From Service-Oriented Architecture to Service-Oriented Enterprise

Marten van Sinderen

University of Twente, Department of Computer Science
Enschede, The Netherlands
m.j.vansinderen@ewi.utwente.nl

Abstract. Service-Oriented Architecture (SOA) was originally motivated by enterprise demands for better business-technology alignment and higher flexibility and reuse. SOA evolved from an initial set of ideas and principles to Web services (WS) standards now widely accepted by industry. The next phase of SOA development is concerned with a scalable, reliable and secure infrastructure based on these standards, and guidelines, methods and techniques for developing and maintaining service delivery in dynamic enterprise settings. In this talk we discuss the principles and main elements of SOA. We then present an overview of WS standards. And finally we come back to the original motivation for SOA, and how these can be realized.

Keywords: Service-Oriented Architecture, SOA Principles, SOA Patterns, Service-Oriented Computing, Web Services, Service-Oriented Enterprise.

1 Introduction

Service Oriented Architecture (SOA) was originally motivated by the need of enterprises to better match information systems with their business goals, combined with the market trend of more and more flexible cross-organizational collaboration between enterprises [6]. Vertical integration (business-IT alignment) and horizontal integration (IT supported cross-organizational collaboration) are considered crucial for modern enterprises, but traditional IT architectures have serious integration deficiencies. Architectures often comprise monolithic (silo) applications that are effective for the specific purpose they were created, but which do not allow integration without custom coded connections. Architectures with component-based applications provide units of business logic, which ease the definition of connections, but still require that the flow of control and the transformation of data formats are bound into the business logic.

SOA is an IT architectural style that tries to achieve integration by way of defining composite applications as an orchestration of services, with services potentially offered by different organizations. A service externalizes public functions of an application that implements a repeatable business task. Since a composite application can also be offered as a service, integration may involve multiple levels of composition, and a service can be internal to an organization or cross-organizational.

van Sinderen M.
From Service-Oriented Architecture to Service-Oriented Enterprise.
DOI: 10.5220/0004463600030013
In *Proceedings of the 3rd International Workshop on Enterprise Systems and Technology (I-WEST 2009)*, pages 3-13
ISBN: 978-989-674-015-3

This short paper aims at surveying the concepts and architectural elements of SOA, and investigating to what extent existing standards supporting SOA enable and have created service-oriented enterprises. In this context, we mean with service-oriented enterprise a business organization whose business and IT are well-aligned to flexibly engage, operate and disengage in cross-organizational collaborations and be (more) effective in the given market by using and providing services according to SOA.

The remaining of this paper is structured as follows: Section 2 provides an overview of SOA concepts, architectural elements and principles; Section 3 briefly discusses the standardization of Web Services, constituting one of the now widely adopted technologies to implement SOA; Section 4 looks into the impact that the adoption of Web Services has on organizations and whether this turned them into genuine service-oriented enterprises; and Section 5 summarizes our main findings.

2 SOA Foundation

The central concept of SOA is, of course, 'service'. There are several possible interpretations of 'service', partly due to the fact that SOA addresses two distinct disciplines, which already have existing and different uses of the term for some time. In a business context, a service involves the exchange of some action, performance or promise for value between a client and provider [13]. Examples are transportation services, health services, education services, outsourcing services, and helpdesk services. In an IT context, a service refers to the external behavior of an IT system, as can be observed and experienced by the users of that system [12]. Examples are data communication services [15] and application services [1]. For convenience, we will use the terms 'business service' and 'IT service' to distinguish between the business view and the IT view on services.

SOA holds the promise to bring business and IT together, by repeated aggregation of IT services into composite applications supporting business services that in turn are aggregated into business processes [14]. Figure 1 shows the basic architectural pattern that underlies SOA. In this pattern, three roles are distinguished: service provider, service broker and service requestor [10]. A service *provider* offers one or more services, which may be implemented using arbitrary technologies and involving backend systems protected by a firewall. Each service has well-defined interfaces referred to in a service description. Service descriptions may be published with a service *broker*, thus opening the possibility for service requestors to find services by providing required service properties to the service broker. The service broker searches for service descriptions that satisfy the required service properties, and the service requestor can select from the result of this search. Based on the location/access details in the service description, the service *requestor* can then bind to a service provider that offers the selected service. After a successful binding, the service requestor can invoke the service, according to the interface details in the service description.

