Christoph Bussler
Val Tannen
Irini Fundulaki (Eds.)

LNCS 3372

Semantic Web and Databases

Second International Workshop, SWDB 2004
Toronto, Canada, August 2004
Revised Selected Papers

Springer

Christoph Bussler Val Tannen
Irini Fundulaki (Eds.)

Semantic Web and Databases

Second International Workshop, SWDB 2004
Toronto, Canada, August 29-30, 2004
Revised Selected Papers

Springer

Volume Editors

Christoph Bussler
National University of Ireland
Digital Enterprise Research Institute (DERI)
Galway, Ireland
E-mail: Chris.Bussler@DERI.org

Val Tannen
University of Pennsylvania
Department of Computer and Information Science
200 South 33rd Street, Philadelphia, Pennsylvania 19104-6389, USA
E-mail: val@cis.upenn.edu

Irini Fundulaki
Bell Labs Research, Lucent Technologies
600 Mountain Avenue, Murray Hill, NJ 07974, USA
E-mail: fundulaki@research.bell-labs.com

Library of Congress Control Number: 2005920538

CR Subject Classification (1998): H.2, H.3, H.4, H.5, I.2, C.2.4

ISSN 0302-9743
ISBN 3-540-24576-6 Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media

springeronline.com

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11390060 06/3142 5 4 3 2 1 0

SWDB 2004 Co-chairs' Message

We would like to welcome you to the Proceedings of the 2nd International Workshop on Semantic Web and Databases (SWDB 2004) that was held in conjunction with the 30th International Conference on Very Large Data Bases in Toronto, Canada.

The Semantic Web is a key initiative being promoted by the World Wide Web Consortium (W3C) as the next generation of the current Web. The objective of this workshop series is to gain insight into the evolution of Semantic Web technologies and their applications to databases and information management. Early commercial applications that make use of machine-understandable metadata range from information retrieval to Web-enabling of old-tech IBM 3270 sessions. Current developments include metadata-based Enterprise Application Integration (EAI) systems, data modelling solutions, and wireless applications. All these different areas utilize databases and therefore the combination of Semantic Web and database technologies is essential.

In total, we received 47 submissions, out of which the program committee selected 14 as full papers for presentation and publication.

SWDB 2004 shared its two very interesting and stimulating keynotes with another one of the VLDB 2004 satellite events, the 5th Workshop on Technologies for E-Services (TES 2004). The first keynote was given by Boualem Benatallah with the title "Service-Oriented Computing: Opportunities and Challenges." The second keynote was given jointly by Alex Borgida and John Mylopoulos with the title "Data Semantics Revisited." The keynote speakers agreed to contribute to these proceedings by providing articles detailing their keynote talks.

We would like to thank all authors who submitted and presented papers at the workshop for their hard work and the keynote speakers for their excellent contributions. We would like to thank the Program Committee members for providing (almost) all reviews in time, and for the quality of their reviews, as it directly reflects the quality of the workshop and of these proceedings. Michal Zaremba did a great job setting up and maintaining the paper management system, we owe him many thanks for making the whole process very smooth. Finally, we would also like to thank all workshop attendees for their active participation, which added the final ingredient to what we believe was a very successful event.

October 2004

Christoph Bussler
Val Tannen

Workshop Organization

Program Chairs
Chris Bussler (Digital Enterprise Research Institute, Ireland)
Val Tannen (University of Pennsylvania, USA)

Steering Committee
Isabel Cruz (University of Illinois at Chicago, USA)
Vipul Kashyap (National Library of Medicine, USA)
Stefan Decker (Digital Enterprise Research Institute, Ireland)

