On the Role and Application of Ontologies in Information Systems

Thanh Tran, Holger Lewen and Peter Haase

Institute AIFB
Universität Karlsruhe (TH)
76131 Karlsruhe, Germany
Email: {dtr, hle, pha}@aifb.uni-karlsruhe.de

***Abstract*— Semantic Web research and recent efforts of large software companies have lead to mature technologies that can enable real-life semantic applications. While benefits such as advanced interoperability, search and data analysis are evident, little guidance is offered for the engineering of applications that can exploit them. To address this problem, the concept of an ontology, which is central to semantic applications, is compared to formalisms currently used in software engineering. More importantly, this paper proposes an extension to the three-tier architecture of enterprise information systems. The development of such a system is then demonstrated by the use of this architecture for an adaptive portal.**

I. Introduction

The Semantic Web (SW) as envisioned by Tim Berners-Lee [1] provides semantics to resources so that they can not only be interpreted by humans but also by computer agents. This is achieved by the use of ontologies containing formal descriptions of terminologies that can be interpreted by software agents in an automated way. These formal descriptions provide "understanding" at the machine-level. In this way, the SW ultimately shall provide a set of formalized corpora of interrelated, interoperable and reusable contents and services. So far, efforts driving the SW have lead to new standardized Knowledge Representation (KR) languages like RDF, OWL and SWRL, and many ontologies for the representation and exchange of artifacts of various kinds. Large-scale research projects have been funded to advance the state-of-the-art and recently, investments made by large companies such as Oracle, IBM, and SAP have resulted in mature Semantic Technologies for the storage, retrieval and exploitation of semantics. Yet, there are only few semantic applications available for the end user.

Partly, this is due to the lack of guidance for software engineers on how to develop semantic applications. While critics claim that the realization of the SW is infeasible in its entirety, they have to acknowledge that Semantic Web technology can solve various problems ranging from advanced data analysis, to enterprise data and application integration, to flexible exchange of content in portal applications [2]. Despite the many benefits of semantic technologies, they cannot be exploited immediately due to the lack of help for software engineers and concrete proposals for building semantic applications. To facilitate the adoption of the semantic paradigm, this paper provides a discussion on the role and potential uses of ontologies in software engineering (SE) and a proposal on how they can be exploited in an architecture for semantic applications.

In section 2, we first discuss related work. Since ontologies are central for the engineering of semantic applications, section 3 offers a quick introduction to ontologies to software engineers by comparing them to similar SE concepts. A semantic extension to the well established three-tier architecture for enterprise information systems is proposed in section 4. Section 5 then demonstrates how this architecture and the various semantic technologies can be applied in an adaptive ontology-based portal. Finally, the conclusion is given in section 6.

II. Related Work

Methodologies and discussions on the development of semantic application have been proposed in literature. CommonKADS [3] is a well-known methodology for the development of knowledge-based applications which can also be applied to building semantic information systems, but rather focuses on knowledge engineering and gives little guidance on how to exploit recent semantic technologies and concepts of the SW. The use of ontologies as a central knowledge platform to provide a single and shared representation of knowledge for all software components has been motivated in [4], but an elaboration on the architecture and functionalities of such a platform is missing. More precisely, it does not show how to use ontologies for these purposes. In [5], a concrete architecture for the development of semantic web services (SWS) has been presented. The focus lies however on the development of descriptions of SWS capabilities and the internal structure on a conceptual level which then can be translated into an ontology of a particular language. Similar work in [6] provides an execution environment for the discovery, mediation and invocation of SWS. Instead of service description, this paper demonstrates the use of semantic technologies for the concrete implementation of functional components. Very close to this is [7], which covers the development of semantic web applications. While it rather focuses on the interplay of the application and the SW—i.e. the potential for reuse of SW resources—some insights are given as to how to develop application components. While some of these ideas are adopted and

extended, this paper underlines the importance of the ontology management facility and specifically discusses the required functionalities and APIs [8].

In summary, our contribution is different from existing work to the extent that it gives guidance on how to exploit ontologies in the concrete implementation of software components. It also defines what the ontology management facility must correspondingly provide and how to integrate it into the three-tier architecture.

III. The Role of Ontologies in Information Systems

A. Introduction to Ontologies

In computer science, Tom Gruber's definition has become the most cited for the term ontology: "An ontology is an explicit and formal specification of a conceptualization" [9]. This definition has been extended by Uschold and Grüniger to highlight the fact that the conceptualization should be shared [10], i.e. it is agreed upon. According to the specification of current standard KR languages for the SW—particularly OWL [11]—an ontology can be seen as containing descriptions of concepts and relations formalized by the use of logical axioms that have the effect of limiting the number of possible model-theoretic interpretations. This axiom-based formalization of concepts makes them less ambiguous for machines. The degree of ambiguity varies with the expressiveness of the respective KR language—the current standard KR languages for the Semantic Web are OWL and SWRL.