Using this pattern, vertical integration is tackled by presenting a service as a virtual component that can be implemented by alternative concrete components using different technologies. The service requestor is therefore decoupled from

implementation concerns of the service provider. Using SOA for application design and proving a service wrapping for legacy applications thus presents a viable approach to Enterprise Application Integration (EAI).

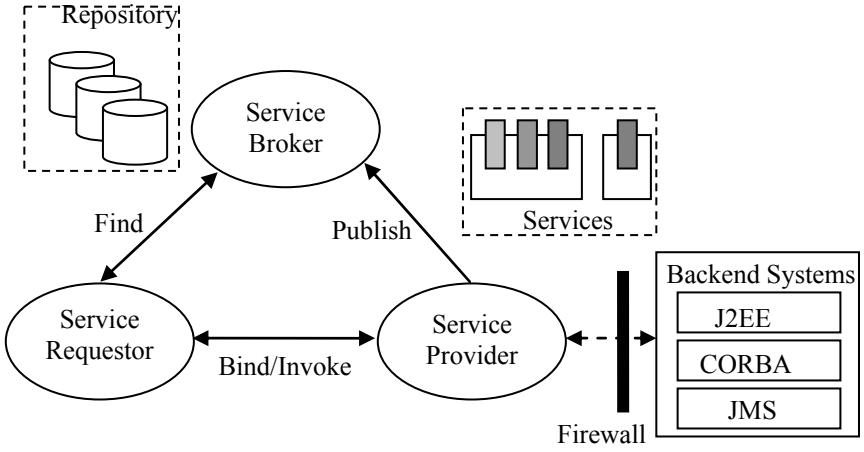

Fig.1. Basic SOA pattern.

Vertical integration, or business-to-business (B2B) integration, requires that each potential business partner defines a public view on its private process, with corresponding services and associated incoming and outgoing message exchanges that allow linking to external partners. The previously presented basic SOA pattern only shows a single service provider and a single service requestor role. In a B2B collaboration scenario, business partners may play either role for any number of supported services. An individual partner coordinates the services used and provided through its private process. Since this in general does not determine the overall coordination involving all partners, a coordination protocol can be defined that concerns the public view on how the partners should work together. Figure 2 shows an example of a SOA-based business collaboration with three partners whose processes are connected through services coordinated in compliance with some coordination protocol.

A coordination protocol, such as the one depicted in Figure 2, does not provide a concrete and executable process for the coordination of service. It only defines the order in which messages should be exchanged, where messages are used to invoke a service or return a service result in accordance to a service provided by one of the partners. A definition at this level of abstraction is also referred to as service *choreography*. If, on the other hand, this definition would be refined into a concrete process, which can be assigned to and executed by some computing node, we use the term service *orchestration* instead [11]. When assigned to a node, this node can in turn offer the external functionality of its process as a service. This service thus allows service requestors to invoke and use the coordinated behavior of several services, while hiding how the composition of services is achieved and which service providers are offering these services.

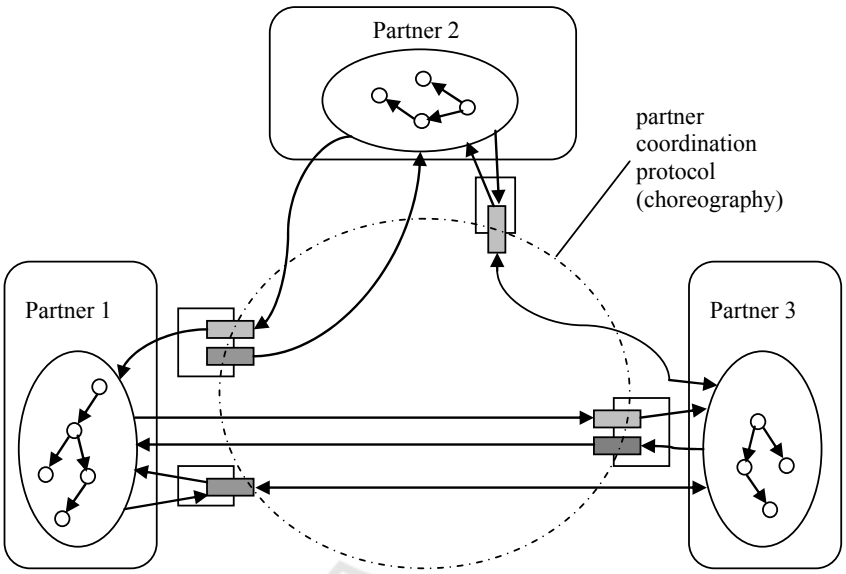

Fig.2. SOA-based business collaboration using a coordination protocol.