Program Committee
Karl Aberer (EPFL, Switzerland)
Sibel Adali (Rensselaer Polytechnic Institute, USA)
Gustavo Alonso (ETH Zurich, Switzerland)
Bernd Amann (CNAM and INRIA, France)
Boualem Benatallah (University of New South Wales, Australia)
Sonia Bergamaschi (Università di Modena e Reggio Emilia, Italy)
Alex Borgida (Rutgers University, USA)
Stephane Bressan (National University of Singapore, Singapore)
Fabio Casati (HP Laboratories, USA)
Vassilis Christophides (ICS-FORTH and University of Crete, Greece)
Isabel Cruz (University of Illinois at Chicago, USA)
Umeshwar Dayal (HP Laboratories, USA)
Tim Finin (University of Maryland, Baltimore County, USA)
Irini Fundulaki (Bell Labs, USA)
Zack Ives (University of Pennsylvania, USA)
Vipul Kashyap (National Library of Medicine, USA)
Christoph Koch (Technische Universitaet Wien, Austria)
Harumi Kuno (HP Laboratories, USA)
Maurizio Lenzerini (Università di Roma "La Sapienza," Italy)
Ling Liu (Georgia Tech, USA)
Dennis McLeod (University of Southern California, USA)
Alex Poulovassilis (Birkbeck College, UK)
William Regli (Drexel University, USA)
Jérome Siméôn (IBM T.J. Watson Research Center, USA)
Rudi Studer (Universitaet Karlsruhe, Germany)
Kevin Wilkinson (HP Laboratories, USA)

Publicity and Publications Chair
Irini Fundulaki (Bell Labs, USA)

RoboCup Rescue Robot League
Adam Jacoff, National Institute of Standards and Technology, Intelligent Systems Divition, USA
RoboCup Rescue Simulation League
Tomoichi Takahashi, Meijo University, Japan
RoboCup Junior
Luis Almeida, University of Aveiro, Portugal
Jeffrey Johnson, Open University, UK
RoboCup Websites
Ansgar Bredenfeld, AIS, Germany

Table of Contents

Service Oriented Computing: Opportunities and Challenges

Boualem Benatallah and H.R. Motahari Nezhad

School of Computer Science and Engineering,
The University of New South Wales,
Sydney, NSW 2052, Australia
{boualem, hamidm}@cse.unsw.edu.au

Abstract. Service oriented architectures (SOAs) are emerging as the technologies and architectures of choice for implementing distributed systems. Recent advances and standardization efforts in SOAs provide necessary building blocks for supporting the automated development and interoperability of services. Although, standardization is crucial by no means is sufficient. Wide spread adoption of service technologies requires high level framework and methodology and identification of appropriate abstractions and notations for specifying service requirements and characteristics to support automated development and interoperability. In this paper, we identify interoperability layers of SOAs, review major approaches for service development and highlight some research directions.

1 Introduction

Service-oriented architectures (SOAs) are emerging as the technologies and architectures of choice for implementing distributed systems and performing application integration within and across companies' boundaries [6][7][8]. The vision of SOAs is to allow autonomous partners to advertise their terms and capabilities, and engage in peer-to-peer interactions with any other partners and enable on demand computing through composition and outsourcing. The foundation of SOAs lies in the modularization and visualization of system functions and exposing them as services that: (i) can be described, advertised, and discovered using (XML-based) standard languages and (ii) interoperate through standard Internet protocols. SOAs are characterized by two trends that were not part of conventional (e.g., CORBA-like) middleware. The first is that, from a technology perspective, all interacting entities are considered to be (Web) services, even when they are in fact requesting and not providing services. This allows uniformity in the specification language and interaction protocols (e.g., the interface of both requestor and providers will be described using the Web Services Description Language -WSDL).

The second trend, that is gathering momentum, is that of including, as part of the service description, not only the service interface but also *the business protocol* supported by the service, i.e., the specification of which message exchange

C. Bussler et al. (Eds.): SWDB 2004, LNCS 3372, pp. 1–8, 2005.

sequences are supported by the service. The interactions between clients and services are always structured in terms of a set of operation invocations, whose order typically has to obey certain constraints for clients to be able to obtain the service they need. In addition to the business protocol, a service may be characterized by other abstractions such as security (e.g., trust negotiation) or transaction policies that also need to be exposed as part of the service description so that clients know how to interact with a service.

While standardization is crucial in making SOA a reality, the effective use and widespread adoption of service technologies and standards requires: (i) high-level frameworks and methodologies for supporting automated development and interoperability (e.g., code generation, protocol compatibility and conformance), and (ii) identification of appropriate abstractions and notations for specifying service requirements and characteristics. These abstractions form the basis of service development frameworks and methodologies.

In this paper, we identify interoperability layers of SOAs and review major approaches for developing service-oriented applications. We also briefly outline some directions.