B. Differences between Ontologies and Other Formalisms

1) Ontologies vs. XML Schema: In general, XML Schema is proposed as a mechanism to define the syntax for XML-documents. Because data can be encoded in different ways using XML, this allow parties to agree on a defined structure and labeling for the exchange of data [12]. While XML Schemas define the structure of the document, they do not define any meaning. Ontologies on the other hand are formalized in a way that limits possible interpretations. In particular, on the basis of possible interpretations, an OWL-aware tool can tell that two classes or individuals are equivalent or different. While XML and XML-Schema have strength in exchanging data, ontologies are used to exchange information.

2) Ontologies vs. Database Schema: Database schemas can be distinguished into the conceptual and the physical schema—the logical schema is a more detailed conceptual schema that is neglected here because it is out of the scope of this paper. The physical schema results from the mapping of the conceptual schema to physical storage objects—e.g. tables of a relational database. In particular, a conceptual database schema mostly reflects only a single or a limited viewpoint—namely that of its creators. When requirements change, the viewpoint and the schema respectively need to be modified. Ontologies in general are required to be shared, that is, reflect multiple viewpoints (domain ontologies in particular). A domain ontology does not need to be modified to meet changes in requirements but can be flexibly used to model any data requirements related to a particular domain. An important distinction between ontologies and database schemas is their behavior at runtime. After being translated to physical tables, the ontology is also available for retrieval and inferencing of new facts at runtime. The ontology can potentially increase semantic interoperability not only of the resources exchanged among systems but also of the data stored in physical databases. For instance, the formal descriptions of two differently labeled concepts may yield the same interpretations and so the corresponding tables can be inferred as containing semantically the same data despite the different table names. Therefore, when queries are formulated using ontology concepts, they can be processed also by external systems that may use different labels for the elements of the physical schema. In this regard, semantic interoperability means that not only the syntax but also the formalized semantics can be exploited for data retrieval. Even though database schema have shortcomings (e.g. being less flexible), they will not likely be replaced by ontologies in the near future. Note that ontologies are complementary to a physical schema or, more precisely, have to be mapped to the physical schema for the persistent storage of facts.

3) Ontologies vs. Object Model: As can be seen in table I, most of the differences between ontologies and object models are just in naming. An important difference to elaborate here is object behavior. Object behavior is specified by interfaces and implemented by methods in the object model. For the object model the concept of interfaces supports the principle of information hiding by encapsulation. Interfaces can be used to specify aspects of behavior which may be commonly shared by several classes. They thereby can be used to hide the underlying design decisions that are often subject to change. These design decisions are reflected in the implementation of classes which encapsulate both data and behavior. When programmers can implement against more stable interfaces, the risk of having to change source code afterwards due to changes in the dependent classes is minimized. In the ontology model, the behavior of classes is not defined. While inheritance is supported in both, it is more comprehensive in the object model. Derived object classes inherit attributes as well as the explicitly coded behavior (methods). This inheritance of behavior is quite powerful with the concept of polymorphism and dynamic binding. In this regard, polymorphism means that subclasses can overwrite the inherited behavior of their parent classes. With dynamic binding, the subclass can exhibit a type-specific behavior (the one of itself or of a particular parent class) depending on the type specified in method calls at runtime. In contrast to the object model, concepts only inherit the properties of parent concepts.

In conclusion, it can be said that both models represent a domain of interest albeit with different intentions. The object model has more advanced concepts for the specification of behavior to accomplish the data processing required for an application. Ontologies target at modeling of general knowledge about the world. Ontology languages such as OWL provide more expressive constructs that fit better for this

TABLE I
COMPARISON ONTOLOGY MODEL VS. OBJECT MODEL

	Ontology Model	Object Model
Identity of Individuals	URIs	OIDs
Object Types	Concepts	Classes
Object Characteristics	Properties	Attributes
Object Behavior	not defined	Methods and Interfaces Information Hiding Encapsulation
Inheritance	Properties	Attributes Behavior (Polymorphism) Dynamic Binding

task, like set operators for complex class expressions, property characteristics and the various types of restrictions. Because ontology models and object models serve different purposes, they are complementary, e.g. where advanced data processing is required, the object model may be chosen, whereas ontologies shall be used for expressive knowledge modeling.

C. Final remarks on ontologies

The use of the term ontology varies from strict scientific definitions in practice. Ontologies used in many applications do not comply with the discussed definitions and in fact, do not need to be shared and highly formalized to be useful. Conversely, the comparison with other models shows that advantages indeed are results of these characteristics. There is the commonly known trade-off between expressiveness and performance of ontologies. It has been shown that semi-formal ontologies with limited expressiveness scale in real-world applications, are highly-performant and require less development effort because they can be populated automatically by agents [13]. However, many scenarios simply require a sufficient degree of formalization and therefore highly axiomatized ontologies, e.g. for consistency checking and classification of proteins [14].