Since the principle of encapsulating processes that compose services can be repeatedly applied, we can build a hierarchy of service aggregations, ranging from simple generic IT services to complex dedicated business-oriented services. Figure 3 shows such a hierarchy, illustrating how SOA supports a way of integrating business as linked services. Although SOA itself does not imply or propose any methodology for designing IT support for business activities, it does make clear that business processes can be seen as a driver of collaboration with services playing a central role at all levels.

Design methodologies centered on SOA [9] should then include an analysis phase that reviews identified business processes with respect to the extent to which SOA can contribute to improvement and adding value. If SOA is deemed to play a role, business services should be identified that represent this SOA potential. In a subsequent design phase, service interfaces should be defined as well as processes that can orchestrate services based on their interfaces, and basic IT services should be identified. Both functional and non-functional (performance, reliability, availability, etc.) requirements on services should be considered during this phase, and legacy applications may be leveraged as service if they match such requirements.

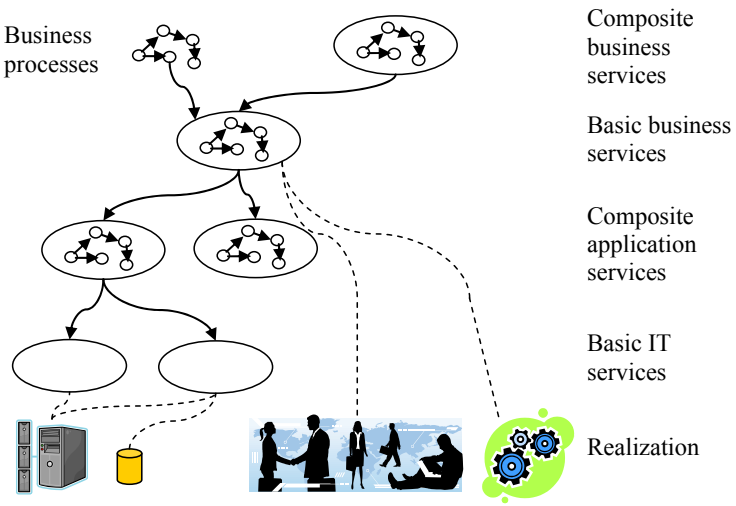

Fig.3. SOA-based business integration by way of linked services.

The above sketch of a design methodology is also useful to illustrate the importance of the guiding principles of SOA [3]:

- *Loose coupling:* a service is defined independent of its implementation and usage context. This means that a service requestor does not have to be aware of the technology used to implement the service, and the service provider has no a priori knowledge of the service requestor. As a consequence, requestors and providers can evolve independently, without affecting interoperability, provided that service (interface) definitions are adhered to.
- *Re-use:* a service is a unit of functionality which is potentially useful in many different contexts and applications. Having service descriptions stored in repositories, which a service broker can search in order to find a service that matches properties specified by a service requestor, further helps to promote reuse.
- *Composable:* the invocation of services can be coordinated and the results can be composed to form composite applications. The functionality of composite applications can in turn be exposed as services, which permits hierarchical composition with different degrees of software reuse and business specificity at each level.
- *Standards-based:* the above mentioned architectural principles can only be realized if technology standards are available that allow services to be described, published, invoked, composed etc. This is the topic of the next section.

3 Web Services

Web services (WS) are a collection of emerging standards, which are widely accepted as the technology of choice for implementing SOA [10]. Web services to a large extent supports the concepts, patterns and principles mentioned in section 2. An application designed and implemented according to WS standards is self-contained and modular, has a description which can be published, can be found on basis of its description, and can be located and invoked over networks.

The core WS standards are the following:

- *Simple Object Access Protocol (SOAP):* this is the Internet protocol for Web (service requestor and service provider) applications to communicate. It suns on top of other standard Internet protocols, including HTTP. SOAP defines how messages are structured and processed in a platform-independent way. It comprises two message exchange patterns, viz. one-way and request-response.
- *Web Service Description Language (WSDL):* this is the language for specifying he interface of Web services. It is used to provide a description of the service for the (potential) service requestors. Such a description includes information on which messages are related to each operation that is supported by the service, how these messages are related (e.g., operation input and output), and how SOAP messages are exchanged.
- *Universal Description, Discovery and Integration (UDDI):* this standard is defined to enable the storage of information for organizing and discovering Web services. UDDI consists of data structures and APIs for publishing and querying Web services. The UDDI APIs are themselves Web services, and thus are described and can be invoked as any other Web services.

