

SEMANTIC MANAGEMENT OF MIDDLEWARE

SEMANTIC WEB AND BEYOND

Computing for Human Experience

Series Editors:

Ramesh Jain
University of California, Irvine
<http://jain.faculty.gatech.edu/>

Amit Sheth
University of Georgia
<http://lsdis.cs.uga.edu/~amit>

As computing becomes ubiquitous and pervasive, computing is increasingly becoming an extension of human, modifying or enhancing human experience. Today's car reacts to human perception of danger with a series of computers participating in how to handle the vehicle for human command and environmental conditions. Proliferating sensors help with observations, decision making as well as sensory modifications. The emergent semantic web will lead to machine understanding of data and help exploit heterogeneous, multi-source digital media. Emerging applications in situation monitoring and entertainment applications are resulting in development of experiential environments.

SEMANTIC WEB AND BEYOND

Computing for Human Experience

addresses the following goals:

- brings together forward looking research and technology that will shape our world more intimately than ever before as computing becomes an extension of human experience;
- covers all aspects of computing that is very closely tied to human perception, understanding and experience;
- brings together computing that deal with semantics, perception and experience;
- serves as the platform for exchange of both practical technologies and far reaching research.

Additional information about this series can be obtained from
<http://www.springer.com>

SEMANTIC MANAGEMENT OF MIDDLEWARE

by

Daniel Oberle

University of Karlsruhe, Germany



Springer

Daniel Oberle
University of Karlsruhe
Institute of Applied Informatics and
Formal Descriptions Methods
D-76128 Karlsruhe
Germany

Library of Congress Control Number: 2005908104

Semantic Management of Middleware
by Daniel Oberle

ISBN-10: 0-387-0-387-27630-0
ISBN-13: 978-0-387-27630-4

e-ISBN-10: 0-387-0-387-27631-9
e-ISBN-13: 978-0-387-27631-1

Printed on acid-free paper.

Dissertation, genehmigt von der Fakultät für Wirtschaftswissenschaften der
Universität Fridericiana zu Karlsruhe, 2005. Referent: Prof. Dr. Rudi Studer,
Korreferenten: Prof. Dr. Bruno Neibecker, Prof. Dr. Steffen Staab

Cover art by Daniel Oberle, showing the architecture of the ontology-based
application server, from another perspective. The green object is a common
symbol for an ontology.

© 2006 Springer Science+Business Media, Inc.

All rights reserved. This work may not be translated or copied in whole or
in part without the written permission of the publisher (Springer
Science+Business Media, Inc., 233 Spring Street, New York, NY 10013,
USA), except for brief excerpts in connection with reviews or scholarly
analysis. Use in connection with any form of information storage and
retrieval, electronic adaptation, computer software, or by similar or
dissimilar methodology now known or hereafter developed is forbidden.

The use in this publication of trade names, trademarks, service marks and
similar terms, even if they are not identified as such, is not to be taken as
an expression of opinion as to whether or not they are subject to
proprietary rights.

Printed in the United States of America.

9 8 7 6 5 4 3 2 1

springeronline.com

Dedicated to ...

... the Supervisors

*Steffen Staab
Andreas Eberhart
Pascal Hitzler
Rudi Studer*

... the Reviewers

*Ed Curry
Robert Delaney
Aldo Gangemi
Peter Haase
Stefan Tai
Denny Vrandečić
Clare Waibel*

... the Colleagues @ Semantic Karlsruhe

*Andreas Abecker, Sudhir Agarwal, Stephan Bloehdorn, Saartje Brockmans,
Philipp Cimiano, Christian Drumm, Marc Ehrig, Stephan Grimm, Siegfried
Handschuh, Jens Hartmann, Andreas Hotho, Steffen Lamparter, Jens Lemcke,
Alexander Mädche, Boris Motik, Kioumars Namiri, Gisela Schillinger, Lars
Schmidt-Thieme, Christoph Schmitz, Ljiljana & Nenad Stojanovic, Gerd
Stumme, York Sure, Julien Tane, Bernhard Tausch, Christoph Tempich, Max
Völkel, Johanna Völker, Raphael Volz, Susanne Winter, Valentin Zacharias*

