KAON - Towards a large scale Semantic Web

E. Bozak[1] and M. Ehrig[2] and S. Handschuh[2] and A. Hotho[2] and A. Maedche[1] and B. Motik[1] and D. Oberle[2] and C. Schmitz[1] and R. Studer[1,2] and G. Stumme[2] and Y. Sure[2] and S. Staab[2] and L. Stojanovic[1] and N. Stojanovic[2] and J. Tane[2] and R. Volz[1,2] and V. Zacharias[1]

[1]Forschungszentrum Informatik FZI, 76131 Karlsruhe,
http://www.fzi.de/wim
[2]Institute AIFB, University of Karlsruhe, 76128 Karlsruhe,
http://www.aifb.uni-karlsruhe.de/WBS

Abstract. The goal of Semantic Web is to enrich the content of Web pages with metadata, thus obtaining a well-defined semantic interpretation of content. Common language for concept description is provided by ontologies, that describe a model of a particular application domain. However, creation and management of ontologies has proven to be a demanding task. This paper introduces, KAON, the Karlsruhe Ontology and Semantic Web framework, providing services for ontology and metadata management and interfaces needed to create and access Web-based semantics-driven E-Services.

1 Introduction

The Web in its' current form is an impressive success with a growing number of users and information sources. Tim Berners-Lee, the inventor of the WWW, coined the vision of a Semantic Web in which unstructured content is enriched by metadata, obtaining a well-defined semantic interpretation of content suitable for more automated information processing. This is perceived as a key solution to the growing number of problems of locating information in the constantly expanding Web space. Semantic interpretation of content is achieved by using ontologies - repositories of common vocabulary and modeling constructs for content descriptions. Ontologies play a central role in providing interoperability between Web applications, e.g. B2B iontegration in the context of E-Business or information and service discovery currently developed on top of the emerging Semantic Web.

RDF (Resource Description Framework) provided by W3C has been identified as a core data model and infrastructure for metadata representation on the Web. Several RDF-APIs, parsers, schema and metadata editors, repositories, etc. have already been developed within the Semantic Web community. However, to establish and use Semantic Web technology as a basis for e-commerce, much more specialized, comprehensive and integrated tools are required. The Karlsruhe Ontology and Semantic Web framework (KAON) builds on available resources and provides tools for the engineering, discovery, management, and presentation of ontologies and metadata. It establishes a platform needed to apply Semantic Web technologies to e-commerce and B2B scenarios. Because of that, important design goals were robustness and scalability, since these

are key quality factors for any enterprise application. In this paper the vision and the current status of KAON are presented. The official KAON community web site [1] also provides up-to-date information about the project and allows downloading the newest version of the software.

Section 2 collects and summarizes requirements for a framework for semantics-based E-Services. Section 3.1 presents conceptual architecture that fulfills the described requirements. Section Section 3.2 presents the Web service interface to KAON and lists examples using this interface. Section 3.3 discusses in detail the KAON API - the most fundamental architectural element. Section 3.4 discusses the architecture and the implementation of the KAON Server - a back-end system and application server for RDF storage. Finally, a short overview of related work and next steps planned within KAON project are presented.

2 Requirements

While building semantics-based applications within E-Commerce, Knowledge Management, Web Portals, etc. we have gained insight into application features that warrant a success. Based on that experience and in order to enabling reuse across projects, we have decided to build a framework addressing these issues. An extensive requirement gathering process was undertaken to come up with a set of requirements that such framework must fulfill. The following key requirements were identified:

- **Accessability**: A framework should enable loose coupling, allowing access through standard web protocols, as well as close coupling by embedding it into other applications. This should be done by offering sophisticated standard APIs.
- **Scalability**: As well for engineering, evolving and deploying ontologies, scalability is an essential point for every kind of application.
- **Consistency:** Consistency of information is a critical requirement of any enterprise system. Each update of a consistent model must result in a model that is also consistent. In order to achieve that goal, precise rules must be defined for model evolution and an evolution service implementing these rules has to be provided. Also, all updated to the model must be done transactionally - either all model updates succeed, or none do.
- **Concurrency:** It must be possible to access and modify information concurrently. Modifications from concurrent users must be isolated from one another, preventing several users modifying the same data at once. This can also be achieved using transactional processing, where objects can be modified at most by one transaction at the time.
- **Persistent Storage**: An almost trivial requirement easily accomplished by reusing existing database technology. A sophisticated storage system must offer facilities for replication: for often used RDF models redundant copies must be maintained to address scalability and availability problems.
- **Security**: Information security means protecting information against unauthorized disclosure, transfer, modification, or destruction, whether accidental or intentional

[1] http://kaon.semanticweb.org

and is also a critical requirement. To realize it, any operation should only be accessible by properly authorized agents. Proper identity of the agent must be reliably established, by employing known authentication techniques. Sensitive data must be encrypted for network communication and persistent storage. Finally, means for auditing (logging) of sensitive operations should be present.
- **Reasoning**: There are many different reasoning engines with different underlying semantics (e.g. frame-based logic, description logic, etc.) available. A comprehensive framework should provide access to different reasoning engines providing flexible means for switching between different semantics.
- **Mediation**: Often multiple ontologies (e.g. based on different product standards) have to be supported by an ontology broker system. Therefore, means for mapping and mediating between heterogeneous ontologies are required.
- **Discovery:** Metadata in the Semantic Web is typically distributed, so means for ontology-focused and intelligent discovery of metadata are required. Based on a semantic description of the search target, the system should be able to discover relevant information on the Web. The same principle can be used to discover metadata about E-Services.
- **Usability:** Tools built using the framework should be easy to use and provide comprehensive user interfaces tailored to needs of different user groups, such as domain experts, ontology creators, business analysts and end users.
- **Internationalization:** The framework should allow users to create ontologies and their instances in different languages and should support non-Latin character sets.

3 KAON

This section describes KAON, the Karlsruhe Ontology and Semantic Web framework, that has been used to develop several semantics-based Web applications.

3.1 Conceptual Architecture

In this section we introduce the general architecture of KAON. The conceptual architecture of KAON follows Layers and Model-View-Controller architectural patterns [10]. Components are organized in three layers: data and remote service layer, the middleware layer and the applications and services layer, as shown in Figure 1.

Applications and Service Layer: Application clients can be stand-alone Java client applications, Web applications and Web service-based clients. OntoMat application framework can be used to provide integration of several components within a common stand-alone application, whereas Web clients can be embedded into KAON-PORTAL portal generation framework. Web service-based clients (e.g. realized within Microsoft's .NET platform) can access middleware layer via SOAP. All application clients connect with the middleware layer via KAON API, an application programming interface accessing ontology elements. The API realizes the application model by providing a set of object-oriented abstractions of ontology elements. Application clients provide views and controllers for model realized by KAON API.