IV. ONTOLOGY-ENABLED INFORMATION SYSTEMS

A. A Semantic Extension of the Three-Tier Enterprise Application Architecture

In the following, our architecture framework for the development of semantic applications will be presented. Essentially, it builds upon the well established architecture for enterprise application development which consist of the Client Tier, the Middle Tier and the Resource Tier. Applications at the Client Tier request for services provided by Middle Tier components of the Web Layer (Thin Client) or components of the Application Layer (Fat Client). These components may make use of information supplied by the Resource Tier. More precisely, the components use Data Access Objects (DAO), which are employed to give transparent access to and minimize dependency from the underlying resources. Fig. 1 implies that the three-tier architecture (see [15] for more details) does indeed not change in structure. Our semantic extension simply

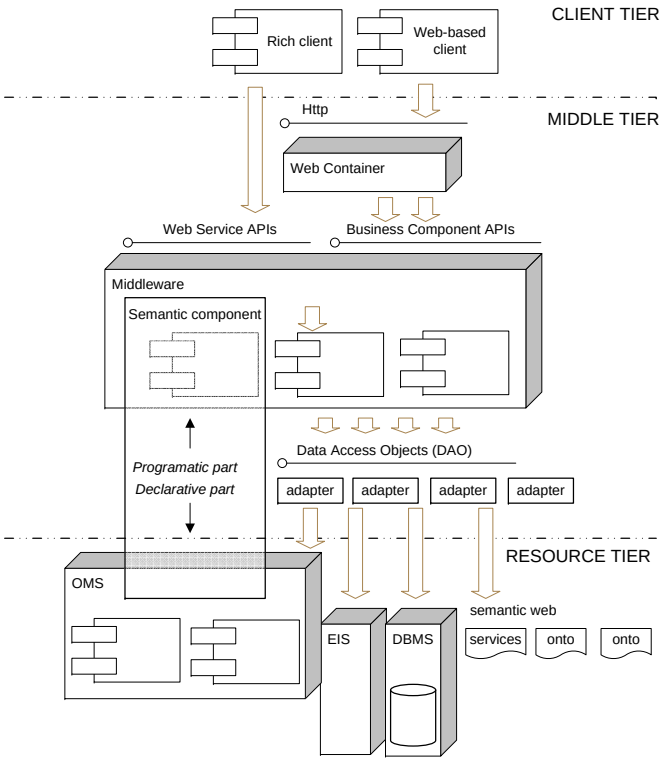

Fig. 1. Three-Tier Architecture for Semantic Applications

adds the infrastructure for ontology management called the Ontology Management System (OMS) that is essential for semantic applications.

The following sections discuss functionalities of the OMS and issues that arise in the development of OMS-enabled components. In principle, these are not completely new ideas but rather insights gained from the work on OMS—in particular KAON2—and the design of semantic applications. The basic functionalities presented correspond to features commonly provided by existing OMS. Also the advanced functionalities simply reflect features we are working on (versioning, mediation) or are targeted by our industrial partners (data integration). Nevertheless, these results and the subsequent guideline for the development of semantic components are of generic nature, i.e. abstract away from any specific OMS and KR languages.

B. The OMS of the Resource Tier

The OMS ensures the management of ontology schema and individuals (also called facts), which in the following will be referred to as "semantic" data as opposed to the mere "syntactic" data managed in databases. Current OMS such as Sesame and Jena (for RDF-ontologies) and KAON2[1] and OWLIM (for OWL-ontologies) rely on the persistence mechanism of an underlying database ontology. This means that semantic data is mapped to physical database tables for the final storage.

[1] http://kaon2.semanticweb.org/

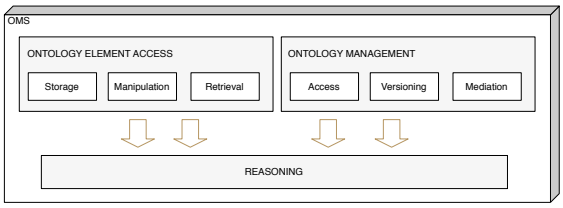

Fig. 2. Functionalities of the Ontology Management System

1) Basic Functionalities of the OMS: In essence, an OMS can be seen as a repository for semantic data. As shown in Fig. 2, this semantic repository provides basic ontology management functionalities such as the import and export of ontologies (referred to in the figure as Access). Also, there is a number of classes and interfaces providing functionalities to manage the elements of an ontology as exemplified in Fig. 3. It shows the five most essential classes and respective methods for working with ontologies which cover basic functionalities commonly provided by the above mentioned OMS. The objective is to illustrate the access mechanism essential to an OMS while in reality a specific OMS may have different and many more classes.

These classes support the storage, manipulation and retrieval of ontologies and ontology elements, namely Concept, Individual, Property and Axiom. For instance, they can be used to update the property value of an individual, to get its type, or to retrieve member individuals, subclasses or superclasses of a class. Note that in addition to accessing instances, the classes Concept, Property and Axiom actually support access and manipulation (setter methods not shown in the figure) of the schematic component at runtime, which is not possible with a DBMS.

Apart from these repository functionalities, an OMS may also provide services for the management of ontology evolution. Ontologies need to be updated to cope with changes in requirements and thus require a mechanism for versioning. Moreover, if the ontology is also intended for external use, like for data exchange among partners or within a community, the need for change management is even higher. For the development of domain ontologies in particular, services for mediating mismatches in understanding about the domain, i.e. for mapping and alignment of concepts, have to be provided.

