Ontobrowse: A Semantic Wiki for Sharing Knowledge about Software Architectures

Hans-Jörg Happel*) and Stefan Seedorf†)

*) *FZI Research Center for Information Technologies*
Research Group Information Process Engineering (IPE)
Haid-und-Neu-Str. 10-14
D-76137 Karlsruhe, Germany
happel@fzi.de

†) *University of Mannheim*
Lehrstuhl für Wirtschaftsinformatik III
Schloss, L 5,5
D-68131 Mannheim, Germany
{seedorf}@wifo.uni-mannheim.de

Abstract

The development and maintenance of a software architecture involves various stakeholders with different interests. While developers primarily require technical support and guidance for their implementation tasks, architects need means for analysis and documentation. We argue that most approaches and tools insufficiently support the requirements of both groups appropriately, leading to a scattering of architectural information into different information spaces. To resolve this problem we propose Ontobrowse – a lightweight solution which is based upon ontologies and the recent paradigm of semantic wikis. Ontobrowse allows the combination of informal with more formal documentation together with the integration of asserted knowledge from external specification resources. The system architecture and prototypical implementation are described. To illustrate the benefits of our approach, we introduce the case of documenting service-oriented architectures in an enterprise setting.

1. Introduction

A software architecture is the gross organization of a software as a collection of interacting components [11], functioning as a bridge between requirements engineering and system design [11, 29]. Its building blocks are arranged in such a way that both the system requirements and architectural constraints are met [9].

However, it is misleading to talk about *the* software architecture. While every software has an architecture, the actual representations of such an architecture matter. Those representations are a "set of structures" [4] that provide certain "views" such as functional, physical or logical [16]. Most of those views can be assigned a certain purpose, such as quality, communication, analysis or reuse [4, 5, 11].

In this context, two schools of thought can be identified. The developer-centric perspective regards software architecture as a shared mental model [13]. Its purpose is to facilitate communication among the developers by providing a shared understanding of the system under development and enforcing conceptual integrity [29]. In contrast, the system-centric perspective seeks to formalize as much constraints as possible, to allow validation and verification of the implemented system. Various so-called architecture description languages (ADL, see e.g. [19]) have been created to model systems in form of components and connectors, and have formally defined semantics for tool-supported analysis.

Both perspectives are valuable, since the development of software is a process of increasing formalization – from fuzzy user requirements to concrete bits and bytes. As bridge between requirements engineering and concrete design, software architecture has to address both issues. What is sought is thus a solution that offers flexibility in documentation and collaboration as well as a sound formal basis for leveraging machine-

interpretable semantics. Although this requirement sounds contradictory, we are suggesting semantic wikis – an extension of the well-known wiki technology – as a candidate technology for solving this trade-off. From our point of view semantic wikis are well-suited to bridge the gap between technical and business documentation, since they a) provide means for collaborative documentation and information exchange regarding a certain subject and b) provide a formal foundation for handling technical descriptions.

The remainder of this paper is structured as follows: First, we introduce the application scenario of enterprise service-oriented architectures (SOA) to further characterize the problem. Second, we describe the basic ideas and concepts behind semantic wikis. In chapter 2, we present the conceptual architecture and prototypical implementation of Ontobrowse. In chapter 3, it is shown how it can be applied to the aforementioned application scenario by providing a SOA ontology for imposing a knowledge structure and plugins for integrating external specifications. After an overview of related work in chapter 4, we conclude by summarizing the potential benefits of a semantic wiki approach.

1.1 SOA Application Scenario

Before we propose semantic wikis as a solution we first need to delve into the main issues addressed by this work. An application scenario which nicely illustrates the common problems of documenting and maintaining architectural knowledge in enterprises is service-oriented architecture (SOA).

Service-oriented computing [14] is an emerging paradigm which is built on the notion of enterprise application systems being assembled from independent, loosely-coupled services. It has lifted the development of business applications to a higher level of abstraction. Instead of thinking in design and implementation categories like components or objects, software functionality is bundled in services that correspond to business operations of the organization. Complex workflows can be realized by aggregating functionality from simple services. A service is a coarse-grained, discoverable software entity which provides its logically-cohesive business functionality through well-defined interfaces. A concrete software infrastructure implementing this paradigm is described as SOA [14].

In an organization pursuing the realization of a SOA, the standard working processes change for both developers and business experts. Service developers have to think in specification terms rather than taking an implementation view. Due to the black-box realization of services, metadata describing their properties is crucial. Also, the enterprise-wide deployment of services calls for better documentation and communication among the responsible developers. Both aspects lead to new kinds of information needs for developers.

