# The strategic impact of service oriented architectures

Philipp Liegl
Research Studios Austria
Thurngasse 8/20, 1090 Vienna, Austria
philipp.liegl@researchstudio.at

## Abstract

*It has not been since the advent of the client/server architecture break through that an architectural concept has changed the face of enterprise systems so significantly as it has been done by service oriented architectures (SOA). The service oriented approach provides plenty of vantages for companies in regard to flexible system integration and adoption of new business cases. However, the adoption of SOA in an actual enterprise system brings along a couple of problems as well. Especially the integration into the existing infrastructure, applications and the innovation, sourcing and investment policies is challenging. A solution can be provided by establishing a SOA roadmap unveiling possible traps and pointing out the foibles and flaws still existing in the SOA approach. In this paper the SOA approach will be reviewed critically and the different sections affected within an enterprise will be examined. Possible problems during the transition and use of SOA will be identified. Where already possible, solutions will be provided. This paper is based on current research conducted during my PhD studies.*

## 1. Introduction

Architectures in information systems have seen many changes over the years beginning from monolithic mainframe applications to the latest development - service oriented architectures (SOA). In a service oriented context program functionality is exposed via an interface which is accessible over a network. These interfaces are usually referred to as "services". A service in the context of computer science is an encapsulation of business logic accessible via XML messages over a network and must not be confused with the idea of a service in the context of business administration. In the context of SOA a service is coarse-grained and loosely coupled and it can be reused in several contexts. The lifecycle-management of a service, from its creation until it is abandoned, is part of a service oriented architecture. A specific infrastructure allowing the different applications to exchange data via services is also defined. The most known standards in use for a SOA today are WSDL (Web Service Description Language) [16], SOAP (Simple Object Access Protocol) [17], BPEL (Business Process Execution Language) [1], WS-CDL (Web Service Choreography Description Language) [18] and UDDI (Universal Description, Discovery and Integration) [10] which became de facto standards for a service oriented architecture. Nevertheless, a service oriented architecture must not necessarily be build with these standards. However, the compliance to industry standards, the use of available tools and a maximum of interoperability benefits strongly suggest their use.

The use of XML as the basic format for the exchange of messages and information facilitates the communication between business partners. Typical technical boundaries which prevented B2B eCommerce in the past such as different systems or communication heterogeneity are circumvented by the use of XML. An enterprise planning to adapt

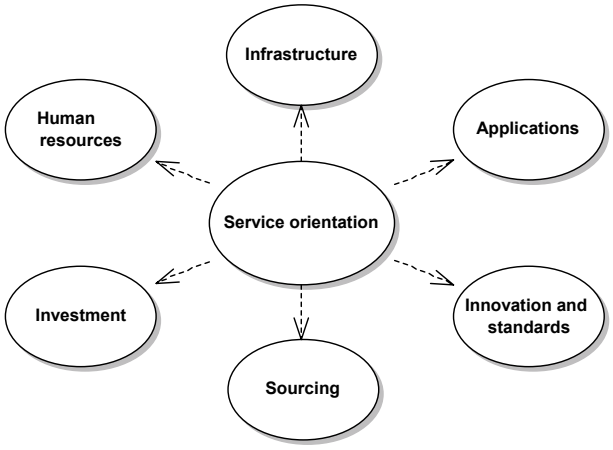


**Figure 1. The influence of SOA**

service orientation for its IT department has to consider several crucial facts. Looking at the architecture simply from a technical perspective is not enough. The business perspec-

Proceedings of the 14th Annual IEEE International Conference and
Workshops on the Engineering of Computer-Based Systems (ECBS'07)
0-7695-2772-8/07 $20.00 © 2007 IEEE

tive and the application perspective must also be taken into account. What is needed, is an overall strategy embracing all three perspectives and guaranteeing a successful transition and seamless integration. In order to avoid possible traps which might occur during the transition to service orientation a SOA roadmap should be established. The several sections of an enterprise affected by the installation of a service orientation are covered by such a roadmap. Although no absolute guarantee for success a roadmap greatly facilitates the transition from a classic client/server architecture to a service oriented approach.