2 Service Oriented Architectures: Overview and Interoperability Layers

When services are described and interact in a standardized manner, the task of developing complex services by composing other (basic or composite) services is considerably simplified. Indeed, as SOA-related technologies mature, service composition is expected to play a bigger and bigger role in service development. Since Web services will be sought during assembly of composite services, their functionality need to be described such that clients can discover them and evaluate their appropriateness and compositions. The above observations emphasize both opportunities and needs in service development. In fact, they raise the issue of how to support the protocol specification lifecycle, and of how to guide the implementation (especially in the case of composite services) by starting from protocol specifications. Business protocols and compositions are not the only aspects presented in this paper. In addition, one of the major concerns of SOA that is interoperability at various abstraction layers is discussed.

Let us consider a motivating example of B2B integration (B2Bi) where Company A wants to purchase a product from company B. Companies A and B after discovering their match for business (e.g., using a public or private registry), need to agree on the joint business process, i.e., activities, message exchange sequence and interaction contracts, e.g., security, privacy and QoS policies. Companies A and B also need to know and understand the content of exchanged messages. For example, company A needs to know how to send a purchase order to B in terms of product description, order and message structure. Finally, there might be a way to communicate the messages that contain requests and business documents between A and B. In the remainder of this section, we discuss interoperability issues at the following layers: *messaging, content, business protocol* and *policy.*

2.1 The Messaging Layer

This layer provides protocols and adapters for interoperable message exchange among business partners over the network in a reliable and secure manner. A communication protocol consists of a set of rules, which determine message format, transmission and processing for the purpose of exchanging information between two or more services. Software applications usually have a close tie to the syntax of protocol. In addition, it is very often the case that business partners use different platforms, communication protocols or different versions of the same protocol. For example, company A may support SOAP 1.2, while company B supports SOAP 1.1; however, changes in SOAP 1.2 are minor and almost exclusive to additions rather than modifications, e.g., adding HTTP GET method, while SOAP 1.1 only supports HTTP POST method. These are changes in the syntax of the protocol but affect the compatibility of communication protocols of partners so an adapter is required to allow both systems to interoperate successfully.

2.2 The Content Layer

This layer provides protocols, languages and mediators for interoperable and consistent interpretation of the content of exchanged messages by hiding encoding, structure and semantic heterogeneities. Encoding differences arise when two services provide the same functionality using different operation signatures, i.e., different operation names and input/output schemas [11]. Structure heterogeneity happens due to presence of structure differences between the interfaces of two or more partner services, e.g., missing/extra operations or input/output messages in operations of one of the services. Semantic heterogeneity means that services provide overlapping but not the same functionality or when they have different interpretations of the same concept in exchanged business documents. For example, the data item "Price" in an invoice document may mean inclusion or exclusion of tax.

2.3 The Business Protocol Layer

This layer deals with the semantic of interactions between partners. The semantic of interactions must be well defined such that there is no ambiguity as to what a message may mean, what actions are allowed, what responses are expected and in what order messages should be sent. For example, if the protocol of a client requires explicit acknowledgement when sending a purchase order message, the protocol of the provider should support that. Interoperability at this layer is a challenging task since it requires understanding the semantics of external business protocols of partner services. In traditional EAI middleware, e.g., CORBA-based solutions, components interface describes very little semantics and collaborative business processes are usually agreed upon offline. In SOAs, richer descriptions are needed, since services should be self-describing. Automation requires rich description models but a balance between expression power and simplicity is important for the success of the technology.

2.4 The Policy Layer

This layer is concerned with the matching and compliance checking of service policies (e.g, QoS, privacy policies). Policies play a vital role in B2Bi by making the implicit information, as in closed environments, explicit, which is essential in autonomous environments. Policies compatibility checking is essential to find a composition of policy assertions that allow autonomous services to interoperate.

3 State of the Art

In this section we discuss three major approaches in service-oriented architectures: Web services, ebXML, and Semantic Web Services.

3.1 Web Services

Web services are self-described and autonomous software entities that can be published, discovered, and invoked over the Internet (using XML-based standard languages and protocols). The basic technological infrastructure for Web services is today structured around two major standards: SOAP and WSDL (Web Services Description Language). These standards provide the building blocks for service API description and service interoperation, the two basic elements of any programmatic interaction. Web service technologies are evolving toward being able to support more advanced functionalities including discovery, security, transactions, reliability, and collaborative processes management. Several (sometimes overlapping and competing) proposals have been made in this direction, including for example UDDI (Universal Description, Discovery and Integration), WS-Security, WS-Transaction, WS-ReliableMessaging, BPEL4WS (Business Process Execution Language for Web Services), and WSCI (Web Service Choreography Interface). These standards, once they mature and become accepted, will constitute the basis on top of which developers can develop reliable and secure communications among Web services.