... the Colleagues @ World

*Sean Bechhofer, Bettina Berendt, Jorge Gonzalez, Nicola Guarino, Frank van
Harmelen, Ian Horrocks, Hans-Arno Jacobsen, Holger Knublauch, Deborah
McGuinness, Sheila McIlraith, Peter Mika, Christof Momm, Doug Lea, Jeff
Pan, Helena Sofia Pinto, Debbie Richards, Marta Sabou, Jorge Santos, Rick
Schantz, Luc Schneider, Swaminathan Sivasubramanian, Heiner
Stuckenschmidt, Phil Tetlow, Peter Spyns, Mike Uschold, Werner Vogels,
Frank Wolff*

... and last but not least to my family & friends!

Contents

List of Figures	xiii
List of Tables	xv
Preface	xvii
Foreword	xxi
Acknowledgments	xxv
Part I Fundamentals	
1. INTRODUCTION	3
1 Motivation	3
2 Research Questions	5
3 Contributions	8
2. MIDDLEWARE	11
1 Middleware for Distributed Application Development	12
2 Middleware for Enterprise Application Integration	16
3 Middleware for B2B Application Integration	17
3.1 Application Servers	18
3.2 Web Services	25
4 Summary	31
3. ONTOLOGIES	33
1 Definition	34
1.1 What is a Conceptualization?	35
1.2 What is an Ontology?	37
1.3 A Suitable Representation Formalism	41
2 Classification	43

2.1	Classification according to Purpose	45
2.2	Classification according to Expressiveness	45
2.3	Classification according to Specificity	46
3	The Role of Foundational Ontologies	47
4	Ontological Choices	50
4.1	Descriptive vs. Revisionary	50
4.2	Multiplicative vs. Reductionist	51
4.3	Possibilism vs. Actualism	51
4.4	Endurantism vs. Perdurantism	52
4.5	Extrinsic Properties	52
5	Summary	53
4.	TOWARDS SEMANTIC MANAGEMENT	55
1	Scenarios	57
1.1	An Application Server for the Semantic Web	57
1.2	Web Services in SmartWeb	62
2	Use Cases	65
2.1	Application Servers	66
2.2	Web Services	70
3	Summary	75
Part II Design of a Management Ontology		
5.	ANALYSIS OF EXISTING ONTOLOGIES	79
1	OWL-S	81
2	Initial Ontology of Software Components	83
3	Problematic Aspects	86
3.1	Conceptual Ambiguity	87
3.2	Poor Axiomatization	88
3.3	Loose Design	89
3.4	Narrow Scope	91
4	Summary	93
6.	THE APPROPRIATE FOUNDATIONAL ONTOLOGY	95
1	Requirements for Ontological Choices	96
2	Alternatives	97
2.1	BFO	98
2.2	DOLCE	99
2.3	OCHRE	101

2.4	OpenCyc	103
2.5	SUMO	104
3	Summary	105
7.	AN ONTOLOGICAL FORMALIZATION OF SOFTWARE COMPONENTS AND WEB SERVICES	107
1	Modelling Basis	109
1.1	DOLCE	109
1.2	Descriptions & Situations	110
1.3	Ontology of Plans	112
1.4	Ontology of Information Objects	113
2	Core Software Ontology	114
2.1	Software vs. Data	115
2.2	API Description	119
2.3	Semantic API Description	121
2.4	Workflow Information	122
2.5	Access Rights and Policies	124
3	Core Ontology of Software Components	127
3.1	Formalization of the Term “Software Component”	128
3.2	Libraries and Licenses	129
3.3	Component Profiles and Taxonomies	131
3.4	Example	133
4	Core Ontology of Web Services	136
4.1	Formalization of the term “Web service”	136
4.2	Service Profiles and Taxonomies	137
4.3	Example	138
5	Proof of Concept	139
5.1	Meeting the Modelling Requirements	139
5.2	Higher Quality	141
5.3	Enabling Reuse	145
6	Summary	145
Part III Realization of Semantic Management		
8.	DESIGN OF AN ONTOLOGY-BASED APPLICATION SERVER	149
1	General Design Issues	150
1.1	Possible Platforms	150
1.2	Obtaining Semantic Descriptions	152
1.3	How to Integrate the Inference Engine?	154

