Semantic Management of Middleware

Daniel Oberle
University of Karlsruhe
Institute of Applied Informatics and Formal Description Methods (AIFB)
Knowledge Management Group
D-76128 Karlsruhe
Germany
oberle@aifb.uni-karlsruhe.de
http://www.aifb.uni-karlsruhe.de/WBS

ABSTRACT

The Ph.D. proposal addresses the complexity of building distributed applications and systems with Application Servers and Web Services middleware, respectively. Despite their flexible XML-based configuration, taming the ever growing complexity remains all but an easy task. To remedy such problems, the thesis proposes an ontology-based approach to support the management (i.e. development and administration) of Application Server and Web Services based applications. The ontology captures properties of, relationships between and behaviors of the components and services that are required for management purposes. The ontology is an *explicit* conceptual model with formal logic-based semantics. Therefore its descriptions of components and services may be queried, may foresight required actions, or may be checked to avoid inconsistent system configurations — during development as well as during run time. Thus, the ontology-based approach retains the original flexibility, but it adds new capabilities for the developer and user of the system.

Categories and Subject Descriptors

H.4.m [**Information Systems**]: Miscellaneous; H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*Web-based Services*

General Terms

Management, Languages, Design

Keywords

Application Server, Service Oriented Architecture, Web Service, Ontology, Middleware, Semantic Technology

1. PROBLEMS AND GOALS

This section introduces the usage of semantic technologies, i.e. ontologies and their reasoning algorithms, both in Application Servers and Web Services middleware (1.1 and 1.2). Although the usage of semantic technologies helps to tame the complexity, a new problem arises, viz. the provision of semantic metadata. Semantic metadata are the actual descriptions in terms of the ontology, in our case of both software components residing in an Application Server and Web Services. While some of the semantic metadata can be generated automatically, a full specification will always require manual provision. This is usually a rather cumbersome and error-prone work for the developer. Hence, 1.3 proposes the crafting and usage of well-designed ontology libraries.

Every subsection is divided into problem and goal description as well as a comment on the significance of the research.

1.1 Usage of Semantic Technology in Application Servers

1.1.1 Problem

Application Servers are component-based middleware platforms offering an environment in which users can deploy self-developed or third-party components [1]. As a sophisticated middleware, Application Servers provide functionality such as dynamic loading, naming services, load balancing, security, connection pooling, transactions, or persistence.

Despite the bundled functionality, realizing a complex distributed application remains all but an easy task. For instance, managing component dependencies, versions, and licenses is a typical problem in an ever-growing repository of programming libraries. In Microsoft environments, this is often referred to as "DLL Hell". Configuration files, even if they are more or less human-readable XML, do not provide an abstraction mechanism to tame the complexity issues arising in such systems.

As a way to abstract from many such low-level and often platform-specific problems, the paradigm of *Model-Driven Architectures (MDA)* [16] has gained wide-spread influence. The principal idea of MDA is to separate *conceptual concerns*, such as which component is using which other component, from *implementation-specific concerns*, such as which version of an application interface requires which versions of windows libraries. MDA achieves this separation by factorizing the two concerns, specifying them separately and compiling them into an executable.

Notwithstanding that MDA already provides conceptual modelling in order to improve management of complex systems, MDA is disadvantaged in two ways. First, MDA requires a compilation step preventing changes at run time which are characteristic for Application Server software. Second, an MDA itself cannot be queried or reasoned about. Hence, there is no way to ask the system whether some configuration is valid or whether further components are needed.

1.1.2 Goal

To remedy such problems, one of the goals of the thesis is an ontology-based approach to support the development and administration of software components in an Application Server. The ontology captures properties of, relationships between and behaviors of the components that are required for development and administration purposes. The ontology is an *explicit* conceptual model with formal logic-based semantics. Therefore its descriptions of components may be queried, may foresight required actions, e.g. preloading of indirectly required components, or may be checked to avoid inconsistent system configurations — during development as well as during run time. Thus, the ontology-based approach retains the original flexibility in configuring and running the Application Server, but it adds new capabilities for the developer and user of the system. A first version of the ontology's design is discussed in [22, 21].