In addition, all WS standards rely on the Extensible Markup Language (XML) to represent structured data. XML documents and schemas are defined to standardize the format and typing of data communicated by Web services.

The basic SOA pattern explained in the previous section can be supported with SOAP, WSDL and UDDI. These standards are, however, not sufficient to correlate messages exchanged between a service requestor and a service provider, to distinguish between multiple instances of the same service, or to coordinate the use of different services. Also they do not address policies that govern the use of Web services, non-functional aspects of Web services such as reliability, security and atomicity. For this purpose, several other WS standards have been developed.

This paper has not the intention to discuss these standards even at a high level of abstraction. Instead, we argue that WS standards are becoming widespread and have reached a certain level of technical maturity. In addition, we can observe that WS standards pretty much cover all the important technical areas that were identified for SOA. Figure 4 shows an overview of standards supporting different aspects of SOA.

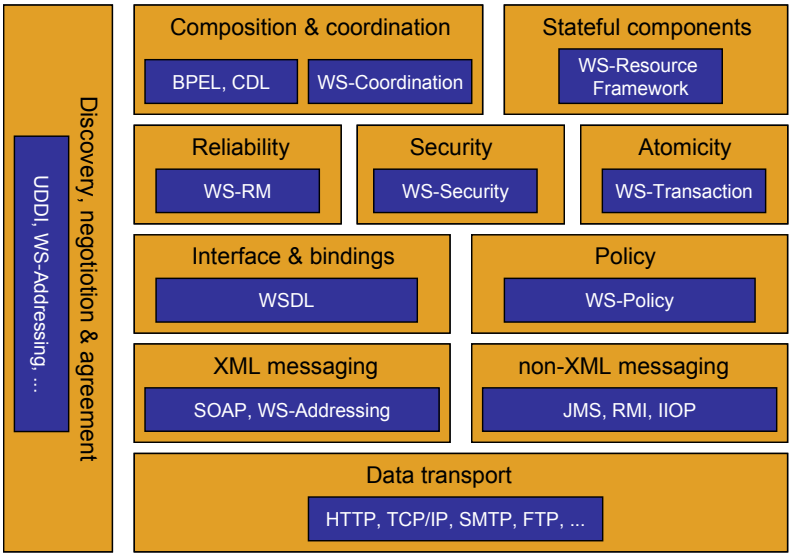

Fig.4. WS (and some other) standards supporting SOA.

The acceptance and the technical maturity and coverage of Web services provide no guarantee that the business objective of SOA is also realized [8]. Web services may be used to extend the existing technology infrastructure with a new layer on top, facilitating technology-level interoperability, integration and maintenance, but overall benefits may still be small if business processes remain unaffected due to a lack of 'service thinking' at business and application level. We will address this further in the next section.

4 Service-oriented Enterprise

Several major technology vendors have invested significant effort in supporting and promoting SOA and corresponding technology standards[1]. As a result, SOA is now generally accepted as a useful architectural style, and adoption of Web services is widespread. Also business-level awareness for SOA has been created, thanks to technology trend and market analysis reports that claimed, among others, the necessary adoption of SOA for most companies in order to stay competitive[2].

Nonetheless, SOA adoption in practice often boils down to the use of Web services as an enabling technology, whereas service-oriented business that applies SOA principles and exploits the potential benefits of SOA technology is less commonplace. Some researchers report that impact of SOA on business organizations and business processes is so far rare and limited [5]. This contrasts with the often heard claim that SOA will change the way business is done and organized. It is argued that not the introduction of new technology, but the application and

[1] See, for example, www.w3c.org and www.oasis-open.org.
[2] See Gartner's press releases over the years: www.gartner.com/it/products/newsroom.

management of that technology delivers real business benefits. Therefore, business itself should be transformed by 'service thinking', leading to added value and innovation. One reason for the still existing mismatch between enabling technology and business exploitation with respect to SOA may be the weak link between business executives and their company's IT organization [2].

The term 'service oriented enterprise' has been coined to refer to business organizations that pursue an optimal business-IT integration using SOA principles and technology [8]. Accordingly, we characterize a service-oriented enterprise as an enterprise that uses service-oriented technology (such as Web services) and that organizes its business model and processes to profit most from the potential benefits of this technology.