2	Semantic Management of Software Components	156
2.1	Requirements	157
2.2	The Microkernel Design Pattern	161
2.3	Integration of an Inference Engine	162
2.4	Architecture	163
3	Semantic Management of Web Services	167
4	Summary	169
9.	IMPLEMENTATION	171
1	The JBoss Application Server	172
2	The KAON Tool Suite	174
3	KAON SERVER	177
3.1	Server Core	178
3.2	Connectors	181
3.3	Interceptors	182
3.4	Functional Components	182
3.5	Management Console	183
4	Example	183
4.1	Modelling the Ontology	184
4.2	Definition of Rules	187
4.3	Setting up the Portal	188
5	Summary	189
10.	APPLYING THE MANAGEMENT ONTOLOGY	191
1	From Core to Domain	192
1.1	MBeans	193
1.2	Profiles	194
2	From Reference to Application	196
3	From Heavyweight to Lightweight	199
3.1	The KAON Language	200
3.2	Adaptation of Definitions and Axioms	202
4	Assessment	211
4.1	Application Server Use Cases	212
4.2	Web Services Use Cases	215
5	Summary	217

Part IV Finale

11. RELATED WORK	221
1 Enterprise Application Management	222
1.1 Application Management Systems	222
1.2 Application Management Schemas	224
2 Model-Driven Architectures	226
3 Web Services	227
4 Semantic Web Services	228
4.1 OWL-S	229
4.2 METEOR-S	230
4.3 WSMO	233
4.4 IRS	233
4.5 KDSWS	234
4.6 Other Approaches	234
5 Miscellaneous	235
5.1 Software Reuse Systems	235
5.2 DL IDL	236
5.3 Microsoft SDM	236
5.4 Integration of Software Specifications	236
5.5 Other Ontologies	237
12. CONCLUSION & OUTLOOK	239
1 Summary	239
2 Contributions	241
3 Open Issues	242
Appendices	245
A Taxonomies	245
References	255
Index	267

List of Figures

2.1	Types of middleware and historical overview.	13
2.2	J2EE API's divided into layers.	20
2.3	Example of indirect permission.	22
2.4	Internal vs. external middleware.	26
3.1	Example: Software components and their dependencies.	36
3.2	Intended models vs. models of the ontology.	40
3.3	Example for the equivalence relation A .	43
3.4	Classification of ontologies.	44
3.5	Foundational ontologies and intended models.	48
3.6	Classification of foundational ontologies.	49
4.1	Working hypothesis.	56
4.2	Static and dynamic aspects of the Semantic Web.	58
4.3	Semantic Web example.	59
4.4	Information flow in the research and academia example.	61
4.5	Simplified SmartWeb Architecture.	64
5.1	The goal of Part II is to design a management ontology.	80
5.2	The OWL-S Service ontology module as UML class diagram.	82
5.3	OWL-S and the initial ontology of software components.	84
5.4	The representation of attribute binding in OWL-S.	90
6.1	BFO Taxonomy.	99
6.2	DOLCE Taxonomy.	100
6.3	OCHRE Taxonomy.	102
6.4	OpenCyc Taxonomy.	104
6.5	SUMO Taxonomy.	105
7.1	Overview of the management ontology.	108