Business experts on the other hand are interested in available functionality and operational efficiency. Since service-orientation leads to a rising level of alignment between business processes and IT implementation, it is important to monitor and guide the development of the service landscape. Governing the evolution of a service-oriented architecture becomes important, because changes at the service level may have a direct impact on business processes.

So the paradigm of service-orientation ties the individual workflows of software developers and business experts much closer together. This situation of multiple stakeholders results in diverse requirements for tool support. Besides managing the appropriate technical and business aspects this includes communication and documentation among the various participants involved in the development process. Moreover, different tools and description formats may be used.

This heterogeneous, dynamic environment motivates the two building blocks of our application. Unlike tools that concentrate on specific aspects such as service orchestration or lifecycle management, we intend to provide an integration space for both developers and business experts, since both worlds often maintain their separate information set about the same subject. Therefore we propose a semantic wiki approach which is able to integrate both informal and formal descriptions typically managed within a SOA project.

1.2 Semantic Wikis

The software genre of a “Wiki” describes a lean approach to web-based content management, allowing multiple users to collaborate on the creation of a document. Basically, a “small web” is imposed by the titled wiki pages and their contained hyperlinks. However, in contrast to the World Wide Web, wikis provide editing capabilities and enforce some conceptual coherence. Due to these characteristics, wikis have become popular in various application areas, in particular software engineering (see e.g. [7] for an overview).

Although traditional wikis provide a top-level structure by its separation in “pages”, the actual information on a certain page is stored in an unstructured manner. Classical wiki software does not provide a standardized way to add structured information to a certain topic. However, if some information was made available in a machine-interpretable format, a site like the Wikipedia could heavily benefit because its pages contain a lot of potentially structured information [21].

Thus, several projects started to implement semantic extensions to the wiki approach[1]. Because most of the implementations are still in an experimental stage, there is no clear definition of what a semantic wiki exactly is. However, some shared characteristics can be identified: Semantic wikis extend traditional wikis in the way that they allow structured knowledge to be described in a formal language, instead of processing solely hypermedia-based content. This is either be done by appending metadata to wiki pages or by including knowledge inside the unstructured text by using extensions to the wiki markup language. The latter approach is used by the SemanticMediaWiki project [21], which extends the existing wiki markup to enrich hyperlinks between wiki pages with semantic relations. The approaches have in common that they interpret the existing wiki pages as entities, and hyperlinks as relations among them. Adding semantics just formalizes this implicit structure, thus transforming the knowledge inside the wiki into a kind of "ontology", which is defined as "an explicit specification of a conceptualization" [12]. Since most semantic wiki implementations are rooted in the semantic web community, they adopt existing standards for ontology representation such as the Resource Description Framework (RDF) and the Web Ontology Language (OWL). RDF is a simple graph-like format for describing metadata about resources [22] that are described using a Uniform Resource Identifier (URI). This makes it easy to annotate wiki pages with arbitrary metadata. OWL is defined on top of RDF(S) and provides a standard ontology vocabulary for describing ontologies based on description logics [23].

While the knowledge representation community typically differentiates between structural knowledge, describing concepts or classes ("TBox") and assertional knowledge, describing instances of those concepts ("ABox"), this distinction is rather blurred in current semantic wikis. For application scenarios like formalizing the knowledge inside Wikipedia, this is perfectly suited, since "conceptual" knowledge may emerge as well as "assertional" knowledge.

However, in a more focused application scenario, a more restrained conceptual structure, specifying concepts and their relations, might be useful to predetermine the initial structure of the information space. In opposite to semantic wikis supporting the emergence of knowledge structures out of existing wiki content, this alternative proposal results in a semantic wiki based on a predefined knowledge structure, that provides a frame for adding further assertional knowledge inside a limited domain. We consider this approach to be viable for the purpose of bringing together formal specification with informal documentation of architectural knowledge. We will now describe the Ontobrowse semantic wiki and then illustrate its contributions by proposing an exemplary knowledge structure for the SOA scenario.

[1] http://wiki.ontoworld.org

2. The Ontobrowse Semantic Wiki

There are various kinds of tools and description formats that can be constructed for sharing architectural knowledge. The main purpose here is to provide a lightweight solution that can be adapted to integrate information from existing environments, e.g. service descriptions managed in a SOA repository.

