work on parts of ontologies two types of awareness modes are supported by the SWAT framework. The first type of awareness is needed when several users work on the same part of the ontology, but not at the same time. For this asynchronous collaboration a CVS[2]-like system is used. Users checkout a public version and thus create a private workspace that cannot be seen by any other user. On the SQLSpaces layer this private workspace is a branch of the public version that belongs to the specific user. When the user finishes his modifications of the ontology he can commit his version to the original version or to a newer version that succeeded the original one. During the commit all differences between both versions are calculated and a list of conflicts is generated that could not be solved automatically (optimistic locking strategy). As soon as these conflicts are solved by the user the commit is executed and the result is available as a new public version.

In contrast to this more loose collaboration, a tight synchronous mode is also supported. Therefore a user can share his private workspace with another user, who will then also be able to modify the ontological data in this workspace. To prevent conflicts, ontological entities are locked during edits (pessimistic locking strategy). All changes are instantantenously visible of all users in the same workspace. All participants of the shared workspace get immediately notified about changes and write locks. If such a synchronous session is finished the private workspace can be committed and will become a new public version.

In order to guarantee copyright protection and advanced security features the SWAT client incorporates a security agent which is capable of registering the data with timestamps and signing the data at a trust center. It is possible to sign and stamp either fine grained data, like single tuples or coarse grained data such as whole (versions of) ontologies.

3 Visualization

For the development of an appropriate user interface a number of problems have to be addressed. Among the most challenging aspects are the visualization of possibly very large ontologies and the assistance for unexperienced users in order to let them understand knowledge structures more intuitively. These problems might occur when browsing, i. e. viewing ontologies.

Ontologies can be presented to the user in different ways. Most approaches concentrate on visualizing the basic structure like the class and the property hierarchy. The user interface of the Ontoverse platform offers a new visualization technique called *SmartTree* for presenting the concept hierarchy in combination with graph views, which are especially useful to explore the network structure of large ontologies. Within the SmartTree, we present ontology concepts as nodes and instances of concepts as special leaf types with different graphical representations. It is also possible to edit the instances in a separate view, if the number of instances is too large to show them clearly arranged inside the tree widget.

[2] **C**oncurent **V**ersioning **S**ystem

Our considerations for a user friendly tree widget leads to new visual functions. First of all we implement a basic *focus and context* technique, where the selected concept representation gets a bigger scale value than those concept representations in the distant areas inside the visualization. We use continuous falling scale values for concept representations that correspond with their distances to the currently selected concept representation (see fig. 2 on the left side). The advantage for the user is the presentation of details concerning the concept of interest, whereas connections to the other concepts remain visual, so that the user has a better overview with respect to the concept hierarchy. However this kind of visualization technique makes only sense if the user navigates within single subtrees and does not select distant context elements all the time.

Figure 2: SmartTree with focus and context (left) and Property-Lines (right)

Another new function that we implemented for the SmartTree are *Property-Lines*. Property-Lines represent OWL object properties [9], where the currently selected concept is part of their domain. Object properties are binary relations for the set of ontology concepts. For example an object property could express that some instances of concept A use some other instances of concept B. Property-Lines connect the actual selected concept with concepts in the range of the object property, so that the user gets an impression about the semantic infor-

mation of the ontology. Using Property-Lines the referenced concepts should also be emphasized, so that we modify the focus and context visualization: Instead of one single focus, we use a set of focus points each with juxtaposed contexts. In fig. 2 (right side) the selected concept *BioinformaticsTask* is connected to the concepts *ComputationalMethod* and *Program* with additional straight lines beside the tree illustration. Also the enhanced focus and context technique is shown in this figure.

Condense & Explode We are currently exploring new interactive functions for the SmartTree. For example we realize a new flap mechanism for hiding subtrees in the concept hierarchy that the user is not interested in. Along with the well-known flap mechanism to hide subtrees by clicking on the minus symbol and showing it again by clicking on the plus symbol, we are implementing a technique called *Condense & Explode*: Selecting a hierarchy line (see left side in fig. 3) every subtree in this hierarchy level with the same parent node will be faded out (condensed) and represented by an elision symbol. Clicking again on the line will show the full tree again. The benefit for the user is to hide uninteresting parts of the concept hierarchy, so that we could better utilize the limited space in the vertical dimension for more important concept representations.

User Adaptation At this time we are extending the SmartTree towards user adaption. An important aspect concerning visualization is to point out those objects that the user is interested in with respect to the application context. As we explained above, the SmartTree uses a focus and context technique with multiple focus elements, so we decided to emphasize the probably most interesting concept representations. In order to estimate the interesting objects the SmartTree components needs to be embedded into an applications that provides additional information about the current user. Unfortunately the information about the application context is not complete, so we have to revert to the observable user interactions. Considering the user interactivity in the time flow, i. e. monitoring the sequence of concept selections by using the computer mouse, the system provides an informative basis about the importance of objects inside the visualization. Every concept representation is associated with a value of its presumed importance — the so called *Degree of Interest* (DOI) [10]. In order to estimate the DOI values of every concept representation we have to distinguish two basic factors:

1. *A Priori Meaning (API).* Independent of any application context the API value of a concept representation depends on the ontology structure and is constant as long as the given ontology remains unchanged. API values have to be updated after a new stable ontology version has been released.
2. The *Distance D(x, FP)* between concept x and the concept *FP* that has the focus (also called *Focus Point*). Beside the geometric distance of the concept representations in the SmartTree, other types of distance can be used. For example the semantic distance D_s is defined as the shortest path in the ontology graph from x to *FP* or the similarity distance D_{sim}, so that

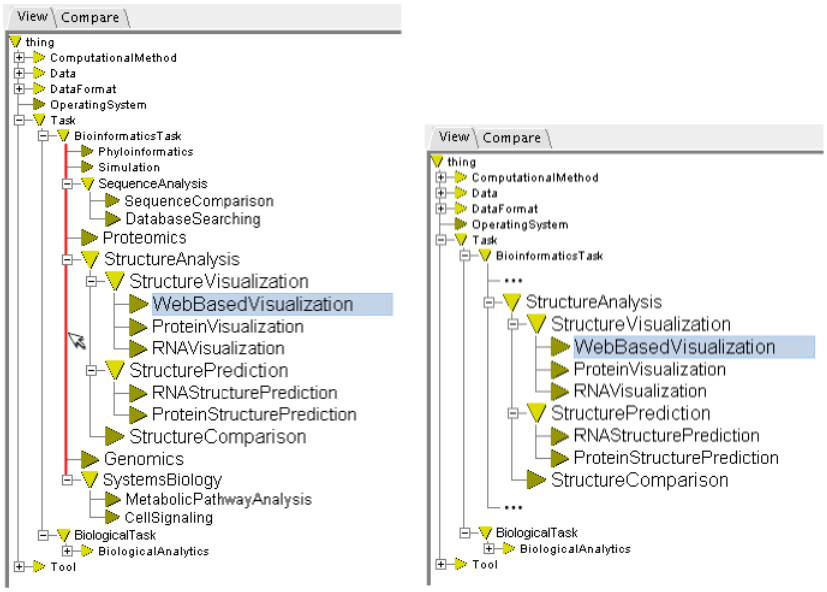

Figure 3: Selecting a hierarchy line (left), condensed SmartTree (right)

likewise concepts (with textual similarity, a set of shared properties, etc.) have a small distance value.

The DOI value for concept x with respect to the concept in focus can be calculated by function F:

$$DOI(x, FP) = F(API(x), D(x, FP)) \tag{1}$$

Ritter [11] describes explorations of interactive visualizations as an iterative process, because the user plans the next steps based on attained information. As a consequence the sequence of concept selections has to be followed:

$$DOI_i(x, FP_i) = \begin{cases} API(x) & : i = 0 \\ F(DOI_{i-1}(x, FP_{i-1}), D(x, FP_i)) & : i > 0 \end{cases} \tag{2}$$

Applying user adaption in this way, fast distant calculations are required, otherwise the SmartTree's performance will be affected adversely.

4 Usage Scenario

Based on the described architecture and visualization techniques the following scenario (among others) is envisaged: A user, expert in the domain, uses a

web browser to access the Ontoverse platform and searches information or software related to sequence analysis of proteins. As a result an ontology browser like in fig. 2 is shown. In addition to the view of the ontology structure, the user can access and navigate through discussions and annotations for the particular concepts found. Also, he has access to related publications and graphical network of researchers derived from co-publications. This network can be used to improve and enrich the existing ontology by relations implied by the underlying social network of the authors.

He can add and share his own annotations to the ontology. If the user is a member of the group of ontology designers for that specific part of the ontology, also editing the ontology concepts and relations is enabled. During the editing process gets notified that another user works on the same part of the ontology. Since the first user is interested in the opinion of the second one, he invites the other into his workspace. In the now shared workspace both researchers collaborate synchronously for some time. When they decide to publish (commit) their version, they get informed about changes of third parties, who caused a conflict because of name conflicts. To solve this matter the system offers either to rename their concept and commit the changes or to begin a discussion with the third party to coordinate the solution with them. If there is no agreement the two researchers may also go for the creation of a branch of the ontology. This leads to two public variants of the same ontology that can be discussed by the whole community.

A general description of the application area of the Ontoverse platform can be found in [12].

5 Ontology Reconciliation based on Community Observation

A lively community will adapt itself to new developments in their field of interest. The adaption will most likely manifest itself by the integration of new members into the community or by the re-orientation of known ones towards new topics (or artifacts) of interest. Sometimes this process is made very explicit by those people announcing the next big issues and grand challenges (such as in [13]) for a community, but most of the time the change is made silently and unnoticed by the community: there are persons who work on topics of two seemingly not connected topics – at least in terms of the current ontology. If more and more people work on two topics not connected in the ontology, it is time to investigate these two topics considering if the ontology has to be revised. This might provide deeper insights into the existing fields or strengthens a new research field. Social network analysis can support the community by highlighting missing links, confirm existing links or even questioning presumably existing links in the community's ontology. We use a weighted and standardized co-occurrence algorithm based on *actors' relationships* in a network of artifacts that can be compared to the existing ontology. The resulting network distinguishes three types of relations in the revised ontology:

- *green relations* indicating that these relations were confirmed by the artefact network, i. e. they are part of the given ontology and various persons are working on both topics.
- *red relations* indicating that these relations could not be confirmed by the artefact network, i. e. they are part of the map but no one is working on both topics.
- *blue relations* indicating that these relations are emerging from the artefact network, but they are currently not included in the ontology.