7.2	Sketch of DOLCE.	110
7.3	The Descriptions & Situations ontology module.	111
7.4	The Ontology of Plans.	112
7.5	The Ontology of Information Objects.	114
7.6	The classification of software and data.	118
7.7	Semantic API description.	121
7.8	The Policy Description.	127
7.9	UML diagram of the software component example.	135
7.10	UML diagram of the Web services example.	140
7.11	Solution to the attribute binding problem.	143
7.12	Wider scope through the Ontology of Information Objects.	144
8.1	The Reverse Engineering Approach.	155
8.2	The Model-Driven Deployment Approach.	155
8.3	The Ontology Run Time Approach.	156
8.4	Architecture of the ontology-based application server.	164
9.1	JMX Architecture.	173
9.2	Basic architecture of JBoss.	173
9.3	KAON tool suite overview.	175
9.4	KAON OI-Modeller screenshot.	176
9.5	Mapping from design to implementation elements.	178
9.6	KAON OI-Modeller as management console.	183
9.7	An instance of KAON SERVER.	185
9.8	Sequence diagram — Oiled with KAON SERVER.	186
9.9	Sequence diagram — OntoEdit with KAON SERVER.	188
10.1	Reuse of the management ontology in the KAON SERVER.	192
11.1	CIM for J2EE Application Servers.	225
A.1	DOLCE.	246
A.2	Descriptions & Situations.	247
A.3	Ontology of Plans.	248
A.4	Ontology of Information Objects.	249
A.5	Core Software Ontology.	250
A.6	Core Ontology of Software Components.	251
A.7	Core Ontology of Web Services.	252
A.8	KAON SERVER Ontology.	253

List of Tables

5.1	Dependencies between requirements and ontology modules.	86
6.1	Foundational ontologies and their ontological choices.	106
6.2	Foundational ontologies and their extrinsic properties.	106
7.1	Meeting the modelling requirements for software components.	141
7.2	Meeting the modelling requirements for Web services.	141
8.1	Dependencies between requirements and design elements.	166
8.2	Dependencies between use cases and design elements.	169
9.1	WSDL mapping.	180
9.2	WS-BPEL mapping.	181
9.3	WS-Policy mapping.	181
10.1	Definitions kept or removed from the management ontology.	197
10.2	Axioms kept or removed from the management ontology.	198
10.3	Effort comparison for the <i>Library Dependencies and Versioning</i> and <i>Licensing</i> use cases.	212
10.4	Effort comparison for the <i>Capability Descriptions</i> use case.	213
10.5	Effort comparison for the <i>Component Classification and Discovery</i> and <i>Semantics of Parameters</i> use cases.	213
10.6	Effort comparison for the <i>Automatic Generation of Web Service Descriptions</i> use cases.	214
10.7	Effort comparison for the <i>Access Rights</i> use case.	214
10.8	Effort comparison for the <i>Exception Handling</i> use case.	214
10.9	Effort comparison for the <i>Transactional Settings</i> and <i>Secure Communication</i> use cases.	215
10.10	Effort comparison for the <i>Policy Handling</i> and <i>Relating Communication Parameters</i> use cases.	216

10.11	Effort comparison for the <i>Detecting Loops in Interorganizational Workflows</i> use case.	216
10.12	Effort comparison for the <i>Monitoring of Changes</i> use case.	217
10.13	Effort comparison for the <i>Aggregating Service Information</i> use case.	217
10.14	Effort comparison for the <i>Quality of Service</i> use case.	217

Preface

We have termed this series “Semantic Web and beyond: computing for human experience.” Ramesh Jain (co-editor of this series) and I believe that semantics is going to be far more pervasive than portrayed by the current vision of the Semantic Web. Its role and values will certainly not be limited to the traditional Web. Semantics will also be one of the important components of a continuum leading to perception and experience, albeit one that will mature earlier in computational context. We also believe that computation, supported by techniques and technologies that deal with perception, semantics, and experiences, will improve and benefit human experience. Such a computation will have a far broader impact than the traditional drivers of information technology, such as improving efficiency, lowering cost, or productivity gains. In this context, we expect that this series intends to offer additional books covering topics in perception, semantics, and experiential computing as they relate to improving human experience involving interactions with computing devices and environments. Our series intends to offer research monographs, books for professional audiences, as well as text books for advance graduate courses.

