



Service Oriented Architecture – Overview of Technologies and Standards

Dimka Karastoyanova, Frank Leymann, University of Stuttgart, Germany

In constantly changing and dynamic markets enterprises are in an even greater need to maintain adequate IT support for their core competencies. This support requires integrating heterogeneous systems and applications that are built using different technologies and infrastructures, which hamper interoperability and seamless integration. The problems are aggravated additionally by the need to cross enterprise boundaries and the necessity to react to changes in that same environment in an ad-hoc manner. Service-oriented computing (SOC) provides a powerful abstraction basically allowing to perceiving all compute resources as entities that can be dynamically discovered and composed. These entities are called *services* in SOC: a service is exposed for use over a network, and it is always ready for use. Services are described in terms of interfaces specifying service functionality independent of platform technology or programming language used. This renders the service abstraction particularly advantageous when applied for tackling problems due to heterogeneity of IT landscapes. Service oriented archi-

ture (SOA) is the accompanying architectural style. The roles in the SOA are: service provider, service consumers and service registry. Applications compliant with the SOA can be implemented in terms of various technologies and programming languages. Services communicate through messaging and are loosely coupled. The single technology that has been created to implement SOA inherently and from its onset is the Web service technology. It provides an abstract component model for representing and using services. Based on the heterogeneity and interoperability problems to solve, this technology is highly dependent on standardization; it enables reuse of legacy applications and aims at solving integration problems.

The Web Service technology defines a stack of composable specification addressing orthogonal concerns that can be easily composed to meet the requirements of various application domains. Web services are the only service-oriented technology that has enjoyed a tremendous success in acceptance by industry and academia. The implication of its composable nature is that the execution infrastruc-

ture for service-oriented applications must support that composability of protocols. Web services mainly target machine-to-machine interaction, which is driven by the need to enable high degree of automation. The execution infrastructure for such applications is referred to as (Enterprise) Service Bus (ESB). It is a piece of middleware enabling the application of the concepts of service orientation.

The papers in this issue of *it* give an overview of the most important aspects of SOA from the point of view of both, industry and academia.

The paper of *Dustdar* and *Papazoglou* provides definitions of what services and composite services are and illustrates the basic principles of service-oriented computing and how they are reflected by the Web service technology. The distinguishing differences between Web services and other programming models are discussed. This overview paper compares the provision of enterprise solutions using the application service provider (ASP) model and the service-oriented computing approach, whereas the major benefit of services is revealed by their



flexibility in reuse and much easier integration. Aggregating functions in more complex ones is supported by service-oriented computing in terms of a recursive integration model. A short overview of service composition in service-oriented environments and its implication on the execution infrastructure are also in the focus of this paper. The two approaches towards service composition that differ in their intended use – orchestration and choreography – are explicitly defined and investigated, while more details on service choreography are discussed in the paper of *Barros, Decker, and Kopp*. Furthermore a short overview of the Web service standards and specifications for description, messaging formats, discovery and composition is given and open research challenges are identified. All these technologies are presented in much more details in the subsequent papers of this *it* issue.

For example, the paper of *Linton et al.* summarizes the current Web service technology standards and their implications on the middleware implementing them. The functionality of an Enterprise Service Bus (ESB) is clearly specified and a particular emphasis is put on the open-source infrastructure development.

Middleware hides idiosyncrasies of the standards it implements, including the support for non-functional properties. The paper of *Curbera et al.* focuses on presenting standards and approaches of addressing QoS in SOA environments and in particular by the Web service technology. The article focuses on the non-functional aspects of service behaviour and their description in standardized contracts. The term “quality of service” is used to represent non-functional aspects of service behaviour. This corresponds essentially to the aspects described using the Web Services Policy specification (WS-Policy) that are used complementary to functional aspects described by the Web Services Description Language (WSDL). The

paper focuses the discussion on traditional middleware protocols such as security, transactions and reliable messaging. In addition, the support for coordination protocols, being also a concern of middleware, is extensively discussed.