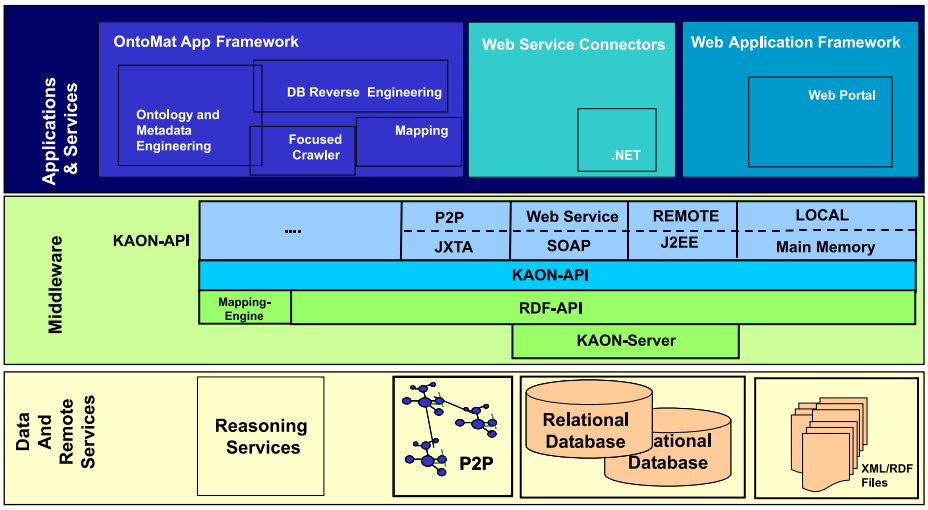

Fig. 1. KAON Architecture

Middleware Layer: The primary role of the middleware layer is to realize application model and provide KAON API interface for model access. KAON API isolates clients from different API realizations and provides a unified interface. Some implementations of KAON API rely on a modified implementation of RDF API[2] for access to RDF. Default implementation is used for in-memory processing of ontologies stored in files. KAON Server is an implementation of RDF API using an application server, allowing concurrent model modification, transaction support and model persistence. Non-RDF data sources may be accessed using another implementation of KAON API, thus creating an ontology-compatible view of data not in a format according to Semantic Web standards. KAON API can be accessed through a local interface (API implementation is embedded in the application), remote interface (API itself is realized on an application server) or through Wed Service interface allowing access using SOAP.

As KAON API is extensible, we are currently working on an implementation of KAON API that realizes virtual ontologies obtained by mapping or merging two intput ontologies (represented through appropriate realizations of the API).

Data and Remote Service Layer: This layer is provided by relational database technology or a flat file system supporting different serialization syntaxes. Additionally, several external services for reasoning, mapping and mediation services, etc. may be connected.

[2] http://www-db.stanford.edu/ melnik/rdf/api.html

3.2 Web Service Clients

The first Web Service client we developed within KAON has solved the following problem: KAON is a framework that is completely based on Java. However, using its SOAP interface it is not limited to usage in Java projects only. In order to provide interoperability with other platforms, the SOAP-based Web service interface is provided, making KAON accessible from virtually any distributed computing platform and client technology. Thus, the KAON's Web service API is strategically important in providing interoperability with Microsoft's .NET platform.

The Web service interface has been applied to create a Web based ontology browsing and querying system built on IIS/ASP technologies. Also, a dedicated plug-in has been created for Microsoft Office to provide metadata extraction from MS Office documents connecting to KAON ontologies[3].

3.3 KAON API

KAON API is the focal point of the KAON framework architecture, since it defines the model portion of a Model-View-Controller architecture. It provides objects representing various pieces of an ontology, such as Concept, Relation, Attribute or Instance, objects for creating and applying changes to ontology entities as well as objects providing query facilities. KAON API itself doesn't realize persistence, concurrency or security. Rather, it relies on lower layers to provide these features. KAON API may be accessed in following ways:

- For single-user scenarios with ontologies stored in XML files, KAON API implementation may be embedded in the application. This results in compact and easily deployable applications.
- If concurrent access of multiple users or ontology persistence is needed, an implementation relying on KAON Server may be used. KAON API may also be implemented remotely on an application server and accessed using CORBA-IIOP protocol. In this case, KAON API is also responsible for synchronizing actions of multiple users, while KAON Server is responsible for providing transactions, concurrency and security mechanisms.
- KAON API may be accessed through SOAP, allowing easy construction of ontology-enabled Web services.
- In peer-2-peer environments, KAON API may be used through Java JXTA protocol[4]. We refer the interested reader to the Edutella project[5] where we focus on connecting KAON with P2P networks.

The Observable design pattern [3] is used for notifications about model changes, thus achieving low coupling between model and associated views. All changes to application model, whether local or remote, are propagated to registered listeners allowing them to display model updates immediately as they happen. Java Messaging Service

[3] Further information is available at http://www.ontologging.org
[4] http://www.jxta.org
[5] http://edutella.jxta.org

(JMS) is used to propagate change notifications in distributed environment. The API is entirely based on interfaces, allowing users to choose the appropriate implementation, depending on his needs

- Default implementation using RDF API as storage facility. A local or KAON Server based implementation of the API may be used.
- To provide ontology-compliant access to data stored in existing systems, such as relational or XML databases, special mapping implementations may be used. These implementations generates ontologies from respective data sources. The conversion is dynamic - all modifications to the ontology and all queries are transformed and propagated to the underlying data source.
- As metioned earlier, we are working on a extensions of KAON API dedicated to ontology mapping. It takes two ontologies and a set of mapping rules (expressed as an instance of a predefined mapping ontology) and creates a new virtual ontology according to specified mapping rules. This mapping between ontologies is also dynamic - a change in any of the underlying ontologies is immediately reflected in the virtual ontology and vice versa.