This premier book in our series by Daniel Oberle is a good example of what we hope to cover in this series. It discusses the role of semantics in middleware — arguable the most important segment of the enterprise software market. This work demonstrates that semantics and the semantic (and Semantic Web) technologies have pervasive applications and uses. It is also an excellent training companion for active practitioners seeking to incorporate advanced and leading edge ontology-based approaches and technologies. It is a necessary preparation manual for researchers in distributed computing who see semantics as an important enabler for the next generation.

Middleware systems are complex. They need to integrate and manage multiple heterogeneous software systems. Just as semantics has been recognized as a key enabler of heterogeneous information integration, can semantics be a key enabler in integrating heterogeneous software systems? Daniel believes

that is indeed the case, and he goes on to provide a detailed road map on how semantics and Semantic Web technologies can play a significant role in creating a middleware system and their use in managing heterogeneous software systems.

The first step in the road map is modelling which centers around using ontologies for specifying semantic models. This leads to the development of a semantic model for software components and Web services. We are introduced to the basics of middleware technology where technologies and design patterns from the past, such as message-oriented middleware or object monitors, help readers who are not familiar with the area to get the necessary background information. The discussion on ontologies achieves the same.

The subsequent part of this book offers a detailed discussion on the different ontology frameworks which can be used as a modelling basis. The requirements for the ontology are well laid out and each ontology framework is analyzed with respect to the requirements. DOLCE is chosen since it meets most of the requirements. The discussion on the semantic modelling of software systems is particularly interesting to read. The author addresses the different modelling issues at different phases of software component design. The modelling captures the intricate details and differences between fundamental concepts, such as data and software, and continues with component profiles, policies, and many more aspects. The modelling also captures the API of components and proposes a technique to discover dependent and conflicting libraries. Also presented is a model to capture workflows. A discussion on how such a modelling can meet the requirements and the advantages of using such an approach are presented in detail.

The next part provides a technical look at the different solutions. This includes discussing each aspect of the middleware system and the techniques to realize them as actual systems. The requirements of such a semantic middleware are presented and the system design is discussed with that central perspective. The system architecture along its different constituting components are discussed in detail. The application and reuse of the proposed ontology in the middleware system is also presented. The book ends with relating this approach to application management, Semantic Web Services and MDA.

Potentially, the most lasting engineering progress in this book, in my personal view, is that of taking semantics to the application server level. I foresee an emergence of Semantic Aware Networking, in which semantics not only facilitates network functions but significantly enhances its capability by pushing more functions. With the industry already taking initial steps in this direction, such as in CISCO's Application Oriented Networking products, the next step is quite likely the interplay between routers and application servers with semantics providing a bridge.

While there is plenty of work related to semantics of information and even Web services, this effort stands out in its attention to modelling the semantics of software components. In this context, it is a unique offering that goes beyond the mainstream Semantic Web research, while demonstrating a detailed and pervasive use of semantics in larger software systems context. This series will endeavor to offer more such books and for wider audiences.

Amit P. Sheth
Director, Large Scale Distributed Information Systems lab
Professor, Computer Science Department, University of Georgia
CTO and Cofounder, Semagix, Inc.
Athens, Georgia,
U.S.A.

Foreword

Which topic in computer science has been attracting researchers *and* developers from *artificial intelligence, business process modelling, conceptual modelling, databases, distributed systems, information systems, programming systems, security, software engineering, Web services and Web systems and engineering* (and probably many others whom I forgot to mention here)? It is the specification, development and management of component- and service-oriented architectures (SOAs).

The topic has become important to all of them. While the development of individual software systems is reasonably well understood and reasonably well-established practice (even when thousands of issues of such systems are and will have to be dealt with in more detail), the specification, development and maintenance of distributed software systems has in general not been understood to an extent that let their stakeholders gain the intended economic network benefits. On the upside, organizations that establish networking of their numerous distributed systems with the ones of other organizations may save costs, gain new customers or increase customer satisfaction. On the downside, if each minor change or minor disruption in one software system leads to a trickle down effect that results in expensive reprogramming of another system, all the potential positive network effects are overshadowed by the costs for joining and remaining in the network.