## 2. A roadmap to a service oriented architecture

A transition to SOA will have an influence on several divisions and departments within an enterprise. Figure 1 gives an overview about the different areas being affected by the introduction of service orientation. Infrastructure and applications in use will experience the major changes. However, innovation and standards, sourcing, investment and human resources will also be affected. A service oriented architecture within an enterprise will not remain a self-contained IT-pattern applied to hard- and software, but become a paradigm embracing all parts of the enterprise. The most important changes in every section will be examined in detail.

### 2.1. Infrastructure

Most of the IT infrastructures in use today are still based on the client/server pattern. The central point of all communication is a server or a mainframe system. These often legacy systems have been around and in use for several years. They are well-tried and the IT staff responsible for them is experienced - failure scenarios are well-rehearsed and occur rarely. The only major flaw these systems have is the inflexibility in regard to extensions and adaptation to new business scenarios. In a constantly faster changing business world such IT systems are not business enhancers but bottle-necks. A solution for more flexibility and faster conformance to new business scenarios is brought by SOA.

Service orientation in a broader sense however, is not really new. With components like CORBA (Common Object Request Broker Architecture) [11] or DCOM (Distributed Component Object Model) [3] service-like approaches were already realized years ago. Nevertheless, these systems were hard-wired and relied on certain standards and operating systems. An adaptation to a new business scenario i.e. through the acquisition of a new company could only be realized with major coding efforts. A real process oriented service composition was not possible.

Within a service oriented approach relevant core program functionality is extracted and offered through an interface. The interface can be accessed via XML messages - so called service providers are created. On the other hand the enterprise's systems also consume other services i.e. from subsidiaries or subcontractors. A service oriented architecture within an enterprise therefore consists of service consumers and service providers. Information about existing services is stored in a so called service registry where the information is made accessible to relevant stakeholders.

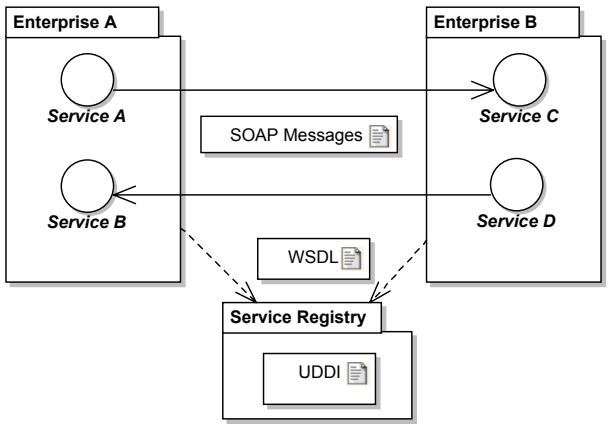


**Figure 2. A sample SOA environment**

Figure 2 shows a sample SOA environment. Enterprise A has two services - a service consumer (*Service A*) and a service provider (*Service B*). *Service C* of Enterprise B acts as a service provider and *Service D* as a service consumer. For the information exchanged between the different services SOAP messages are used. In order to retrieve information about existing services both enterprises can access a service registry which is based on UDDI. A service registry provides particular functionality, allowing the search and retrieval of relevant service information. Information about existing services is returned in the form of WSDL files. With the use of pertinent software user accessible interfaces can be build on the basis of the WSDL files. The advantage of a service oriented architecture is the flexibility to exchange services. Enterprise A could abandon *Service B* and set up *Service E* instead. In the business case of figure 2 such an exchange must first be communicated to Enterprise B. Any other enterprise willing to consume the new *Service E* however, would know about it because the WSDL information can be retrieved from the service registry.