To enable complex business functionality, services can be composed in more complex services. WS-BPEL (Web Service Business Process Execution Language) is the standard in the WS protocol stack that deals with composing Web services using a process-based approach. BPEL enables a recursive composition model, which can flexibly create complex business functionality and expose it again as Web services. BPEL is an extensible language and can be enhanced with additional features that are not present in the core specification. Motivated by the needs to support human interactions as well as modularization of process models two such extensions have been provided recently: The involvement of human participants in processes is enabled by the BPEL4People specification and the accompanying WS-Human Task specification. Reuse of processes and process fragments has led to the development of the BPEL-SPE specification; it enables the use of stand-alone processes as sub-processes and the lifecycle control of the parent process over the child process, which is regulated by a coordination protocol. The work of *Kloppmann et al.* published in this *it* issue presents an overview of the currently available specifications for process definition and execution, in particular WS-BPEL, BPEL4People, and BPEL-SPE.

The variety of data models in the different business domains implies the need of a richer semantics when discovering appropriate services in an SOA environment. To enable a more powerful discovery and hence composability capabilities of services, in recent developments the SOA reference model has been refined to support the use of semantics in that field. Semantic

Web services are the technology enabling these features. Semantic Web services enhance the Web service technology with semantic descriptions of the functionality services provide independent of their signatures, which introduces an additional abstraction layer in the description of service functionality. In this respect, the paper of *Norton et al.* presents a reference ontology and architecture for semantically enriched SOA.

In the paper of *Barros, Decker, and Kopp* an overview of the basic principles of service orchestrations and choreographies is presented. Particularly, the authors pay attention to the relationship between the lifecycles of orchestrations and choreographies being the major difference between the two. A taxonomy of existing choreography languages is presented that classifies these languages according to the interconnection models and interconnected interface behaviour models they support. The authors use a comprehensive example to illustrate the different models used by different choreography languages. Furthermore they present the choreography languages BPEL4Chor and WS-CDL in detail.

Each of the papers discusses the major open issues in their domain. These include QoS models and their use in service composition, compliance and adaptability of services and service compositions, both orchestrations and choreographies. Coordination protocols for complex interactions among composite services in cross-organizational settings are still being investigated only in research. Complex fault handlings in particular in choreographies are also challenges to be dealt with in future. In addition, in the field of semantic Web services the problems of mediation using knowledge described in term of semantic descriptions of services and processes is yet to be addressed. The solutions to the above mentioned deficiencies will pave the way to enabling automatic composition of services according to

use case requirements; adaptability of such service-based applications will also be enabled by finding such solutions.



1

1 Dr.-Ing. Dimka Karastoyanova received a Master's Degree in Computational Engineering from the University Erlangen-Nuremberg, Germany. She received a PhD in Computer Science from the Technische



2

Universität Darmstadt, Germany, in 2006. Currently she is a teaching and research assistant at the Institute of Architecture of Application Systems at the University of Stuttgart, Germany. Her research interests lie mainly in the fields of service oriented computing, middleware, the Web Service technology and process-based applications. Of particular interest are novel approaches to enabling adaptability and flexibility of Web Service based processes in terms of both process languages and execution infrastructure.

Address: Institute of Architecture of Application Systems (IAAS), Universität Stuttgart, Universitätsstraße 38, 70569 Stuttgart, Germany, E-Mail: dimka.karastoyanova@iaas.uni-stuttgart.de

2 Prof. Dr. rer. nat. Frank Leymann is a full professor of computer science and director of the Institute of Architecture of Application Systems at the University of Stuttgart, Germany. His research interests include service oriented computing and middleware, workflow- and business process management, programming in the large, transaction processing, integration technology, and architecture patterns. Before accepting his professor position he worked for two decades for IBM Software Group building database and middleware products; from 2000 on he was member of a small team of architects creating the core of the Web Service platform.

Address: see above, E-Mail: frank.leymann@iaas.uni-stuttgart.de