On top of the problem description in the chapter 1 three main requirements are identified:

R.1 Browse, search and query architectural knowledge
R.2 Manage (informal and formal) documentation
R.3 Enable consistency checking and verification

Although R.3 is not discussed in this paper, the proposed ontology-based solution provides general support for it. Moreover, there are two key constraints for a practical solution:

C.1 Combine formal and informal knowledge of a software architecture
C.2 Enable the integration and augmentation of knowledge from external sources

Semantic wikis make it possible to combine unstructured and machine-interpretable knowledge (C.1), and ontologies define a knowledge structure, which can serve as a contract for integrating instance data from external architectural description resources (C.2). Both semantic wiki and ontologies therefore constitute the building blocks of the Ontobrowse architecture.

2.1 Wiki architecture

The core of our wiki architecture consists of one or more ontologies and a corresponding knowledge base. While the ontologies define the knowledge structure, i.e. the boundaries in which instances can be described; the knowledge base holds the instances. Within the given knowledge structure, it is possible to create or augment instance descriptions in two different ways: First, external tools can plug into the wiki application and map architectural description resources to instances in the knowledge base. Second, a wiki user can use the interface to describe properties – may it be formal or informal – about instances of concepts.

In our SOA scenario, a knowledge structure may be described by the concepts "service" and "business object" together with their properties and axioms. Another example is a concept in a domain vocabulary, which can be used to describe SOA elements with additional

semantics. The instances are represented by actual services and business objects developed in a SOA project. Each concept, relation or individual is visible to the user as a "wiki page". A wiki page typically consists of unstructured content and properties that make statements about this page, e.g. a business object which is semantically described by a domain concept. We also refer to a wiki page as an "entity", because it is contained in the knowledge base and can be requested with a unique identifier (URI).

Following the characterization of the wiki structure, we now introduce the key components of the system architecture as depicted in Figure 1: a Web interface, a wiki manager, an ontology API to access the knowledge base and a plugin manager.

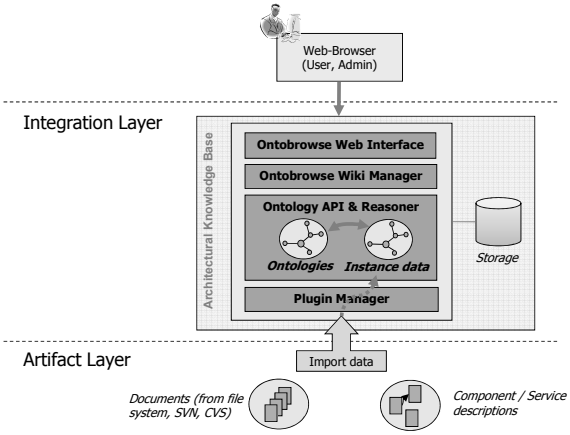

Figure 1: Ontobrowse architecture

Starting with the client's perspective, all functionality is exposed by a Web interface to both users and administrators. In the application layer, a wiki manager bundles the functions for fulfilling the requirements, such as processing page requests, editing textual documentation and instance property values, searching and deductive querying, and verifying user authorizations. Entity (page) descriptions are returned by an ontology API, which wraps the underlying reasoner and ontology processing tools.

Administration tasks include ontology management and plug-in management. In most cases, ontologies are constructed during the setup phase using an ontology editor such as Protégé[2] and then uploaded by an administrator using the wiki manager. However, it is still possible to add new properties to concepts or entire ontologies throughout the operation phase.

Plug-in management refers to a distinctive feature, which allows mapping and importing instance data from external sources. To a great extent the instance data will be embodied in applications and artifacts that are managed outside the wiki, e.g. service specifications in the case of SOA documentation. The data thus has to be imported from external sources, such as configuration management systems (see Figure 1). Therefore, the plug-in manager exposes standard interfaces that allow tools to retrieve artifacts, map them according to an ontology, and create or update instance data in the knowledge base. As described in more detail in the SOA case, imported instance data can be augmented and referenced for further documentation.

2.2 Prototypical Implementation

The system architecture described above was implemented in Java. Our main concern was to achieve a clean separation into loosely coupled components so that some parts can be easily substituted by other implementations, e.g. using a different reasoner in the ontology API. The application and persistence layer were implemented with Spring[3] and Hibernate[4], for the Web interface we used Java Server Faces[5].