Note that all above-mentioned services can make use of the reasoning service provided by an inference engine of the OMS. For retrieval operations, this engine returns, in addition to asserted facts and axioms, also the entailed information which is derived according to semantic entailments of the respective KR formalism. For instance, when querying all member individuals of a class, instances that are explicitly asserted to belong to this class, as well as instances with property values that fulfil the sufficient conditions specified for class membership, are returned (supported by OWL but not RDF). Among other uses, reasoning can be employed to support the resolution of version conflicts and mismatches among concepts.

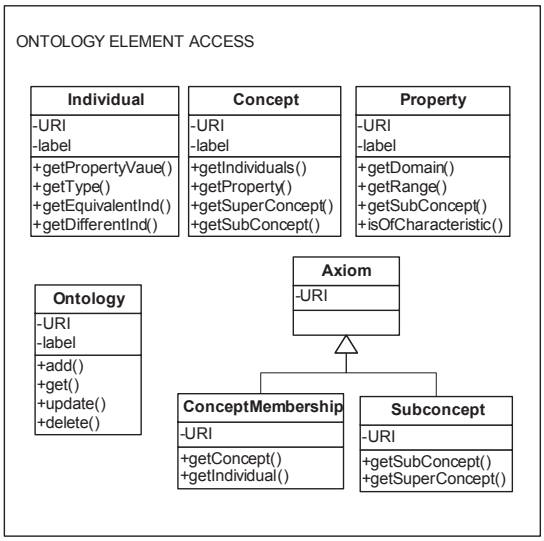

Fig. 3. Ontology Element Access API

2) OMS as the Universal Access Mechanism to Data Sources: Basically, the OMS API introduced above allows the management and particularly the retrieval of both the schema and instance information that are designed and populated by the enterprise. The knowledge engineers model ontologies to capture the specific knowledge needed by the application which could describe both the data (in the sense of a database schema) and the logic to execute tasks in the form of rules.

With the advance of the Semantic Web, there are and will be many more publicly available domain ontologies. They can be reused and extended by the application ontology to maximize interoperability with external systems and minimize efforts in ontology development. Fig. 4 demonstrates such a scenario where the OMS manages an application ontology consisting of the concepts `Content`, `User` and `Process` and relations like `has part`, `has interest` etc. When populated with instance information, the OMS can provide components at the domain layer with the knowledge needed to deliver personalized content and services to the user (the modeling of the actual ontology and details of the respective components will be discussed in the subsequent section). Additionally, the OMS is responsible for the management of relevant external resources that are available on the SW stored as RDF or OWL files and can be identified by fixed URIs. The example shows that the application ontology in fact makes use of a domain ontology called ODAS which provides the shared conceptualization for the exchange of knowledge among adaptive systems. In particular, information related to the user, the content and other concepts can be exchanged and interpreted by cooperating adaptive systems on the web.

Besides, the Mappings shown as part of the OMS in Fig. 4 hint at a current trend to employ the OMS as a single point of access to the heterogeneous data sources of the enterprise. These mappings can be seen as knowledge about

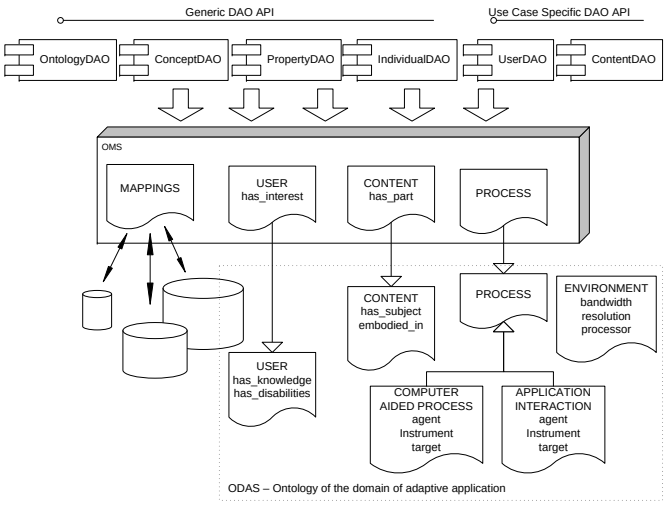

Fig. 4. OMS—Accessing Heterogeneous and Distributed Data Sources

schema translations that can be expressed in terms of rules. On the basis of this thereby specified knowledge about the translations, adapter classes can be implemented to integrate a specific data source, e.g. to wrap a DBMS in order to provide access to all its data. Access to data sources thus becomes independent of their location and the nature of the knowledge being accessed.

C. Ontology-Enabled Components of the Middle Tier

Most of the processing performed by an enterprise application is handled by components at the application layer. In particular, a component can be seen as a grouping of operations, entity types and data types which together implement a set of similar use cases. In object-oriented approaches, a component can be formed by a set of Services, Entity Classes (and Entity Manager). The Service Class consists of a set of operations (methods) which encapsulate the processing logic. The operations involve domain entities encapsulating the data. These domain entities are represented by Entity Classes which provide methods such as "Getters" and "Setters" to operate on the data of one concrete entity instance.