In reality the use case scenarios of a service orientation are not as simple as described in figure 2. It is necessary to align different service calls, assemble different services to a new service etc. Figure 3 shows a more complex scenario. Enterprise A's *Service A* invokes *Service B* of Enterprise B. *Service B* is a composed service which itself calls two more services namely *Service C* and *Service D*. The orchestration of the calls is achieved via BPEL. With the use of BPEL the

Proceedings of the 14th Annual IEEE International Conference and
Workshops on the Engineering of Computer-Based Systems (ECBS'07)
0-7695-2772-8/07 $20.00 © 2007 IEEE

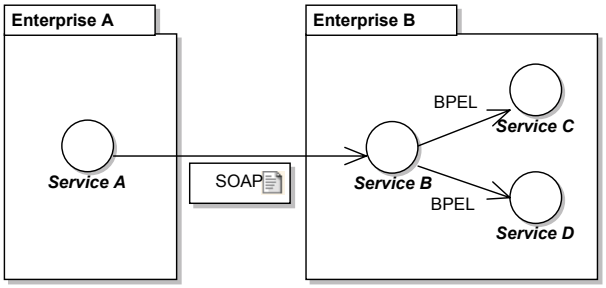


**Figure 3. Service orchestration with BPEL**

order in which specific services are called can be controlled. In the terminology of web services controlling of service call orders is referred to as *orchestration*. If the sequence of web service calls between two parties is controlled we refer to it as *choreography*. Orchestration must be viewed from a partner specific perspective whereas choreography must be viewed from an overall perspective between two participating parties.

It is not important for Enterprise A what specific software or application is hidden behind *Service B* as long as it serves the needs of Enterprise A. Enterprise B on the other hand is flexible in assembling the logic and functionality of *Service B* by simply changing the BPEL file, orchestrating the calls. The logic hidden behind *Service B* can easily be changed by Enterprise B to fulfill changing process needs required by Enterprise A. *Service B* could also be regarded as a certain interface covering the logic behind it. So far such a concept is not new and well established also in conventional client/server architectures. The major advantage of the SOA approach however, is the possibility to easily assemble a new logic behind *Service B* by simply changing the BPEL definition. Furthermore a new service could be set up by Enterprise B which might be called *Service E* and serves the specific needs of a fictional Enterprise C. *Service E* could reuse the existing services *Service C* and *Service D*. At this point the importance of a service oriented architecture becomes evident - it provides a maximum of flexibility and reuse.

A classic enterprise system based on a client/server architecture does not provide such a flexibility needed for fast integration of new business scenarios. Moreover it now also becomes evident that a pertinent infrastructure is a mission critical factor for implementing a service orientation.

The realization of such an approach requires significant changes in the overall system architecture and infrastructure. In order to guarantee a seamless transition, a SOA roadmap is needed. A SOA roadmap approach consists of four major phases which will be broken down into specific activities: *education*, *assessment and benchmarking*, *planning* and *roll out*. In every phase specific tasks are performed and artifacts are generated which are then used in the next phase. A similar proposal has already been made by SUN Microsystems in [13].

**Education:** With the increasing popularity of service orientation the perception of its concepts in many cases has became blurry. Two different people talking about SOA often mean two different things. However, a common understanding of the concepts and techniques is a mission critical factor. The business executives and more important the responsible persons within the IT department must have a clear understanding of SOA principles and concepts. Only if a common understanding of the relevant technologies and tools is available, an appropriate assessment and planning of a SOA in the next phases is possible.

**Assessment and benchmarking:** In this phase the IT representatives must assess the current state of the IT infrastructure within the enterprise. The major question to be answered will be "How ready is the company for SOA?". If possible, service oriented systems in use at subsidiaries or subcontractors should be analyzed in order to get a benchmark for the own future architecture. In particular the question whether to develop the SOA architecture in-house or to buy an external application and customize it to the specific needs must be answered.

**Planning:** Within this phase a detailed transition plan with the relevant software engineering and project management tasks must be elaborated. A SOA project group should be established, consisting of relevant stakeholders not only from the IT department but from all departments of the company.

**Roll-out:** In this phase the old architecture is gradually switched off and the service oriented approach is introduced. The transition from the old architecture to the new design is a dynamic process and therefore a vigilant supervision by the SOA project group is necessary.