As the spectrum of interests indicates, the solution to this dilemma may require a multi-faceted approach. This book by Daniel Oberle significantly contributes towards this objective. Hence, the methods he proposes, revises and extends contribute to the plentiful, seminal research of several communities. I will name the ones that are most immediately affected, though I strongly believe that all of the above cited interest groups may benefit from building on his contribution:

Distributed Systems: Current middleware systems, such as application servers, are complex beasts that are very hard to tame because of the intricacies of distributed interactions. Hence, it has been a long-established practice to factorize configuration aspects of distributed interactions into corresponding declarative description files and — more recently — into XML files that follow the specification given by the various Web service standards.

Unfortunately, the semantics of these files is either given by the code of the concrete middleware system or — probably worse — by thousands of pages of specification documents consisting of raw textual explanations. We all know what went wrong with compilers in the 1960ies when programming language specifications were still at that stage.

Daniel here builds a rigorous approach towards giving the declarative descriptions of software components/Web services a well-defined meaning by defining ontological foundations *and* by showing how such foundations may be realized in practical, up-and-running systems.

Artificial Intelligence — Ontologies: Though all software developers use programming languages, only few specialists are actually able to formally define a programming language and develop a corresponding compiler: the formal foundation is not used to tutor the beginner, but to clarify the discussion and development by experts. The same is true for ontologies that underly a software management approach. They need to outlive many software development cycles, i.e., they need to have a formal foundation, yet one must also tutor the domain experts how to use them.

It is one of the successes of this work that it shows how to develop *and* use the ontological foundations of this work in a concrete software environment. This is done in a way that the usage of the resulting middleware infrastructure seems amenable to a sophisticated software developer even though the development of a complex foundational ontology may have to be left to some few specialists.

Web Services — Semantic Web Services: The analysis of the ontologies Daniel develops makes evident that *very few concepts* actually differ when “upgrading” from conventional middleware to Web services. It also makes clear that the use of declarative specifications, such as done in Web services, or formal declarative specifications, such as done for Semantic Web Services comes with economic modelling costs that need to be justified by savings in other places. This lets us presume that formal specifications with the objective of fully automatic Web service composition and orchestration remain a valid research topic — but one that will find its applications in niches rather than in wide-spread adoption by software developers.

Thus, the book covers an incredible depth and breadth of approaches. Its value lies in revising and extending existing methods thereby providing the cornerstones for specifying, developing and managing distributed applications in the coming decades — using semantics.

Prof. Dr. Steffen Staab
ISWeb — Information Systems and Semantic Web
Institute for Computer Science
University of Koblenz-Landau
Germany

Acknowledgments

This work was financed by WonderWeb — “Ontology Infrastructure for the Semantic Web,” a European Union Information Society Technologies (IST) Future Emerging Technologies (FET) funded project under contract number IST-2001-33052 (2002-2004). I feel indebted to all the colleagues for the much valued cooperation and fruitful discussions we had throughout the project.

<http://wonderweb.semanticweb.org>

This work was financed by SmartWeb, a research project funded by the German Federal Ministry of Education and Research (BMBF). My appreciation to all the colleagues of another great project still running at the time of writing this book (2004-2007). It has been a joy working with you all!

<http://smartweb.semanticweb.org>

My appreciation to all the members of the Software Engineering Task Force (SETF). The task force is a part of the World Wide Web’s Consortium (W3C) Semantic Web Best Practices and Deployment Working Group (SWBPD), where this work was also successfully exposed and revised [Tetlow et al., 2005].

<http://www.w3.org/2001/sw/BestPractices/SE/>

Finally, I would like to express my gratitude to Amit Sheth — the series editor — as well as the ladies at Springer: Susan Lagerstrom-Fife and Sharon Palleschi, who made the publication of this book possible.