There are many differences that need to be addressed when an Entity Class is employed to encapsulate the data of the OMS. Firstly, there is a design choice. In fact, only five classes are required to access all ontology data, namely Ontology, Axiom, Concept, Property and Individual (see OMS API above). This corresponds to the dynamic object model concept where the actual object types are identified by names only [16]. Therefore it is flexible but lacks type safety and thus results in code that is harder to maintain and test. Instead, custom tailored classes can be used which can be generated from concepts and properties of the ontology using tools like RDFReactor [17]. Methods can be attached to these classes leading to object-oriented design and so, enable object-orientated access and manipulation. The recommended way is obviously a combination of both. That is, having two APIs where one is used for generic ontology access and the other for object-orientated manipulation of ontology elements. However, this may be not reasonable in scenarios where the ontology is manipulated at runtime such as in an ontology modeling application. When the ontology evolves, the generated classes may run out of sync and thus need to be taken care of.

Secondly, changes to data encapsulated by the Entity Class are not only propagated top down but also bottom up. Typically, changes result from user interactions and subsequent processing at the Middle Tier which are finally made persistent in the Resource Tier. When domain knowledge is conceptualized using ontology languages with high expressivity like OWL, the therewith populated semantic data may lead to inferences of new data. For instance, when property values of a particular instance change, also its class memberships may change as the result of a classification performed by the inference engine. While this is desirable when these inferences are deliberately exploited, it requires appropriate synchronization with the OMS data.

Semantic technologies also bear impact on the Service Classes. Processing logic can be both programmatic and declarative. In the former case, the logic is coded in a procedural or, more often, in an object-oriented manner using constructs of the respective programming languages. In the latter case, the logic will be specified in the form of rules. While SWRL [18], the current rule language recommended by the W3C, does not have this degree of expressivity yet, more advanced languages proposed by the RuleML initiative[2] suggest that in the future also highly expressive rules can be specified following the style of Logic Programming. Consequently, components of semantic applications can be coded fully programmatic, fully declarative or as a combination of both. While object-oriented code is processed at the Middle Tier, the declarative part of the component will be stored in and processed by the inference engine of the underlying OMS. Combining both worlds leads to an even higher amount of data inferred by the Resource Tier that has to be synchronized with the Middle Tier.

V. A Personalized Knowledge Portal Prototype

A. The adaptive Functionalities of the Portal

This section presents a prototype of a personalized knowledge portal we created using the proposed multi-tier architecture framework. It is based on an open source portal framework called Liferay[3] and can be downloaded at `http://ontoware.org/projects/xxplore/`. As shown in the screenshot in Fig. 5, our Liferay-extension encompasses a module for the generation of recommendations, a module for the presentation of system resources, a search module for content retrieval and a navigation component that illustrates the adjacencies of the currently active content unit. The adaptive functionalities will be discussed here and the first two modules making up this behavior will be used to demonstrate