KAON API is responsible for providing consistency of the underlying ontology. All access to the API is performed through a dedicated evolution strategy whose purpose is to define and implement a set of change rules. For example, when a concept is removed from an ontology, it must be decided what to do with its subconcepts - they may be deleted, attached to the parent of the deleted concept or attached to ontology's root concept. Several evolution strategies have been implemented for each of these policies, allowing the user to choose the appropriate one when the ontology is instantiated. Finally, in order to improve performance, KAON API allows using a pluggable caching scheme. In that way many costly requests to the application server may be avoided and the overall application performance increased. The caching scheme is responsible for maintaining cache coherency in case of distributed operation.

3.4 KAON Server — RDF Application Server

As experience gathered in first Semantic Web applications shows, ontologies and metadata are usually created in a collaborative and incremental way, creating a need for concurrent updates of ontologies and metadata. KAON Server is responsible for providing a persistent, transactional and secure RDF repository accessible by multiple users are the same time. It is realized within J2EE framework, thus making the server deployable on any compliant EJB application server. KAON Server has been tested with JBOSS[6] - Open Source application server. The conceptual architecture of the system follows a Layers architectural pattern, as presented in Figure 2.

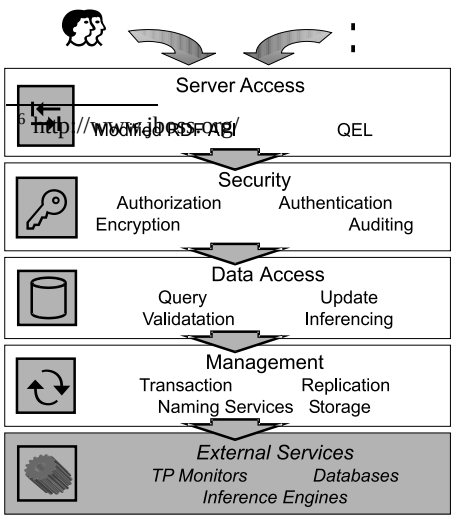

Fig. 2. RDF Application Server

API layer allows accessing the KAON Server. A modified version of RDF API is used for modifying RDF models, and

querying API based on Edutella QEL language is used for model querying and inferencing. Since KAON Server is realized within J2EE framework, API layer is implemented using CORBA-IIOP protocol.

Security layer makes server operations available to the client only if the caller is properly authenticated and authorized to access them. Authentication and authorization are implemented using Java Authentication and Authorization Service (JAAS) allowing easy integration to any existing security services. Using JAAS authentication scheme identities are mapped into abstract roles, and for each role a set of privileges is determined.

Data access layer layer allows management of RDF model elements, inferencing and querying. Queries are supported using RDF-QEL query language designed in the Edutella project [4], while updates are possible via a modified version of RDF API. KAON Server does not implement inference itself but interfaces to other systems. The integration with these systems is seamless - the users of KAON Server do not distinguish between inferred and ground facts.

Management layer encapsulates all ”basic” services such commonly found in information systems. Transaction management system is responsible for ensuring the commonly known ACID transaction properties. The replication service must ensure all external systems work with the same data sets (e.g. inference engine must be kept in synchrony with the persistent storage). By interplaying with the naming service, the replication service can also manage duplicate RDF models to enhance scalability and availability. The naming service maps model identifiers (as presented by URIs) to persistent identifiers (URNs) and keeps information about the location of the information. Finally, system configuration modules are realized in this layer.

External services are systems and services external to KAON Server. Databases are used for persisting RDF model data. Inference engines are reused to offer reasoning capabilities. Transaction Processing Monitors ensure transactional integrity if data is replicated to external systems.