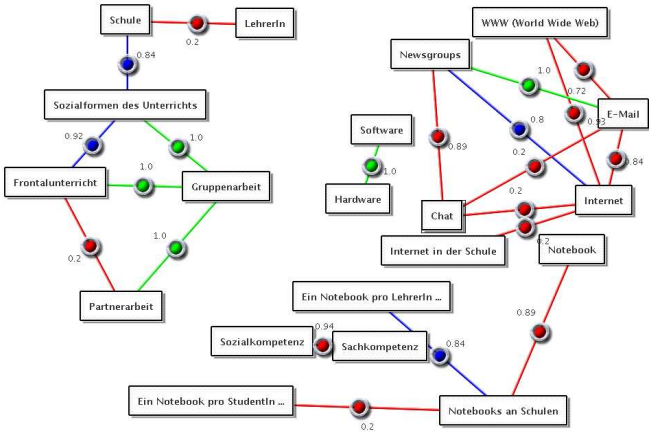

Figure 4: Semantic Web from Beats Biblionetz (http://beat.doebe.li/bibliothek) concept network compared to the author network also contained in Beats Biblionetz.

All three types of relations are valuable. Although the green ones may seem to be trivial because they are already known, they confirm the validity of the given ontology. The red ones are important because the observer might want to investigate if either the ontology has to be revised, because existing relations are not valid (anymore) or the relation could not be observed because of the given data. The most promising consequences might be drawn from the blue relations. These relations may indicate missing links in the ontology. If there are strong ties between two topics because of the amount or the reputation of the persons working on both topics the observer should carefully consider the inclusion of this link into the ontology. Thus the analysis of relations between actors and topics can be used to evolve an ontology semi-automatically.

6 Conclusion & Outlook

In this paper we presented the middleware and visualization concepts of the Ontoverse platform. Both are flexible and extensible to be able to constantly

evolve and improve the support of collaborative ontology development. Although a basic support for interactive and collaborative web-based ontology engineering is already established, further work needs to be spent on sophisticated awareness mechanisms in the user interface to make the ontology designers and users aware of other people working in the same field. Additionally we plan to integrate a comprehensive security layer starting with a trust center to secure the intellectual property of the contributors to the ontology.

Besides the interactive means to improve and evolve the ontologies hosted by our platform, we plan to deploy software agents to discover hidden relations in the ontological data — either by means of logic or by incorporating external knowledge sources like publication databases.

References

1. T. Gruber. A translation approach to portable ontology specifications. *Knowledge Acqusition*, 5(2):199–220, 1993.
2. W. E. Grosso, H. Eriksson, R. W. Fergerson, J. Gennari, S. W. Tu, and M. A. Musen. Knowledge modeling at the millennium (the design and evolution of protege-2000). In *Proceedings of the 12th Knowledge Acquisition Workshop (KAW'99)*, Canada, 1999.
3. A. Díaz, G. Baldo, and G. Canals. Co-protege: Collaborative ontology building with divergences. In *Proc. of DEXA'06*, pages 156–160, 2006.
4. Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, and D. Wenke. OntoEdit: Collaborative ontology development for the semantic web. In *Proc. of ISWC 2002, 2002*, volume 2342 of *LNCS*. Springer, 2002.
5. D. Gelernter. Generative communication in Linda. *ACM Trans. Program. Lang. Syst.*, 7(1):80–112, 1985.
6. Sun Microsystems. Javaspace specification, 1998.
7. T. J. Lehman, S. W. McLaughry, and P. Wycko. TSpaces: The Next Wave. In *HICSS*, 1999.
8. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson. Jena: Implementing the semantic web recommendations, 2003.
9. Michael K. Smith, Chris Welty, and Deborah L. McGuinness. Owl web ontology language guide. `http://www.w3.org/TR/owl-guide/`, February 2004. W3C Recommendation.
10. G. W. Furnas. Generalized fisheye views. In *CHI '86: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 16–23, New York, NY, USA, 1986. ACM Press.
11. F. Ritter, editor. *Interaktives Illustrieren von Informationsräumen: Räumliche und funktionale Zusammenhänge spielerisch begreifen*. Der Andere Verlag, 2005.
12. Ingo Paulsen, Dominic Mainz, Katrin Weller, Indra Mainz, Jochen Kohl, and Arndt von Haeseler. Ontoverse: Collaborative knowledge management in the life sciences network. In *Proceedings of the German E-Science Conference 2007*. available online at `www.ges2007.de`, 2007.
13. Ron Brachman. (AA)AI— more than the sum of its parts. *AI Magazine*, 27(4):19–34, winter 2006.