[2] `http://www.ruleml.org/`
[3] `http://www.liferay.com/web/guest/home`

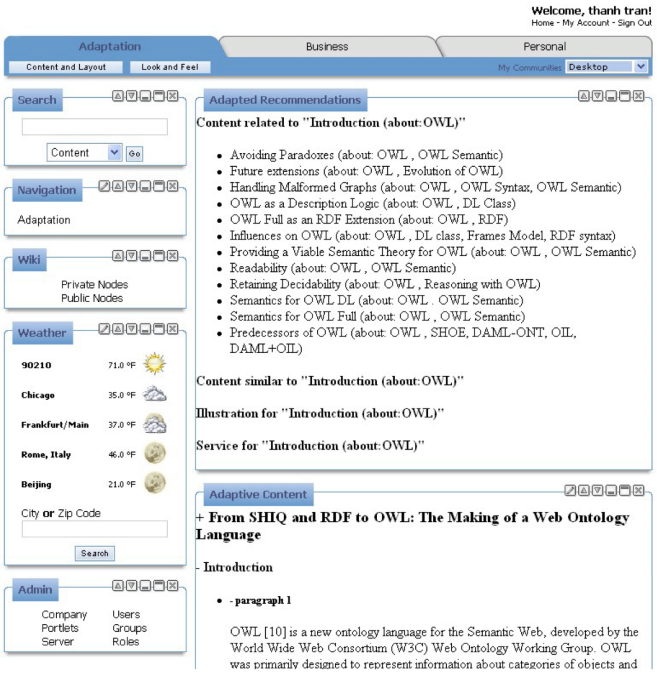

Fig. 5. A Personalized Portal Prototype

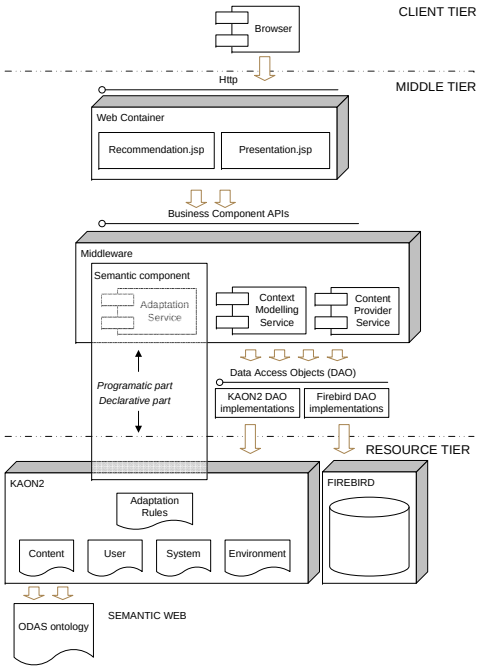

Fig. 6. The Personalized Knowledge Portal Architecture

the development of ontology-enabled applications on the basis of the proposed architecture in the following sections.

Our portal is able to adapt content resources to the present needs of the individual user. As shown in Fig. 5, the content is presented in the Adaptive Content module. It shows a scientific paper with the title "From SHIQ and RDF to OWL: The Making of the Web Ontology Language". This has been presented because the user had activated the corresponding link in the Recommendation module before. This module generates a list of content units that have been assessed by the system as being relevant with respect to the current user context. These recommendations are based on multiple contextual dimensions which are represented using concepts of an ontology. The content resource the user currently interacts with is such a dimension. As the content currently shown in the Adaptive Content window describes the entity `OWL`, which is a section of the paper mentioned above called "Introduction", recommendations shown encompass content describing relationships and processes in which this entity is involved such as "Avoiding Paradoxes" or "OWL as a Description Logic". As discussed later, this means the system makes recommendations about semantically related content.

B. The Architecture of the Portal

Our portal prototype adds semantic technologies to the skeleton architecture provided by the Liferay platform. Fig. 6 illustrates the architecture with a focus on our extension to Liferay. This includes KAON2 as the OMS, DAO's implementation as adapters to KAON2 and Firebird, and a number of components for the realization of the functionalities discussed previously.

The main tasks of components at the web layer are the assembly of content units retrieved from lower level components and the propagation of interaction information to them. At the application layer, the Content Provider Service delegates the task to the corresponding DAO to retrieve the requested resources. The Context Modeling Service computes contextual information on the basis of information propagated down from the web layer, including the user, the resources the user interacts with, and the application and information related to the client system. This information is represented by "tailored" Entity Classes, e.g. Content, User, System and is stored in the OMS using corresponding DAO implementations for KAON2. The use of these classes is possible as the service simply adds attribute values to these classes that are turned into facts in KAON2 and so no changes to the ontology schema are required at runtime.

KAON2 is an infrastructure for the management of OWL-DL, SWRL, and F-Logic ontologies. Fig. 6 shows this OMS manages the application ontology containing concepts for the representation of the context like Content, User etc. as well as adaptation rules. Also, it imports the ODAS domain ontology which has been used for many descriptions contained in this application ontology. Committing to such a domain ontology this way facilitates the exchange of semantic data, e.g. data about Content, User, with external adaptive systems committing to the same conceptualization. Note that semantic data is managed by KAON2 but the content resources are stored in the Firebird database[4]. While all data can be stored in KAON2, employing an OMS only for semantic data prevents an unnecessary increase in the size. KAON2 is relatively high-

[4] `http://www.firebirdsql.org`

performant [19], but query processing has still polynomial complexity and thus an increase in size bears a negative impact on performance.

So far, the instantiation of the proposed architecture has been discussed. The following sections elaborate on the ontology used for the representation of contextual information and finally demonstrate how the adaptation component exploits this ontology to provide adaptive functionalities.

C. An Ontology for the Representation of Adaption Context

The ontology for the domain of adaptive systems (ODAS) we have developed and present here aims to provide a conceptualization of concepts and relations that are relevant for the adaptation of hypermedia resources to the user context. In principle, adaptation can be seen as a process where the resources provided by the system are matched to the user's needs. This section illustrates the use of ODAS to capture the system's resources and to represent the context. The following section then discusses a number of rules that can infer the user needs from this context and perform adaptation.

Central to the representation of the adaptation context is the notion of `Process`. `Application Interaction` for instance, tells the system that a particular `User is involved` (*user model*), and currently, is interacting with a `Content` resource (*domain model*) of the `Application` (*system model*) to accomplish a task. Indirectly, this task is captured as a `Composite Process` of which the atomic interaction `is part of` (*task model*). That is, we employ a process-orientated representation of tasks. The workflow required to accomplish the task is modeled in the system as a `Computer-aided Process`. The `output` of this process can be implicitly assumed to be the user's objective so as to make adaptation goal-directed.