A major goal of the current research will be the elaboration of a SOA roadmap. In particular the activities of the four phases will be analyzed in detail, while also pointing out possible shortcomings and foibles of the service oriented approach per se.

## 2.2. Applications

With service oriented architectures a new type of application has evolved as well - so called composite applications. Composite applications are similar to component-based software (CBS) focusing on building large software systems by integrating previously build software components. The nature of a composite application is the fact, that it is build by combining existing services provided by other applications. An example for a composite application is shown in figure 4. *Service B* can be regarded as a composite application, consisting of two services namely *Service C*

and *Service D*. When referring to the terminology used in the last paragraph a composite application is an assembly of service consumers. The newly created functionality of a composite application can then be exposed to other applications consuming its functionality. Hence composite applications can also be seen as service providers. The organization of the service composition is done within a so called composition platform. Again the major advantage is the possibility to exchange the services which are behind *Service B* by simply changing the service orchestration. When viewed in

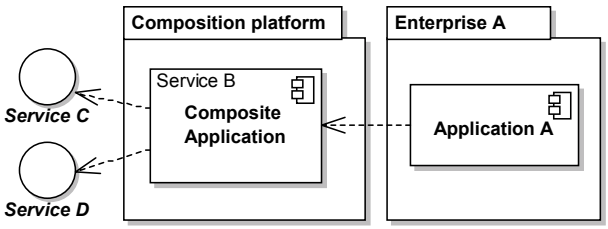


**Figure 4. A sample composite application**

a coarse-grained fashion, a composite application has three different layers: a user interface layer, a choreography layer and a service layer. Within the user interface layer the different forms and functions are presented to the user. The choreography layer defines the correct order in which the different services the composite application consists of, are called. Most likely BPEL will be the process execution language of choice to perform the orchestration. Services used by the composite application are represented by the service layer. The engineering of such applications will be a major challenge when introducing SOA in an enterprise. One very promising approach is the use of appropriate models to generate the software artifacts.

**Application development:** One major advantage promised by SOA is the rapid development of new service-based applications. Classic software development is often too complex and development of extensions or the adaptation to new business scenarios takes too long. Knowledge about the software engineering process however, is well accepted and known in the professional world though often only to programmers. In order to facilitate the creation of composite applications a model driven development (MDD) approach is helpful. On the first layer the user must be assisted by the MDD tool in creating a UI for the application. After the UI layer is finished the modeler must assign functionality to the various UI elements. This is done by connecting the UI layer to the processes represented by the second layer. On the second layer the different business processes are orchestrated in a specific order. The user retrieves the available services from a registry. Services can then be dragged and dropped onto a canvas of a modeling tool and be connected according to the specific needs. It is important to notice that the data returned by one service must not necessarily match the data required by another service. A solution is provided in the form of a so called "data mapper", allowing to transform the output of one service into the correct input of another service. The transformation will mostly be done from XML to XML. E.g. BPEL offers such a mechanism through the *assign* tag. After the process execution model is finished, the modeler can derive orchestration or choreography languages from the model. The languages most known for the orchestration and choreography of services are BPEL and WS-CDL. An MDA approach in the field of SOA has already been presented in [19].

**Derivation of service orchestrations:** BPEL describes the business process from a particular partner's point of view. In most cases, however, the service calls will cross enterprise boundaries and involve other entities such as subsidiaries or subcontractors. If each business partner describes the business process from his own point of view, the final process specifications will most likely not match. By using a more holistic modeling approach such as UMM (UN/CEFACT's Modeling Methodology) [14] the modeler can describe the inter-organizational business process in a semantically unambiguous way. A UMM model which is based on UML [12] can then be used to generate artifacts for a service oriented environment. We have already shown the feasibility of such an approach with BPEL in [6] which is based on research conducted in [4]. The transformation shown in [6] is limited to abstract processes in BPEL because UMM currently does not incorporate service bindings. Nevertheless with service bindings implemented, UMM promises to be a valuable design methodology for applications in a service oriented context.