In [21] we show how an overall system architecture of an ontology-based Application Server could look like. The left side of Figure 1 outlines potential sources, which provide input for the framework. This includes web and Application Server configuration files, annotated source code, or metadata files. This information is parsed and converted into semantic metadata. Thus, this data is now available conforming to a harmonizing conceptual model. The semantic metadata and the ontology are fed into the inference engine which is embedded in the Application Server itself. The reasoning capability is used by an array of tools at development and at run time. The tools either expose a graphical user interface (e.g. security management) or provide core functionality (e.g. the dynamic component loader).

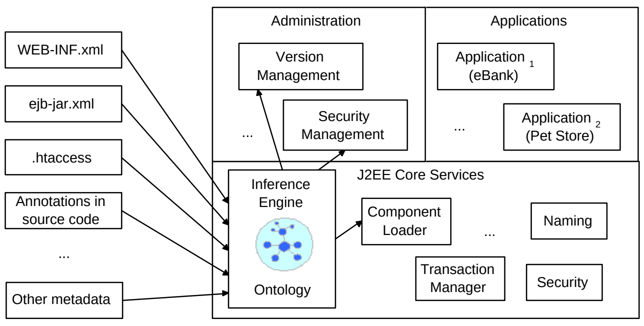

Figure 1: System architecture of the Application Server. Semantic metadata and the ontology are loaded into the inference engine. Value-added services and tools leverage the reasoning capability embedded in the Application Server.

1.1.3 Significance

Given the wide-spread usage of Application Servers and the growing complexity of distributed applications, this will be a significant improvement for the management and administration of such applications.

1.2 Usage of Semantic Technology in Web Services Middleware

1.2.1 Problem

Distributed systems based on Service Oriented Architectures (SOA) factorize the functionality in loosely-coupled and independent services rather than in components like done in Application Servers. The Web-based middleware for such systems is called "Web Services" subsuming a set of protocols and XML-languages for invocation, discovery, and interface description of services.

However, building a distributed system based on a SOA constitutes similar problems to the ones in an Application Server. There are already a multitude of description files (for the interface description [10], for composition[1] purposes [3, 5, 2, 9], for discovery [26] etc.). The complexity will increase by incorporating further aspects like security (WS-Security [4]), transactions (WS-Transaction [8]) and trust (WS-Trust [6]). Existing and currently arising ontology-based efforts like [25, 12, 27] focus mostly on the capability description of a service to improve discovery.

1.2.2 Goal

Therefore, the second goal of the thesis is an ontology-based approach to support the management of Web Services based systems. The ontology is able to capture all relevant aspects (interface description, security information, etc.) of a service in a harmonizing explicit conceptual model that can be queried and reasoned with.

The idea is to leverage the ontology infrastructure introduced in 1.1 to achieve this goal since most Web Services are and will be exposures of the functionality hosted in software components residing in Application Servers. However, the situation becomes more complex compared to Application Servers as the system is distributed over several organizational units. Hence, the research has to take into account additional aspects like network delays, reliability, trust and security issues. Also, new problems have to solved, e.g. the translation of the different vocabularies used, where to do the reasoning, who gathers the data and who gets access to them.

1.2.3 Significance

SOAs are considered as the basis of the next generation of distributed software systems. Hence, the significance and interest in this work will be rather high. Research on Web Services is currently rising with first ontology-based efforts (e.g. [25]) as well as novel conferences and workshops in this area.

1.3 Reusable Ontology Libraries

1.3.1 Problem

The two goals stated above are dependent on the provision of semantic metadata in terms of the ontology describing

[1]Depending on the effort, composition is also called orchestration, workflow, choreography or conversation.

both the components and the services. While some of the semantic metadata can be generated automatically, a full specification will always require manual provision. This is usually a rather cumbersome and error-prone work for the developer.