At the messaging layer, Web services use SOAP for document exchange and encapsulation of RPC-like interactions. However, the extensibility points provided in the specification are the source of interoperability issues. In addition, incorporation of security and reliability features are still evolving. At the content layer, WSDL describes Web services as collections of endpoints (port types). Port types described the structure of messages the endpoint support. Port types are not enough to define business protocols. Several efforts that recognize the need for extending existing service description languages to cater for constraints such as the valid sequence of service invocations exist [1]. These include work done in standardization efforts such as WSCL (Web Services Conversation Language) and WSCI. However, these protocol languages offer only limited primitives to describe important abstractions such as temporal constraints (e.g., a maximum interval between the invocation of two operations) or the implications and the effects of service invocations from requester perspective (e.g., whether requesters can cancel an operation and what is the cancellation fee) [3].

At the business protocol layer, while proposals like BPEL4WS and WSCI feature some support for defining the conversations that a Web service supports, they are not entirely adequate for specifying business protocols. The conversation functionality provided by BPEL4WS is essentially driven from its composition nature: in other words, BPEL4WS has been primarily designed as a composition language, in which the same formalism used for composition (a process) can also be used for defining conversations. WS-Transaction and the OASIS Business Transaction Protocol (BTP) also deal with conversations and in particular with transactional conversations. However, their goal is that of providing a framework through which services can be coordinated to enforce transactional protocols, rather than providing conversation abstractions and high-level modeling [1].

At the policy layer, WS-Policy defines a base set of extensible constructs for Web services to describe their policies. WS-PolicyAssertions provides an initial set of general message-related assertions such as preferred text encoding. However, neither a high level framework and abstractions for modeling various polices nor a methodology for analyzing relationships between policies (e.g., matching, refinement) is provided.

To summarize, current efforts in Web services area focus on identifying different aspect of services such as interface descriptions, business protocols and policies and propose specifications to cater for such requirements. However, there is no high-level modeling framework and notation for identifying and describing important abstractions such as transactional implications and trust negotiation. Nor is there any framework for helping developers on where and how to apply such abstractions, e.g., security, privacy policies in Web service environment. In addition, the description of policies is mainly characterized by ad-hoc methods that can be time consuming and error prone. Hence, there is a need for high-level frameworks and tools to guide developers on how to use Web service infrastructures (e,g., standards) and provide support for automating the development, enforcement, and evolution of protocols and polices of services.

3.2 ebXML

ebXML (Electronic Business XML) [4] presents a set of specifications and standards for collaborative B2B integration. It takes a top-down approach by allowing partners to define mutually negotiated agreement at a higher level, i.e., business protocols and contracts, and then working down towards all the details of how to exchange concrete messages.

At the messaging layer, partners exchange messages through the messaging service (ebMS). ebMS extends SOAP for secure and reliable payload exchange using existing security infrastructure (e.g., SSL, digital signatures). However, it does not support advanced security features such as federated access control, identity management and trust negotiation. At the content layer, ebXML uses business documents, which consist of a set of fine-grained information items that are interchanged as a part of business process. It allows the use of domain vocabularies derived from standardized core components. However, the shared documents are agreed upon collaboratively.

At the business protocol layer, ebXML defines collaboration protocol agreements (CPAs) using informal descriptions. At the policy layer, ebXML does not explicitly support expression of policies. However, collaboration protocol profiles (CPPs) can be used for this purpose. A CPP defines capabilities of a party to engage in business and so policies can be listed as the capabilities of a company in its CPPs. In addition to the fact that ebXML does not provide for the fragmentation of different policies, the lacks of high level modeling and reasoning about protocols and policies hinders the specification of relevant properties in a way that is useful for activities such as formal analysis, consistency checking of system functionalities, refinement and code generation, etc.