The Web Ontology Language (OWL) was employed as knowledge representation format. For processing OWL files and reasoning in the ontology API we worked with the Jena 2 Semantic Web framework[6]. Deductive queries with SparQL [25] are supported accordingly. Jena also supports user defined rules for knowledge generation and consistency checking, however, this feature is not discussed in this paper. A feature that had to be added on top of the ontology API is full text search spanning both entity descriptions in the knowledge base and textual descriptions.

One important aspect for the user acceptance of a Wiki is an easy to understand Web interface. If the wiki contains hundreds of concepts, the knowledge space will quickly get too difficult to navigate. That is why administrators can assign an OWL annotation property to a concept in order to mark it as visible on top-level. Moreover, annotations have been used to control editing of instance properties. In the SOA case, a service instance could be augmented with additional metadata, e.g. about the responsible developer. Instance properties that have been imported via plug-ins are thus marked as "non modifiable". This way (one-way) consistency with external sources is ensured.

[2] http://protege.stanford.edu

[3] http://www.springframework.org/

[4] http://www.hibernate.org/

[5] http://java.sun.com/javaee/javaserverfaces/

[6] http://jena.sourceforge.net/

3. Application to the SOA scenario

The primary goal of Ontobrowse is to provide a non-invasive solution, which can be extended and tailored to individual project needs. Depending on the enterprise setting project-specific architectural knowledge may be distributed throughout various sources.

Concerning the SOA scenario we assume that architectural descriptions are managed in a single file system or alternatively in a service repository. A typical candidate artifact is the specification of a service together with its interfaces. The services may either be specified in a custom or standardized format, e.g. proprietary XML or Web Service Description Language (WSDL). Other artifacts also hold valuable knowledge. A SOA project may use a considerable number of WS* standards, e.g. Business Process Execution Language (BPEL4WS), WS-Policy or WS-Security, to cover important aspects such as service orchestration and non-functional properties.

Two steps are necessary for integrating architectural descriptions into the wiki: First, we require a conceptual mapping from a description format to a unifying *SOA ontology*. Since more than one format might be used to describe a "service", the ontology reduces conceptual ambiguity and enables information integration. Second, a *plug-in* has to be defined, which performs the actual mapping of instances from a source into the knowledge base.

3.1 SOA ontology

The purpose of the SOA ontology is to supply the initial structure to the semantic wiki enabling the documentation of services by both business experts and developers. For this reason, it has to include central concepts of a SOA like services, business objects and domains concepts. Moreover, the ontology should provide a basic abstraction, in which the actual information about SOA elements from the external data sources is mapped onto.

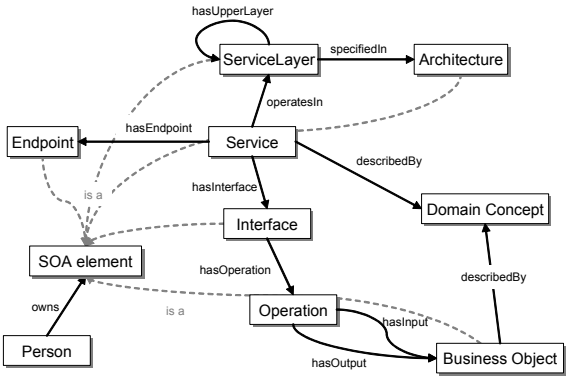

Figure 2: SOA ontology

There exist a number of specification standards and ontologies which provide valuable input during ontology development. For example, the service component model in WSDL 2.0 is partly reflected in the ontology. Nevertheless a strict ontology mapping proposed for WSDL [26] is not applicable here because it would lead to scattering of the page structure. Other useful sources are the foundational ontologies being developed within Semantic Web Web Services [2, 10, 20] and Web services architecture [24].

The SOA ontology is visualized in Figure 2. The presented ontology is generic in the way that it incorporates common characteristics of widely-accepted standards. Figure 3 shows an example wiki page returned for the concept "SOAElement" with its direct subconcepts. Other information displayed for a concept are its instances (or individuals), object properties (e.g. "hasInterface") and data type properties (e.g. "version").

Figure 3: Concept hierarchy of SOA Element

It is possible to develop additional ontologies that cover a particular information need in the SOA project or organization. Instance data corresponding to the ontology may either be maintained in the wiki or imported from external sources by creating appropriate plugins. This ensures high flexibility and enables to augment SOA elements with further knowledge. This may e.g. include domain concepts given by a domain ontology or organizational knowledge such as persons responsible for SOA elements.