One reason for that is the lack of ontology quality of existing efforts (e.g. [25, 22]). Like discussed in [7], there are three criteria for the evaluation of ontology quality: *extensional coverage* (concerning the amount of entities that are supposed to be described by an ontological theory), *intensional coverage* (concerning what kinds of entities are described by an ontological theory), and *precision* (concerning what axioms are required to describe just the models the ontology designer intends to cover). According to these criteria, a good ontology should approximate the domain of discourse that is supposed to be described, it should have a signature that maps all the kinds of entities intended by the designer, and it should axiomatize the predicates in order to: 1) catch all the intended models, and 2) exclude the unintended ones.

In [20] we analyzed an existing ontology for the semantic description of web services, called OWL-S [25]. The analysis yielded four shortcomings that are typical also for other ontologies. The first one (conceptual ambiguity) features both insufficient intensional coverage and overprecision. The second and the third (poor axiomatization and loose design) are cases of insufficient precision. In the third problem, the weakness is mainly inherited by limitations of the expressivity of the ontology language. The fourth (narrow scope) is a case of both extensional and intensional coverage.

Conceptual Ambiguity Since there is no clear conceptual framework behind OWL-S, it is often difficult for users to understand the intended meaning of some concepts, the relationship between these concepts as well as how they relate to the modelled services. Many concepts are still being clarified both within the OWL-S coalition and in public mailing lists.

Poor Axiomatization The primary goal of OWL-S is to be machine processable and it operates in an open environment. Hence, it is important that each concept is characterized by a rich axiomatization in order to support meaningful inferences. In contrast to conceptual ambiguity, poor axiomatization reflects the lesser problem when the definition of concepts is clear, but axiomatization in the ontology itself needs improvement. In many respects, OWL-S shows the characteristics of a typical application ontology: there is no firm concept or relation hierarchy (most concepts and relations are direct subconcepts of the top level concept or relation) and several relations take the top level concept as their domain or range.

Loose Design A further problematic aspect from an ontologist's point of view is its entangled design. At the heart of this problem lies the purpose of OWL-S in providing descriptions of various views on Web Services required to support a number of different service related tasks (discovery, composition, invocation). Besides the functional dimension, Web Service descriptions should be contextualized to represent various viewpoints on a service, possibly with different granularity. Most of these views, however, are overlapping in that they concern some of the same attributes of a service. A straightforward modularization in such cases results in an entangled ontology, where the placement of certain knowledge becomes arbitrary and intensive mapping is required between modules. This phenomenon is well described in object-oriented design, where the notion of *aspects* [11] was recently proposed to encapsulate concerns that cross-cut the concept hierarchy of a software.

Narrow Scope Finally, the scope of OWL-S needs to be extended to represent real world services that naturally cross the lines between information systems and the physical world. While OWL-S acknowledges this aspect of services, it is unclear how a distinction could be made between the objects and events within an information system (regarding data and the manipulation of data) and the real world objects and events external to such a system.

1.3.2 Goal

Hence, the third goal of the thesis is to enable the reuse of existing ontologies in such a way that high quality can be rather easily achieved by just re-using a proven *foundational ontology*. Apart from that, a high-quality foundational ontology should be general enough to allow for mediation between many interesting (meta)data sources. Finally, a foundational ontology that attracts wide-spread agreement from its community may also lead to a wide-spread understanding of its definitions — and correspondingly to more concise semantic (meta)data.

Therefore, a foundational ontology is used to build an ontology library for both the ontologies used in Application Servers and in Web Services middleware, carefully investigating ontological choices known from philosophy, linguistics and mathematics. Foundational ontologies (aka "upper level ontologies") respond to the aforementioned shortcomings as they provide a reference point for easy and rigorous comparisons among different ontological approaches.