3.3 Semantic Web Services

Semantic Web aims at improving the technology to organise, search, integrate, and evolve Web-accessible resources by using rich and machine-understandable abstractions for the representation of resources semantics. Ontologies are proposed as means to address semantic heterogeneity among Web-accessible information sources and services. Efforts in this area include the development of ontology languages such as RDF, DAML+OIL, and OWL. In the context of Web services, ontologies promise to take interoperability a step further by providing rich description and modelling of services properties, capabilities, and behaviour. OWL-S (formerly called DAML-S) [5] is an ontology for describing Web services.

OWL-S consists of three interrelated subontologies, known as the profile, process model, and grounding. The profile describes the capabilities and parameters of the service. The process model details both the control structure and dataflow structure of the service required to execute a service. The grounding specifies the details of how to access the service, via messages (e.g., communication protocol, message formats, addressing, etc).

At the messaging layer, semantic Web services rely on the efforts in Web services approach. At the content layer, OWL-S uses the profile. At the business protocol layer, OWL-S uses the process model. Although, it does not cater for important abstraction such as transactional implications, temporal constraints.

At the policy layer, OWL-S does not explicitly formalize and specify policies. However, the profile of OWL-S can be used to express policies such as security and privacy as a part of unbounded list of service parameters of the profile. But, there is no consideration for fragmentation of different policie and identification and representation of important service abstractions such as transactional implications and trust negotiation. Although, it should be noted that ontologies provide the basis for defining vocabularies to represent policies (e.g, [10] uses an ontology-based approach for representing security policies).

4 Directions

Recent advances in Web service technologies provide necessary building blocks for supporting the development of integrated applications within and across organizations. A number of XML-based standard languages and protocols exist today (e.g., SOAP, WSDL, BPEL). Service development tools (e.g., BPEL4WJ, Collaxa) that support emerging standards and protocols also started to appear. However, the effective use and widespread adoption of Web service technologies and standards requires: (i) high-level frameworks and methodologies for supporting automated development and interoperability (e.g., code generation, compatibility), and (ii) identification of appropriate abstractions and notations for specifying service requirements and characteristics. These abstractions form the basis of service development frameworks and methodologies [2].

We argue that abstracting Web services protocols will benefit several automation activities in Web services lifecycle. We believe that once the research and development work on the aspects identified above has been completed, this approach will result in a comprehensive methodology and platform that can facilitate large-scale interoperation of Web services and substantially reduce service development effort. This will foster the widespread adoption of Web service technology and of the service-oriented computing paradigm by providing pillars abstractions and mechanisms to effectively discover, integrate, and manage services in large, autonomous, and possibly dynamic environment. It should be noted that model driven development of applications is a well-established practice [9]. However, in terms of managing the Web service development lifecycle and model-driven Web service development, technology is still in the early stages. In particular, with regard to model driven approaches to Web service protocols prior work are either [1]:

- too low-level and consequently not suitable for automating activities such as compatibility checking, code generation, and protocol specification refinement and conformance, or
- do not explicitly take important service abstractions into account, and are consequently ineffective for automating services discovery, interoperation, development, and evolution.

It is worth mentioning that several ongoing efforts in the area of Web services recognize the need for the high-level specification of conversation protocols. These efforts focus on conversation protocols compatibility and composition. Similar approaches for protocols compatibility exist in the area of component-based systems. These efforts provide models (e.g., pi-calculus -based languages for component interface specifications) and algorithms (e.g., compatibility checking) that can be generalized for use in Web service protocol specifications and management. Also, in the area of business process modeling, several approaches based on formal formalisms such as Petri nets, labeled transition graphs, and state charts exist. However, the conversation protocol specification languages used in these approaches do not consider important abstractions such as temporal constraints (e.g., when an operation should occur), the implications and the effects of service invocations from requester perspective (e.g., whether requesters can cancel an operation and what is the cancellation fee).

To summarize, effective abstracting of service protocols and policies can form the basis of the building blocks of a scalable and agile service oriented infrastructure. For example, richer conversation models enable a more effective static and dynamic binding, as clients can be more selective on the behavior properties of the services they bind to. Clients for instance may require that the selected service allow the cancellation of a given operation within a certain time interval from its completion. Other automation that will benefit from service protocols abstraction are compatibility of protocols, validation of service composition models, generation of service composition skeletons, and joint analysis of compositions and protocol specifications [1].