3.2 Mapping SOA artifacts

We now explain how actual service descriptions are imported into the wiki and enriched with additional metadata. WSDL 2.0 service descriptions serve as an example, while the process is analogous for other formats and source types. First a one-way mapping between WSDL service descriptions and the SOA ontol-

ogy has to be defined. We extended the WSDL format to accommodate additional service properties such as version and architectural layer. The actual mapping is executed by a Java program which conforms to the Ontobrowse plug-in interface. It takes a WSDL file as input and produces an OWL file conforming to the SOA ontology. A wiki administrator is then responsible for configuring input sources (CVS, file system) and update types (manual, timer task, update event). Based on this configuration the plugin manager component is responsible for updating the knowledge base automatically.

Figure 4: Service instance

Figure 4 shows a service instance imported from a WSDL file. Wiki users can now browse and search the information space, edit textual descriptions and assign additional property values (metadata) to an instance, e.g. the responsible person for a service. However, editing is restricted to especially annotated properties. Moreover, these property values are exclusively visible *within* the wiki.

A query interface enables users to define chained queries consisting of sentences with *subject*, *predicate* and *object* (e.g. all services "x" defining interface operations with the output "Customer"). Matching entities are returned for the variables defined by the query.

4. Related Work

The basic building blocks of our approach – wikis for software documentation and ontologies as formal models for software systems – have been used in some earlier works. Aguiar and David present a wiki-based approach to integrate heterogeneous software specification resources into a single document [1], while Bachmann and Merson investigate the advantages of wikis compared to other architecture documentation tools [3]. However both approaches lack a formal model – the information is managed in an unstructured way.

On the other hand, formal ontologies have been presented for architectural documentation [28] and for building "software information systems", describing the interrelationships of domain models and source code [8, 27]. These works either lack appropriate tool support or follow a very strict philosophy of software architecture.

Some aspects of a software architecture that have been covered in this paper might also be described using an ADL [19]. However, the case described in this paper substantially differs from the purpose of architecture description languages. Whereas ADLs solely focus on the formal specification of concrete architectures, our approach is also capable of supporting informal aspects of software architecture. Our model of architectural description allows exchanging knowledge between different architectures and is not limited to a fixed set of language elements. It may be extended to support further architectural dimensions such as organizational issues or requirements traceability.

The notion of "architectural knowledge" is currently discussed in terms of representing decisions in the architecture development process and the "rationale" behind them [15, 18]. While our tool is flexible enough to incorporate such information (e.g. based on [17]) it was not the focus of our underlying use case. Thus, the interpretation of architectural knowledge in the context of our work is a much broader one, incorporating unstructured textual documentation as well as formal architectural specifications.

5. Conclusion and Outlook

In this paper, we presented semantic wikis as a novel approach to share knowledge about software architectures. The semantic wiki architecture and functionality was illustrated by the example of service-oriented architectures. However, the selected application scenario can easily be generalized to support arbitrary architectural styles. It can be tailored to project-specific needs by providing an ontology to set up the initial structure of the wiki.

We think that the Ontobrowse semantic wiki approach contributes to several issues in architecture documentation and knowledge sharing. First, it builds upon the general advantages in software documentation offered by wikis. Wikis respond well to collaborative settings and provide a scalable way for the documentation of large software projects.

Second, it bridges the gap between informal documentation and technical service descriptions. We consider this an important issue, since software architectures operate at the intersection of user requirements and system design. Semantic wikis enable an informal style of documentation while also supporting to incorporate machine-interpretable knowledge about technical descriptions. Thus, they allow collecting relevant information about a software architecture at a central place that has so far been maintained separately.

Third, the formal model underlying the semantic wiki supports a better searching and browsing of architectural elements, ensuring semantic consistency and the incorporation of content from external repositories. The formal model can also be easily extended. This may be used to include external knowledge about standards or information about organizational structure, which is particularly important in distributed development settings [6]. Finally, referencing specification and implementation artifacts contributes to improving traceability throughout different stages of the software lifecycle.

6. Acknowledgements

This work was partly supported by the European Commission (IST-35111-TEAM), the BMBF-funded project WAVES and by the Landesstiftung Baden-Württemberg foundation (Project CollaBaWue). The authors are responsible for the content of this publication.

7. References

[1] Aguiar, A., and David, G.: WikiWiki weaving heterogeneous software artifact. In: Proc. of the 2005 international symposium on Wikis, San Diego, CA, 2005, pp. 67-74.

[2] Akkiraju, R., et al.: Web Service Semantics - WSDL-S, W3C Member Submission, 7 Nov. 2005.