The usage of a common foundational ontology also allows to harmonize both ontologies for a simpler translation acknowledging the fact that services are often exposures of components residing in an Application Server. [15]

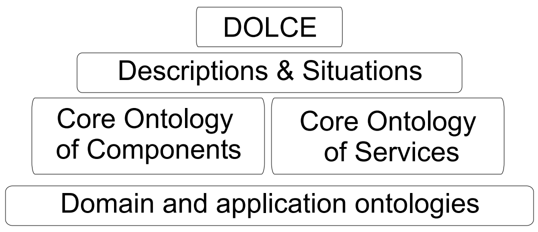
DOLCE
Descriptions & Situations
Core Ontology of Components
Core Ontology of Services
Domain and application ontologies

Figure 2: Ontology Library.

Figure 2 depicts the ontology library. DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) [19] has been chosen as the basis of the library. It features a rich axiomatization of domain independent concepts, explicit construction principles, careful reference to interdisciplinary literature as well as common sense-orientedness. DOLCE is axiomatized in a modal logic, but it is maintained also in other languages, used according to the particular trade-off between expressivity and computational complexity that is required by a certain application.

Several additional theories exist for DOLCE that come in the form of ontology modules. Descriptions & Situations (D & S) is such a module and axiomatizes a theory of ontological contexts [14]. The descriptions of services show a clear contextual nature, one may only have to consider the number of different views that may exist on a service: the view of a service provider, that of the service requestor or the legal view of a contract etc. The concepts used to formulate any given view are clearly separate from the actual objects they act upon and often independent from the concepts appearing in other views.

We applied both DOLCE and the additional modelling capabilities of D & S to formalize core ontologies for components and for services. While the first axiomatizes typical concepts in an Application Server (most prominently software components and their interrelationships) the second deals with similar aspects of services. Domain and application ontologies can finally reuse the core ontologies.

1.3.3 *Significance*

Interest in foundational ontologies is rising as existing ontology-based efforts suffer several problems that create ambiguity and complicate the provision of semantic metadata.

2. METHODOLOGY

Studies were carried out in the WonderWeb project[2] whose objective was, among others, to link new and existing Semantic Web[3] tools in a comprehensive technical infrastructure.

Based on those needs, we derived a set of requirements and possible design solutions. We iterated several times over requirements and design solutions and took into account existing efforts and methods (e.g. MDA or OWL-S [25]).

The resulting infrastructure is an *Application Server for the Semantic Web.* Besides integrating semantic technology within the server itself, it additionally facilitates plug'n'play engineering of ontology-based modules and, thus, the development and maintenance of comprehensive Semantic Web applications. Its design and development are based on existing Application Servers. However, their underlying concepts are applied and augmented for use in the Semantic Web. [24]

The infrastructure will be extended by means for a) the semantic management of Web Services and b) semi-automatic generation of corresponding semantic metadata.

3. EVALUATION

It is usually difficult to substantiate the advantages of ontology-based applications in numbers. The best way to demonstrate their benefits is to have a modularized application and perform a controlled experiment. Modules providing the same functionality with and without the usage of ontologies have to be applied and the application evaluated each time. Such experiments are difficult to set up and in many cases the nature of the application makes it impossible. This is the case with Application Servers and Web Services based systems where ontology-usage is deeply rooted in the infrastructure.

Hence, the thesis takes the following approaches for evaluation. First, a significant part of the architecture has been prototypically implemented and test-driven in a system called KAON SERVER[4]. This prototypical implementation successfully shows that some of the goals of the thesis are already met. Second, another strategy is to show that ontology-based applications make it possible to do something that was hitherto impossible or too costly. Therefore, the thesis will estimate possible cost reductions by usage of ontologies in a company. The idea is to achieve the cost reductions by easier maintenance and thus fewer man power. Both approaches are further discussed in the following subsections.

3.1 KAON SERVER

KAON SERVER [23] implements a significant part of the aforementioned architecture and is optional part of the KArlsruhe ONtology and Semantic Web Toolsuite (KAON) [18]. It makes use of the Java Management Extensions (JMX [17]) — an open technology for component management. With JMX it becomes possible to configure, manage and monitor Java applications at run time, as well as break applications into components that can be exchanged. Basically, JMX defines interfaces of managed beans (*MBeans*) which are JavaBeans that represent JMX manageable resources. MBeans are hosted by an *MBeanServer* which allows their run time deployment and manipulation. All management operations performed on the MBeans are done through interfaces on the MBeanServer.