There have been recent reports on failed SOA projects and statements that for this reason and because of the current recession SOA popularity is on its return. However, SOA projects often focus too much on the technology to be used, and disregard project management. Gartner forecasts that lack of working SOA governance arrangements will be the most common reason for SOA failures[3].

Also, companies may loose initial enthusiasm if they learn that the introduction of SOA may be expensive, and that building their first SOA application may take longer than building the same application using traditional approaches and existing technology. However, subsequent SOA applications and changes to existing SOA applications can be expected to be less costly. This is inherent to any evolutionary approach. SOA offers no one-time gain, no immediate return on investment, but promises benefit over time [4].

Independent on their success or failure, a handful of SOA applications within a company cannot prove much about SOA. Companies should be aware of SOA principles, have strategies and practices in place, and persistently apply them throughout their business. In other words, they should become service-oriented enterprises. This led to the development of SOA maturity models [2, 8, 4], to position enterprises with respect to their service orientation and to provide a roadmap towards higher maturity levels. For example, in [8] the following levels of maturity are identified:

- *Usable:* an organization has standards and protocols that are usable across the organization's platforms and technologies.
- *Repeatable:* an organization has the capabilities to develop, deploy and maintain services, and scale the use of services.
- *Supportable:* an organization has the capabilities to provide and maintain services for its mission-critical applications.
- *Extensible:* an organization has the capabilities to apply service aggregation and realize business agility, and can provide this directly to customers and/or partners through services that generate new revenue channels.

Although it is difficult if not impossible to precisely assess and score the maturity of an organization, there is general agreement that maturity models are useful as a roadmap to improve upon a current situation. Achieving a higher maturity requires organizational actions, such as establishing proper IT directives, governance policies

[3] Gartner press release April 2, 2009.

etc. In general, also several technical obstacles and issues need to be addressed in order to transform into a service oriented enterprise, including performance and Quality of Service (QoS) [4, 7]. Mission-critical applications have to meet certain minimal QoS requirements. Determining what exactly are the QoS requirements for SOA applications, and how to specify, negotiate and monitor Service Level Agreements (SLAs), is a major and complex task [7]. Especially QoS management in composite applications is a critical issue for SOA systems, since service aggregation is the cornerstone for reuse and agility. So far there are only some academic studies in this area [14], and little empirical data.

Despite experienced setbacks and still existing obstacles, Gartner recently claimed that SOA is emerging from the Trough of Disillusionment within Gartner's hype cycle, and is climbing the Slope of Enlightenment[4]. This phase of the hype cycle is entered if mainstream organizations start to establish best practices to effectively use a technology and begin to experience benefits.

5 Summary

SOA is an IT architectural style that tries to achieve integration by way of defining composite applications as an aggregation of services, with services potentially offered by different organizations. Integration has a vertical (business-IT alignment), horizontal (cross-organizational interoperability) as well as a time (agility with respect to changes) dimension. The guiding principles of SOA are loose coupling, re-use, composability and reliance on standards. Web services constitute an emerging set of standards which are widely adopted as technologies to implement SOA. The acceptance and the technical maturity and coverage of Web services provide no guarantee that the business objective of SOA is also realized. In order to realize this business objective, i.e. to achieve integration paired with productivity benefits, companies should become service oriented enterprises. Companies should be aware of SOA principles, have strategies and practices in place, and persistently apply them throughout their business. SOA should be understood as an architecture, not as a technology. Technology, such as provided by Web services, is enabling, but not realizing the potential benefits of SOA. Consequently, only introducing a Web service technology infrastructure and blindly converting existing applications to become service-enabled is not enough. Business should determine which applications should be service-oriented, and have good governance in place to help decision-making. SOA maturity models can help to provide a roadmap to transform into a service oriented enterprise.

References

1. Almeida, J.P.A., van Sinderen, M.J., Ferreira Pires, L., Quartel, D.A.C. The role of the service concept in model-driven applications development. In: *Workshop Proceedings of*

[4] Gartner press release April 2, 2009. For an explanation of hype cycles, see also www.gartner.com/technology/research/methodologies.