[3] Bachmann F., and Merson, P.: Experience Using the Web-Based Tool Wiki for Architecture Documentation. Technical Note CMU/SEI-2005-TN-041. September 2005.

[4] Bass, Len; Clements, Paul; Kazman, Rick: Software Architecture in Practice. 2. Addison Wesley, 2003.

[5] Bosch, J.: Design and use of software architectures: adopting and evolving a product-line approach. ACM Press/Addison-Wesley Publishing Co., 2000.

[6] Cockburn, A.: The interaction of social issues and software architecture. In: Commun. ACM 39, October, Nr. 10, 1996, pp. 40-46.

[7] Decker, B., Rech, J., Ras, E., Klein, B., Hoecht, C.: Self-organized Reuse of Software Engineering Knowledge supported by Semantic Wikis. In: Proc. of Workshop on Semantic Web Enabled Software Engineering, November 2005.

[8] Devanbu, R. J. Brachman, P. G. Selfridge and B. W. Ballard: LaSSIE - A Knowledge-Based Software Information System, ACM Comm., 34(5), 1991, pp. 34-49.

[9] Eden, A.H., and Kazman, R.: Architecture, design, implementation. In: Proc.of the 25th International Conference on Software Engineering (ICSE-03). Piscataway, NJ: IEEE Computer Society, May 3-10 2003, S. 149-159.

[10] ESSI WSMO: Web Service Modeling Ontology (WSMO). http://www.wsmo.org/, 2005.

[11] Garlan, D.: Software Architecture: A Roadmap. In: Proceedings of the 22th International Conference on Software Engineering (ICSE-2000), ACM Press, 2000, pp. 91-101.

[12] Gruber, T.R.: A translation approach to portable ontology specifications. Knowl. Acquis. 5, 1993, 199-220.

[13] Holt, R.C.: Software Architecture as a Shared Mental Model. In: ASERC Workhop on Software Architecture, University of Alberta, August 2001.

[14] Huhns, M.H., and Singh, M.P.: Service-Oriented Computing: Key Concepts and Principles. IEEE Internet Computing, vol. 9, no. 1, 2005, pp. 75-81.

[15] Jansen, A. and Bosch, J.: Software Architecture as a Set of Architectural Design Decisions. In: Proc. of the 5th Working IEEE/IFIP Conference on Software Architecture (Wicsa'05), Washington DC, 2005, pp. 109-120.

[16] Kruchten, P.: The 4+1 View Model of Architecture. In: IEEE Softw. 12 November, Nr. 6, 1995, pp. 42-50.

[17] Kruchten, P.: An Ontology of Architectural Design Decisions. In: Proc. of 2nd Groningen Workshop on Software Variability Management, Groningen, NL, 2004, Rijksuniversiteit Groningen.

[18] Kruchten, P., Lago, P., van Vliet, H., and Wolf, T.: Building up and Exploiting Architectural Knowledge. In: Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture (Wicsa'05), Washington DC, 2005, pp. 291-292.

[19] Medvidovic, N., and Taylor, R. N.: A Classification and Comparison Framework for Software Architecture Description Languages. In: IEEE Trans. Software Eng. 26(1): 2000, pp. 70-93.

[20] OWL Services Coalition: OWL-S Semantic Markup for Web Services. http://www.daml.org/services/owl-s/, 2004.

[21] Völkel, M., Krötzsch, M., Vrandecic, D., Haller, H., Studer, R.: Semantic Wikipedia. In: Proceedings of the 15th international conference on World Wide Web, WWW 2006, Edinburgh, Scotland, May 23-26, 2006.

[22] W3C: Resource Description Framework (RDF), 2004.

[23] W3C: Web Ontology Language (OWL), 2004.

[24] W3C: Web Services Architecture. W3C Working Group Note, 11 February, 2004.

[25] W3C: SPARQL Query Language for RDF. W3C Working Draft 4 October 2006.

[26] W3C: Web Services Description Language (WSDL) Version 2.0 - RDF Mapping, 2006.

[27] Welty, C.A.: Software Engineering. In: Description Logic Handbook, 2003, pp. 373-387.

[28] Welty, C.A., and Ferrucci D.A.: A Formal Ontology for Re-Use of Software Architecture Documents. ASE, 1999, pp. 259-262.

[29] Witt, B., Baker, F. and Merritt, E.: Software Architecture and Design: Principles, Models and Methods, Van Nostrand Reinhold, 1994.