Acknowledgement. Authors would like to thank Fabio Casati, Farouk Toumani and Halvard Skogsrud for their valuable contributions to this work.

References

1. Benatallah, B., Casati, F., Skogsrud, H., and Toumani, F.: Abstracting and Enforcing Web Service Protocols. Int'l Journal of Cooperative Information Systems (IJCIS). World Scientific, December (2004) (To appear).
2. Benatallah, B., Casati, F., Toumani, F.: Analysis and Management of Web Service Protocols. Proceedings of the 23rd International Conference on Conceptual Modeling (ER). Springer-Verlag, Shanghai, China, November 2004 (To appear).
3. Benatallah, B., Casati, F., Toumani, F.: Web Service Conversation Modeling: A Cornerstone for e-Business Automation. IEEE Internet Computing, vol. 8, no. 1, Jan/Feb (2004) 46-54.
4. ebXML: www.ebxml.org.
5. OWL-S: Semantic Markup for Web Services. www.daml.org/services/owl-s/.
6. Alonso, G., Casati, F., Kanu, H., Machiraju. V., Web Services: Concepts, Architectures, and Applications. Springer Verlag, (2004).
7. Papazoglou, M.P., and Georgakopoulos, D.: Special Issue on Service-Oriented Computing. Communiocation of ACM, vol. 46, no. 10, (2003) 24-28.
8. Chung, J.Y., Lin, K.J., Mathieu, R.G.: Special Section on Web Services Computing. IEEE Computer, vol. 36, no. 10, (2003) 35-71.
9. Mellor, S., Clark, A. N., and Futagami, T.: Special issue on Model-Driven Development. IEEE Software, vol. 20, no. 5, (2003).
10. Kagal, L., Paolucci, M., Srinivasan, N., Denker, G., Finin, T., Sycara, K.: Authorization and Privacy inSemantic Web Services. IEEE Intelligent Systems, vol. 19, no. 4, July-Aug (2004) 50-56.
11. Ponnekanti, S., Fox, A., Interoperability Among Independently Evolving Web Services, 5th International Middleware Conference, October, 2004.

Data Semantics Revisited

Alexander Borgida and John Mylopoulos

[1] Dept. of Computer Science, Rutgers University, NJ, USA
borgida@cs.rutgers.edu
[2] Dept. of Computer Science, University of Toronto, Toronto, Canada
jm@cs.toronto.edu

" ... It struck me that it would be good to take one thing in life and regard it from many viewpoints, as a focus for my being, and perhaps as a penance for alternatives missed. ... "
R. Zelazny: *24 Views of Mount Fuji* (1985) [1]

Abstract. The problem of data semantics is establishing and maintaining the correspondence between a data source and its intended subject matter. We review the long history of the problem in Databases, and contrast it with recent research on the Semantic Web. We then propose two new directions for research on the problem and sketch some open research questions.

1 Introduction

Two panels, held at SIGMOD'98 (Seattle, June 4) and CAiSE'98 (Pisa, June 11), discussed the topic of data semantics and its place in Databases research in the next millennium. The first, titled "Next Generation Database Systems Won't Work Without Semantics" included as panelists Philip Bernstein, Umesh Dayal, John Mylopoulos (chair), Sham Navathe and Marek Rusinkiewicz. The second one, titled "Data Semantics Can't Fail This Time!" included as panelists Michael Brodie, Stefano Ceri, John Mylopoulos (chair), and Arne Solvberg.

Atypically for panels, participants to both discussions generally agreed that data semantics will be *the* problem for Databases researchers to tackle in the near future. Stefano Ceri summed up well the sentiments of the discussions by declaring that

> *"... The three most important research problems in Databases used to be 'Performance', 'Performance', and 'Performance'; in years to come, the three most important and challenging problems will be 'Semantics', 'Semantics', and 'Semantics' ... "*

What is the data semantic problem? In what sense did it "fail" in the past? ... "And why did the experts agree – unanimously – that the situation was about to change?

We review the data semantics problem and its long history in Databases research, noting the reasons why solutions of the past won't work in the future. We then consider recent work on the Semantic Web and the directions it is taking. Finally, we sketch two new directions for research on data semantics.

C. Bussler et al. (Eds.): SWDB 2004, LNCS 3372, pp. 9–26, 2005.