4 Related Work and Conclusion

This section gives a short overview on related work and an outlook on the next steps that are to be carried out within the Karlsruhe ontology and semantic web infrastructure project. With the upswing and proliferation of ontologies and metadata, the need for

comprehensive managing infrastructure has been recognized recently. A comprehensive overview and state-of-the-art survey on ontology library systems with respect to the dimensions management, adaptation and standardization has been provided by [2]. Whereas these ontology library systems mainly focus on ontology storage and reuse, our approach provides a RDF-based framework including ontology management for semantics-driven applications. Comparing to available RDF data stores[7] our apporoach is also the only available RDF data store dealing with replication and connectable in a peer-to-peer manner. An approach that comes close to our open-source framework is the commercial system, ontology builder and server, proposed in [1]. Nevertheless, in contrast to this system, our approach is completely based on RDF and is therefore Semantic Web conform.

In this paper we have introduced KAON, the Karlsruhe Ontology and Semantic Web framework. Ontologies and metadata are important means for realizing the vision of the Semantic Web. In this paper it has been argued that the visionary goal of E-Services to providing interoperability between heterogeneous information providers and consumers requires practical and research experience gained in Semantic Web. Since Semantic Web is a vision of providing meaning to Web content, similar principles can be used and provide E-Service communication with formal semantics. KAON is a framework that allows to establish a connection between Semantic Web technology with the Web Service world.

Acknowledgements: The research presented in this paper has been partially funded by EU-IST in the Ontologging project and EU-FET in the WonderWeb project.

5 Appendix

Definition 1. *A* core ontology *is a structure*

$$\mathcal{O} := (C, \leq_C, R, \sigma, \leq_R)$$

consisting of

- *two disjoint sets C and R whose elements are called* concept identifiers *and* relation identifiers, *resp.*,
- *a partial order $\leq_C$ on C, called* concept hierarchy *or* taxonomy,
- *a function σ: $R \to C^+$ called* signature,
- *a partial order $\leq_R$ on R, called* relation hierarchy, *where $r_1 \leq_R r_2$ implies $|\sigma(r_1)| = |\sigma(r_2)|$ and $\pi_i(\sigma(r_1)) \leq_C \pi_i(\sigma(r_2))$, for each $1 \leq i \leq |\sigma(r_1)|$.*

Remark 1. Often we will call concept identifiers and relation identifiers just *concepts* and *relations*, resp., for sake of simplicity.

[7] http://www.w3.org/2001/05/rdf-ds/DataStore lists existing RDF data stores.

Definition 2. *For a relation $r \in R$ with $|\sigma(r)| = 2$, we define its* domain *and its range by* $\mathrm{dom}(r) := \pi_1(\sigma(r))$ *and* $\mathrm{range}(r) := \pi_2(\sigma(r))$.

If $c_1 \leq_C c_2$, for $c_1, c_2 \in C$, then c_1 is a subconcept of c_2, *and c_2 is a* superconcept of c_1. *If $r_1 \leq_R r_2$, for $r_1, r_2 \in R$, then r_1 is a* subrelation of r_2, *and r_2 is a* superrelation of r_1.

If $c_1 <_C c_2$ and there is no $c_3 \in C$ with $c_1 <_C c_3 <_C c_2$, then c_1 is a direct subconcept of c_2, *and c_2 is a* direct superconcept of c_1. *We note this by $c_1 \prec c_2$.* Direct superrelations *and* direct subrelations *are defined analogously.*

Definition 3. *Let $\mathcal{L}$ be a logical language. A* $\mathcal{L}$-axiom system *for an ontology $\mathcal{O} := (C, \leq_C, R, \sigma, \leq_R)$ is a pair*

$$A := (AI, \alpha)$$

where

- *AI is a set whose elements are called* axiom identifiers *and*
- *$\alpha\colon AI \to \mathcal{L}$ is a mapping.*

The elements of $A := \alpha(AI)$ are called axioms.