International Middleware Conference - First Workshop on Model-driven Approaches to Middleware Application Development (MAMAD 2003), PUC-Rio, 2003, pp. 288-296.
2. Arsanjani, A. and Holley, K. The service integration maturity model: achieving flexibility in the transformation to SOA. In: *Proceedings of IEEE International Conference on Services Computing* (SCC 2006), IEEE Computer Society, 2006, pp. 515-515.
3. Erl, T. *SOA principles of service design*. Prentice Hall, 2007.
4. Inaganti, S. and Aravamudan, S. SOA maturity model. *BPTrends*, April 2009, 1-23.
5. Luthria, H. and Rabhi, F. Service oriented computing in practice - An agenda for research into the factors influencing the organizational adoption of service oriented architectures. *Journal of Theoretical and Applied Electronic Commerce Research*, 2009, 4(1): 39-56.
6. OASIS. *Reference model for service oriented architecture 1.0*. OASIS standard, 12 October 2006. URL: http://docs.oasis-open.org/soa-rm/v1.0/
7. O'Brien, L., Brebner, P., Gray, J. Transformation to SOA: aspects of the migration and performance and QoS issues. In: *Proceedings of Second International Workshop on Systems Developments in SOA Environments* (SDSOA 2008), ACM, 2008, pp. 35-40.
8. Oelermann, W. Enabling the service-oriented enterprise. *Microsoft Architect Journal*, No. 7, April 2006, 27-32.
9. Papazoglou, M.P. What's in a service? In: *Proceedings of First European Conference on Software Architecture* (ECSA 2007), LNCS 4758, Springer, 2007, pp. 11-28.
10. Papazoglou, M.P. *Web services: principles and technology*. Pearson Prentice Hal, 2008.
11. Peltz, C. Web services orchestration and choreography. *IEEE Computer*, 2003, 36(10): 46-52.
12. van Sinderen, M.J. and Ferreira Pires, L. Protocols versus objects: Can models for telecommunications and distributed processing coexist? In: *Proceedings of Sixth IEEE Workshop on Future Trends of Distributed Computing Systems* (FTDCS 1998), IEEE Computer Society, pp. 8-13.
13. Spohrer, J., Maglio, P. P., Bailey, J., and Gruhl, D. Steps toward a science of service systems. *IEEE Computer*, 2007, 40(1): 71-77.
14. Unger, T., Mietzner, R., Leymann, F. Customer-defined service level agreements for composite applications. *Enterprise Information Systems*, 2009, 3(3): 369-391.
15. Vissers, C.A. and Logrippo, L. The importance of the service concept in the design of data communications protocols. In: *Proceedings of Fifth IFIP WG6.1 International Conference on Protocol Specification, Testing and Verification* (PSTV 1985), Elsevier North-Holland, 1985, pp. 3-17.

Brief Biography

Marten J. van Sinderen is associate professor at the Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS) of the University of Twente, Enschede, The Netherlands. He is a member of the Information Systems (IS) group since 2008, and before that led the Architecture and Services of Network Applications (ASNA) group between 2004 and 2008. He is currently also coordinator of research in the area of Service Architectures and Health Applications, on behalf of the Centre for Telematics and Information Technology (CTIT), the ICT research institute of the University of Twente.

During his professional career, he has been active in the areas of network interconnection, communication protocols, middleware, application protocols, and enterprise interoperability. His current main research interests are design methods and architectures for networked systems, particularly mobile middleware, service platforms, and context-aware mobile applications. Among the design paradigms being considered are Service Oriented Architecture, Model Driven Architecture and the Semantic Services. He was co-chairman of the Program

Committee of EDOC 2004, and general (co-) chair of IDMS 2000, PROMS 2001, EDOC 2005 and EUNICE 2007. He has been a member of the steering committees of IDMS, PROMS, MIPS and EDOC, and involved in numerous program committees of major international conferences, including INFOCOM 2006 and ECMDA 2005, 2006 and 2007. He participated in European initiatives/projects including MODA-TEL (Model Driven Architecture for Telecommunications Systems Development and Operations, IST 37785), E-NEXT (Emerging Networking Experiments and Technologies, IST 506869), AMIGO (Ambient Intelligence for the Networked Home Environment, IST 004182), and SPICE (Service Platform for Innovative Communication Environment, IST 027617). He currently leads the Dutch Freeband A-MUSE project (BSIK 03025) on model-driven service design for context-aware mobile applications, and the Dutch GenCom U-Care project (IGC0816) on user-tailorable healthcare services for the home environment.

He was invited reviewer for the European Union of C-ARCTIC (Concurrent Environment and Architecture for Telecollaboration Integrated in the Company, IST 1999-20087). He is a member of the Editorial Board of the Enterprise Information Systems journal, published by Taylor & Francis. He is also member of the Managerial Board of IFIP WG5.8 on Enterprise Interoperability.