It will heavily depend on the availability of appropriate tools, whether composite applications are successful or not. The development of composite applications is the second crucial issue after the infrastructural question, an enterprise is facing when introducing a service oriented architecture. Due to the importance of the applications used in a service oriented context, a potential SOA roadmap must incorporate a detailed strategy for the software to be used.

### 2.3. Innovation and standards

Whether the introduction of a service oriented architecture has an impact on innovation policies or not depends on several factors. If the enterprise is rather small, the introduction of a service orientation will most likely be outsourced and hence no real innovation takes place. For medium and large-sized enterprises the introduction of SOA increases the potential for innovation. Especially when the architecture is developed in-house and not outsourced the SOA approach can be regarded as an innovation promoter. Due to the joint effort all departments of the enterprise have

to make, new processes and techniques are developed ultimately leading to a new architecture - namely the service oriented architecture.

Most of the client/server systems are self-implemented solutions. The necessary programming languages and standards are well known to most of the IT staff today. Self-implemented software is often written on the basis of an enterprise server such as JBoss or similar products. The standards and techniques necessary to extend the software, are well known. With the rise of service oriented architectures however, a whole new set of technologies and methods starts to dominate the IT sector. Apart from the core XML-based standards already mentioned a whole new set of WS-* standards has been developed such as WS-Interoperability (WS-I), WS-Security (WS-S), Security Services (SAML) and Web Services Reliable Messaging (WSRM) just to name a few. With the increased proliferation of a service orientation such standards and their correct use become an important issue in IT departments. Although important, the current knowledge of IT staff in regard to these standards is quite low. One crucial part of a SOA roadmap must be the identification of the standards necessary. In the next step the knowledge level of the standards within the enterprise must be elicitated and necessary steps must be taken in order to overcome any knowledge gaps.

### 2.4. Sourcing

The effective implementation of a service oriented architecture very much depends on the capabilities of the IT department. In order to guarantee a seamless transition from the status quo to a service oriented environment, knowledge about service integration, composition and life-cycle management within the IT department is necessary. Especially in small and medium sized businesses such a knowledge is often not present. Whether a service oriented architecture is realized or not therefore often relies on a *make* or *buy* decision.

**The make decision:** When talking about a *make* decision the IT responsible refers to the in-house creation of specific software. In regard to service oriented architectures the advantage of a *make* decision becomes clear. The enterprise can choose the technology used and contribute with the internal know-how to the creation of the software. Furthermore, the integration of existing technologies and partners such as suppliers, subsidiaries and subcontractors is possible. The architecture created is tailored to the specific needs of the business and there is no dependence on external partners in regard to software maintenance and use. After the IT department has successfully carried out the in-house implementation of and transition to SOA a valuable amount of knowledge is available for further projects and potential for future synergy effects is given. On the other hand the *make* approach is cost and time-intensive and contains a certain amount of risk which must not be underestimated. In order to pursue a *make* decision specific domain knowledge, especially in the field of service engineering, orchestration and implementation is needed. If the knowledge is not available additional staff must be hired or existing staff be trained. The fact that such knowledge or staff is often not present leads us to the *buy* decision.

**The buy decision:** If the know-how of the IT department is not sufficient to implement a holistic service oriented architecture internally, an external partner and external software must be engaged respectively. Another important factor for the external development and implementation of a service oriented architecture is the lack of resources in the enterprise. Hiring new qualified employees just for the implementation of the service orientation contains a high financial risk whereas outsourcing the task minimizes the financial and the failure risk. The engagement of an external partner for the realization of a service oriented architecture is often cheaper and faster than the in-house realization. Although the planning and implementation of the service orientation can be outsourced, an enterprise should train in-house staff for the maintenance of the systems. As already mentioned before a service oriented context is volatile - services are abandoned or assembled to new services often within relatively short intervals. If necessary staff for service adaptation or correction is available in-house and the enterprise can react faster to new business needs.