Note that recorded `Application Interactions` contain information about the `Content` currently processed by the `User`. Since `Content` types are distinguished by the subjects they describe, content-based adaptation rules can be used to trigger recommendations suggesting users to navigate to related content units of a different type. A special type of `Content` is `Executable Content`, which differs from others in that it `is embodied` in a `UI Element` and `is representation` of a `Service`. In this way, any service can be indirectly represented as individual of `Executable Content` and adapted to the `User` in the same way as other content types. Also, we introduce the concept of `Content Bearing Object (CBO)` to distinguish the actual materialization from the abstract `Content` embodied in it. Thereby, concepts become available for the representation of layout- and presentation-related requirements (*presentation model*).

Further concepts that deliver contextual information are `User` and `Environment` and their properties.

D. A Semantics-Enabled Component for Adaptive Behavior

Components making use of semantics can have both a programmatic and a declarative part. Much of the logic of the adaptive component presented here is declaratively specified by the use of rules. These so-called adaptation rules make use of concepts and relations reflecting the different adaptivity dimensions as discussed in the previous section. In general, atoms in the rule body capture the conditions that need to be fulfilled for the recommendation stated in the head to apply. As shown in Fig. 7, conditions in the rule body are split up into three blocks, the context-related, adaptation-related and constraints-related category.

The first block ensures that the right context is given. This is represented by the concept of `Process`, which serves as the "entry point" to access various adaptivity dimensions, i.e. `Content`, `User`, `Application`, `Environment` and indirectly also the task via `Computer-aided Process`.

In the second block, adaptation is performed on the basis of the semantics of the `Content` concept. Given the `User` is `Reading` a content unit—which in Fig. 7 is represented by the variable `c` that the inference engine tries to bind to an individual of `Content about Entity`—the rule leads to the recommendations of related resources. Due to the semantics of `Content about Entity`, which has a particular entity of a domain ontology as subject, resources can be considered as related if they have the same entity or related entities as subjects. The rule given in Fig. 7 captures the semantics of the latter, that is, recommends resources (represented by the variable `rec`) which have subjects in common that are related entities in the ontology. So, if the user is reading `Introduction` (OWL), which is `Content about Entity` describing `OWL` (entities in brackets stand for the subjects), then recommendations would include `Predecessors of OWL` (`OWL`, `SHOE`, `DAML-ONT`, `OIL`, `DAML+OIL`) and `Future extensions` (`OWL`, `Development of OWL`) for instance.

This is simply an illustration of how the second block can be instantiated to yield adaptive behaviour. Many further examples for content-based and also task-based adaptation are discussed in [20]. Eventually, these styles of adaptation yield a set of resources related to the current one. The last category consists of conditions acting as constraints that, when applied, have a minimizing effect on this adapted set. The example shows how user information can be used to restrict the set of related resources to a set for which the user has credentials. Other user characteristics as well as environmental information can be applied in the same manner. When all these conditions are met, a rule fires and recommends content, for example individuals of `Content about Entity Relation` as shown in Fig. 7.

When using adaptation rules this way, the programmatic part of the adaptation component stays simple. Upon request, it continuously fires adaptation rules until a pre-configured maximum number of recommendations have been generated by the inference engines. It does so by telling KAON2 to run the rules and retrieving the resulting recommendations. As no processing is performed on the data by this component, the synchronization of results is straightforward. The only logic that has been hard-coded is the sequence in which rules are fired. This may be extended to support a more comprehensive

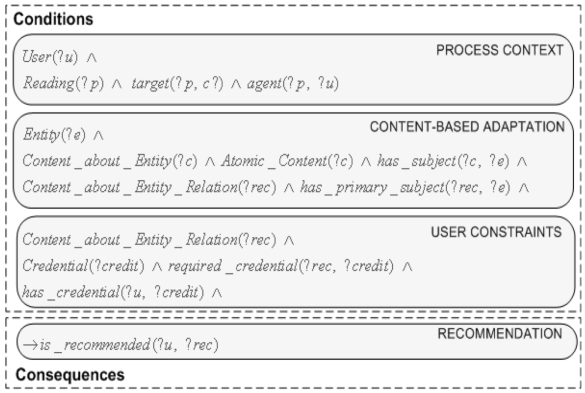

Fig. 7. Structure and Content of an Adaptation Rule

prioritization scheme for the firing of rules.

VI. Conclusion

This paper discusses the role and benefits of ontologies in the context of semantic software engineering. It showed that while being different in intent, ontologies are complementary to XML-Schema, physical Database Schema and the Object Model. Therefore, they will not replace these concepts but rather produce advantages when being combined with them. Our portal could demonstrate the benefits of such additional use of ontologies. By committing to the external ODAS domain ontology, semantic interoperability can be achieved. User model, content model and other data that have been specified with concepts of the ontology can be exchanged and recipient adaptive applications can interpret these data according to the semantics specified by ODAS.

To realize the benefits of ontologies, we extended the state-of-the-art architecture with a management facility called OMS. In the development of components which can exploit such a facility, certain aspects have to be taken into account. Synchronization of middle tier components with information inferred at the resource tier has to be considered. Also, there is a design choice concerning employing "tailored" vs. generic Entity Classes. There is also the choice concerning the extent of processing logic that should be defined declaratively. The logic in the form of rules can be easily reused in other OMS. Besides, rules can be defined in a way that is more generic than if-then statements of object-oriented or procedural languages. This is desirable with respect to some particular use cases, as has been shown for adaption. However, Logic Programming has existed for long time and yet has not proven to be superior to Objected-oriented Programming. Besides performance issues, many types of processing logic just cannot be straightforwardly captured by rules.