JMX only provides an API specification with several available implementations. We have chosen *JBossMX* which is the core of the open-source JBoss Application Server [13] that augments J2EE by dynamic component deployment. This choice allows us to inherit all the functionality provided by JBoss in the form of its MBeans (Servlet Containers, EJB Containers etc.). We deploy our inference engine as an additional MBean and augment the existing component loader and dependency management to exploit the inferencing. A version and security management tool allows to browse and query the ontology at run time. Several separate description languages and corresponding files are substituted by the ontology and corresponding semantic metadata. Thus, it is possible to use the KAON SERVER as a "semantically enhanced JBoss".

3.2 Reduction of Maintenance Costs

The usage of semantic technology will be explored at a particular company in order to estimate possible reduction of maintenance costs.

The goal is to show that the company's software developers can better maintain their multitude of databases and servers by inferencing over access rights, system dependencies etc. The relationships between software components, services and databases will be formalized by an ontology. Applying an inference engine will allow powerful semantic queries like "which services are affected by a change of software component x?".

[2]`http://wonderweb.semanticweb.org`

[3]The Semantic Web augments the current WWW by giving information a well-defined meaning, better enabling computers and people to work in cooperation. This is done by adding semantic metadata to web resources.

[4]The KAON SERVER can be downloaded at `http://kaon.semanticweb.org`

Acknowledgements. The author would like to thank Steffen Staab, Andreas Eberhart and Rudi Studer from the Institute AIFB for their much valued supervision. Also Mike Uschold from Boeing as well as the colleagues of the WonderWeb projects for their fruitful hints.

4. REFERENCES

[1] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services*. Springer, Sep 2003.

[2] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. K. F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana. Business Process Execution Language for Web Services Version 1. Specification, May 2003. `http://www.ibm.com/developerworks/library/ws-bpel/`.

[3] A. Arkin, S. Askary, S. Fordin, W. Jekeli, K. Kawaguchi, D. Orchard, S. Pogliani, K. Riemer, S. Struble, P. Takacsi-Nagy, I. Trickovic, and S. Zimek. Web Service Choreography Interface (WSCI). W3C Note, aug 2002. `http://www.w3.org/TR/wsci`.

[4] B. Atkinson, G. Della-Libera, S. Hada, M. Hondo, P. Hallam-Baker, J. Klein, B. LaMacchia, P. Leach, J. Manferdelli, H. Maruyama, A. Nadalin, N. N. H. Prafullchandra, J. Shewchuk, and D. Simon. Web Services Security (WS-Security). Specification, Apr 2002. `http://www-106.ibm.com/developerworks/webservices/library/ws-secure/`.

[5] A. Banerji, C. Bartolini, D. Beringer, V. Chopella, K. Govindarajan, A. Karp, H. Kuno, M. Lemon, G. Pogossiants, S. Sharma, and S. Williams. Web Services Conversation Language (WSCL). W3C Note, Mar 2002. `http://www.w3.org/TR/wscl10/`.

[6] BEA Systems, Computer Associates International, IBM Corporation, Layer 7 Technologies, Microsoft Corporation, Netegrity, Oblix, OpenNetwork Technologies, Ping Identity Corporation, Reactivity, RSA Security, VeriSign, and Westbridge Technology. Web Services Trust Language (WS-Trust). Specification, May 2004. `http://www-106.ibm.com/developerworks/library/specification/ws-trust/`.

[7] S. Borgo, A. Gangemi, N. Guarino, C. Masolo, and A. Oltramari. Ontology RoadMap. WonderWeb Deliverable D15, Dec 2002. `http://wonderweb.semanticweb.org`.

[8] F. Cabrera, G. Copeland, B. Cox, T. F. J. Klein, T. Storey, and S. Thatte. Web Services Transaction (WS-Transaction). Specification, 2002.