An ontology with $\mathcal{L}$-axioms *is a pair*

$$(\mathcal{O}, A)$$

where $\mathcal{O}$ is an ontology and A is a $\mathcal{L}$-axiom system for $\mathcal{O}$.

Definition 4. *An ontology with $\mathcal{L}$-axioms $(\mathcal{O}, A)$ is* consistent, *if $A \cup \{\forall x\colon x \in c_1 \to x \in c_2 \mid c_1 \leq c_2\} \cup \{\forall \boldsymbol{x}\colon \boldsymbol{x} \in r_1 \to \boldsymbol{x} \in r_2 \mid r_1 \leq r_2\}$ is consistent.*

Remark 2. In the sequel, *ontology* stands for either a core ontology or an ontology with $\mathcal{L}$-axioms.

Definition 5. *A* lexicon *for an ontology $\mathcal{O} := (C, \leq_C, R, \sigma, \leq_R)$ is a structure*

$$Lex := (S_C, S_R, Ref_C, Ref_R)$$

consisting of

- *two sets S_C and S_R whose elements are called* signs for concepts *and* relations, *resp.,*
- *a relation $Ref_C \subseteq S_C \times C$ called* lexical reference for concepts, *where $(c, c) \in Ref_C$ holds for all $c \in C \cap S_C$.*
- *a relation $Ref_R \subseteq S_R \times R$ called* lexical reference for relations, *where $(r, r) \in Ref_R$ holds for all $r \in R \cap S_R$.*

Based on Ref_C, we define, for $s \in S_C$,

$$Ref_C(s) := \{c \in C \mid (s, c) \in Ref_C\}$$

and, for $c \in C$,

$$Ref_C^{-1}(c) := \{s \in S \mid (s, c) \in Ref_C\} \ .$$

Ref_R and Ref_R^{-1} are defined analogously.

An ontology with lexicon *is a pair*

$$(\mathcal{O}, Lex)$$

where $\mathcal{O}$ is an ontology and Lex is a lexicon for $\mathcal{O}$.

Definition 6. *A* knowledge base *is a structure*

$$KB := (C_{KB}, R_{KB}, I, \iota_C, \iota_R)$$

consisting of

- *two sets* C_{KB} *and* R_{KB}*,*
- *a set* I *whose elements are called* instance identifiers *(or* instances *or* objects *for short),*
- *a function* ι_C*:* $C_{KB} \to \mathfrak{P}(I)$ *called* concept instantiation*,*
- *a function* ι_R*:* $R_{KB} \to \mathfrak{P}(I^+)$ *called* relation instantiation*.*

Definition 7. *An* instance lexicon *for a knowledge base* $KB := (C_{KB}, R_{KB}, I, \iota_C, \iota_R)$ *is a pair*

$$IL := (S_I, R_I)$$

consisting of

- *a set* S_I *whose elements are called* signs for instances*,*
- *a relation* $R_I \subseteq S_I \times I$ *called* lexical reference for instances*.*

A knowledge base with lexicon *is a pair*

$$(KB, IL)$$

where KB *is a knowledge base and* IL *is an instance lexicon for* KB*.*

References

1. Aseem Das, Wei Wu, and Deborah McGuinness. Industrial strength ontology management. In *Proceedings of the First Semantic Semantic Web Working Symposium, SWWS-01, Stanford, USA, August 2001*, 2001.
2. Ying Ding and Dieter Fensel. Ontology library systems — they key to successful ontology re-use. In *Proceedings of the First Semantic Semantic Web Working Symposium, SWWS-01, Stanford, USA, August 2001*, 2001.
3. Erich Gamma, Richard Helm, Ralph Johnson, and John Vlisside. *Design Patterns*. Addison-Wesley, 1995.
4. Wolfgang Nejdl, Boris Wolf, Changtao Qu, Stefan Decker, Michael Sintek, Ambjoern Naeve, Mikael Nilsson, Matthias Palmer, and Tore Risch. Edutella: A p2p networking infrastructure based on rdf. In *In Proceedings of the 11th World Wide Web Conference — WWW-11, Hawaii, USA, 2002*, 2002.