In summary, like ontologies, rules have advantages and are complementary to the state-of-the-art and will not likely replace it. Whether and how they can be employed in enterprise information systems has to be examined and assessed case-by-case.

Acknowledgements

Research reported in this paper has been partially financed by the EU in the IST projects: X-Media (ISTFP6-026978) http://www.x-media-project.org and NeOn (IST-2006-027595) http://www.neon-project.org.

References

[1] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Scientific American*, vol. 284, no. 5, pp. 34–43, 2001.

[2] X. Lopez and S. Stephens, "Semantic Data Integration for the Enterprise," March 2006. [Online]. Available: "http://www.oracle.com/technology/tech/semantic_technologies/pdf/semantic_grid_wp_0603.pdf"

[3] G. Schreiber, H. Akkermans, A. Anjewierden, R. de Hoog, N. Shadbolt, W. V. de Velde, and B. Wielinga, *Knowledge Engineering and Management: The CommonKADS Methodology.* MITpress, 2000.

[4] M. A. Musen, "Domain ontologies in software engineering: Use of protégé with the EON architecture," *Methods of Information in Medicine*, vol. 37, pp. 540–550, 1998.

[5] Ó. Corcho, A. Gómez-Pérez, M. Fernández-López, and M. Lama, "ODE-SWS: A semantic web service development environment," in *Proceedings of SWDB'03, The first International Workshop on Semantic Web and Database*, I. F. Cruz, V. Kashyap, S. Decker, and R. Eckstein, Eds., 2003, pp. 203–216.

[6] A. Haller, E. Cimpian, A. Mocan, E. Oren, and C. Bussler, "WSMX - A semantic service-oriented architecture," in *ICWS*. IEEE Computer Society, 2005, pp. 321–328.

[7] H. Knublauch, "Ontology-Driven Software Development in the Context of the Semantic Web: An Example Scenario with Protege/OWL," *1st International Workshop on the Model-Driven Semantic Web (MDSW2004)*, 2004.

[8] M. Brodie, C. Bussler, J. de Brujin, T. Fahringer, D. Fensel, M. Hepp, H. Lausen, D. Roman, T. Strang, H. Werthner, *et al.*, "Semantically Enabled Service Oriented Architectures: A Manifesto and a Paradigm Shift in Computer Science," DERI TR-2005-12-26, Tech. Rep., 2005.

[9] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge Acquisition*, vol. 5, no. 2, pp. 199–220, 1993.

[10] M. Uschold and M. Grüninger, "Ontologies and semantics for seamless connectivity," *SIGMOD Record*, vol. 33, no. 4, pp. 58–64, 2004.

[11] M. Dean and G. Schreiber, "OWL Web Ontology Language Reference. W3C Recommendation," *World Wide Web Consortium (2004) Latest version: http://www. w3. org/TR/owl-ref*, vol. 7.

[12] "XML schema part 1: Structures, W3C recommendation," W3Consortium, May 2001. [Online]. Available: http://www.w3c.org/TR/xmlschema-1/

[13] N. Shadbolt, T. Berners-Lee, and W. Hall, "The semantic web revisited," *IEEE Intelligent Systems*, vol. 21, no. 3, pp. 96–101, 2006.

[14] K. Wolstencroft, P. Lord, L. Tabernero, A. Brass, and R. Stevens, "Protein classification using ontology classification," in *ISMB (Supplement of Bioinformatics)*, 2006, pp. 530–538.

[15] I. Singh, B. Stearns, and M. Johnson, *Designing Enterprise Applications with the J2EE Platform, Second Edition*. Addison-Wesley Professional, June 2002. [Online]. Available: http://java.sun.com/blueprints/guidelines/designing_enterprise_applications_2e/titlepage.html

[16] D. Riehle, M. Tilman, and R. Johnson, "Dynamic Object Model," *Proceedings of PLoP2000. Technical Report# wucs-00-29, Dept. of Computer Science, Washington University Department of Computer Science, October*, 2000.

[17] M. Volkel and Y. Sure, "RDFReactor-from ontologies to programmatic data access," *Proceedings of the International Semantic Web Conference-Demo Session, Galway, Ireland, NOV*, 2005.

[18] I. Horrocks, P. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, "SWRL: A Semantic Web Rule Language Combining OWL and RuleML," *W3C Member Submission*, vol. 21, 2004.

[19] B. Motik and U. Sattler, "A Comparison of Reasoning Techniques for Querying Large Description Logic ABoxes," *Proc. of the 13th International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR 2006), Phnom Penh, Cambodia, November*, 2006.

[20] T. Tran, P. Cimiano, and A. Ankolekar, "Rules for an Ontology-based Approach to Adaptation," in *Proceedings of the 1st International Workshop on Semantic Media Adaptation and Personalization*, Athen, Greece, 2006.