[9] L. F. Cabrera, G. Copeland, W. Cox, M. Feingold, T. Freund, J. Johnson, C. Kaler, J. Klein, D. Langworthy, A. Nadalin, D. Orchard, I. Robinson, J. Shewchuk, and T. Storey. Web Services Coordination (WS-Coordination). Specification, Sep 2003. `http://www-106.ibm.com/developerworks/library/ws-coor/`.

[10] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web Services Description Language (WSDL). `http://www.w3.org/TR/wsdl`, Mar 2003. W3C Note.

[11] T. Elrad, R. E. Filman, and A. Bader. Aspect-oriented programming: Introduction. *Communications of the ACM*, 44(10):29–32, October 2001.

[12] D. Fensel and C. Bussler. The Web Service Modeling Framework WSMF. *Electronic Commerce: Research and Applications*, 1:113–137, 2002.

[13] M. Fleury and F. Reverbel. The JBoss Extensible Server. In M. Endler and D. C. Schmidt, editors, *Middleware 2003, ACM/IFIP/USENIX International Middleware Conference, Rio de Janeiro, Brazil, June 16-20, 2003, Proceedings*, volume 2672 of *Lecture Notes in Computer Science*, pages 344–373. Springer, 2003.

[14] A. Gangemi and P. Mika. Understanding the semantic web through descriptions and situations. In *DOA/CoopIS/ODBASE 2003 Confederated International Conferences DOA, CoopIS and ODBASE, Proceedings*, LNCS. Springer, 2003.

[15] A. Gangemi, P. Mika, M. Sabou, and D. Oberle. An ontology of services and service descriptions. Technical report, Laboratory for Applied Ontology (ISTC-CNR), Viale Marx, 15, 00137 Roma, 2003.

[16] A. Gokhale, D. C. Schmidt, B. Natarajan, J. Gray, and N. Wang. Model driven middleware. In Q. H. Mahmoud, editor, *Middleware for Communications*, chapter VII, pages 163–188. Wiley, 2004.

[17] J. Lindfors and M. Fleury. *JMX — Managing J2EE with Java Management Extensions*. Sams, 2002. The JBoss Group.

[18] A. Maedche, B. Motik, and L. Stojanovic. Managing multiple and distributed ontologies in the Semantic Web. *VLDB Journal*, 12(4):286–302, 2003.

[19] C. Masolo, S. Borgo, A. Gangemi, N. Guarino, and A. Oltramari. Ontology Library (final). WonderWeb Deliverable D18, Dec 2003. `http://wonderweb.semanticweb.org`.

[20] P. Mika, D. Oberle, A. Gangemi, and M. Sabou. Foundations for Service Ontologies: Aligning OWL-S to DOLCE. In *The 13th International World Wide Web Conference Proceedings*, pages 563–572. ACM, May 2004.

[21] D. Oberle, A. Eberhart, S. Staab, and R. Volz. Developing and managing software components in an ontology-based application server. In *5th International Middleware Conference*, LNCS. Springer, 2004.

[22] D. Oberle, M. Sabou, and D. Richards. An ontology for semantic middleware: extending DAML-S beyond web-services. Technical Report 426, University of Karlsruhe, Institute AIFB, 76128 Karlsruhe, Germany, 2003.

[23] D. Oberle, S. Staab, R. Studer, and R. Volz. KAON SERVER Demonstrator. WonderWeb Deliverable D7, 2003. http://wonderweb.semanticweb.org.

[24] D. Oberle, S. Staab, R. Studer, and R. Volz. Supporting Application Development in the Semantic Web. *ACM Transactions on Internet Technology (TOIT)*, 4(4), Nov 2004.

[25] The DAML Services Coalition. OWL-S 1.0 draft release. `http://www.daml.org/services/owl-s/1.0/`, Dec 2003.

[26] UDDI Coalition. UDDI Technical White Paper. `http://uddi.org`, Sep 2000.

[27] M. Voskob. UDDI Spec TC V4 Requirement - Taxonomy support for semantics. OASIS, 2004. `http://www.oasis-open.org`.