### 2.5. Investment

A traditional investment, regardless if undertaken by department *x* or the IT department in most cases has one aim: a positive return on investment (ROI). Apart from being positive the investor expects the ROI to be quick. Considering the investments necessary for a service oriented architecture we can identify three different types according to [7]: organizational, architectural and infrastructural investments. Investments from the organizational perspective include putting in place new and specific processes such as service set-up or service deployment as well as human resources. The architectural investments include the evaluation, planning and testing of new software architectures. On the infrastructural side the investments mainly affect the software artifacts and the hardware necessary for a service orientation. All three investment types have one financial aim in common with the service oriented approach - the lowering of process costs. However, the initial investments for establishing a SOA architecture are higher than pursuing and maintaining a traditional architecture. A SOA project does not produce a quick ROI - hence it must not be induced on a ROI opportunistic basis. In the long-term however, the SOA approach produces a positive ROI due to in-

creased competitiveness, faster reaction on market changes and needs and lowered process costs. Furthermore the total cost of ownership of software is decreasing and the time-to-market of new products and services is lowered.

### 2.6. Human resources

The human resources sector will encounter significant changes with the introduction of a service oriented environment. In a classic client/server environment the development of new systems and applications was commissioned by the business executives to the responsible project managers. Project managers hired programmers who coded the applications according to the requirements elicitated. At the time the development of the application was finished and it was released the initial business requirements had often already changed. Adaptations to changes and extensions to the software were complicated and time and cost intensive. This classic view on the roles in software engineering must not be regarded as outdated - however, in a service oriented world a new paradigm prevails. Figure 5 gives

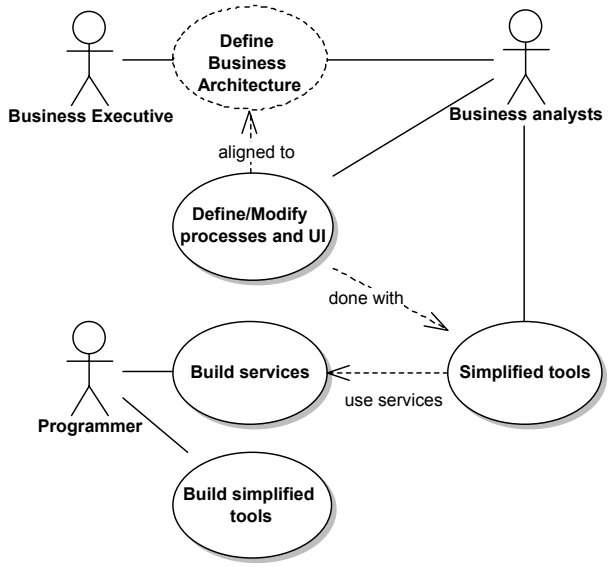


**Figure 5. Roles in a service oriented environment**

an abstract overview about the roles and the relevant use cases in a service oriented world. A similar approach has already been scrutinized in [2]. The business architecture is defined by the business executives and the business analysts. Because of the application of model driven design tools the business analysts can define and modify processes and user interfaces. The help of programmers will still be needed for special cases and exceptional behavior, but generally the business analysts can lower the workload of programmers. The tools used by the business analyst are developed by programmers who incorporate the services they have build into the tools. This approach greatly enhances the speed of application development. It however, implies that business analysts are available and have the knowledge about the relevant domain. Therefore a major task in adopting SOA within an enterprise is the hiring of business analysts with adequate know-how or the training of existing employees.

For the programmers within an enterprise it is important to get acquainted to the new technologies required for a service orientation. The main technology areas of interest are the different XML-based standards in use and the internal architecture necessary for service orientation. Especially when setting up a service orientation on top of existing systems, a wrapping of old business logic is necessary. I.e. the mainframe system used for accounting could be wrapped with a web service in order to make its functionality available to subsidiaries. Building such wrappers requires either the implementation directly in the legacy system - in this case on the mainframe - or writing an intermediate layer performing the conversion from a system call to an XML message. In both cases profound knowledge of the legacy system is required. However, such legacy systems are often in use for 20 years or longer and the programmers often have already retired. One major task in making legacy systems available in a service oriented world will be the hiring of qualified programmers who can implement the necessary interfaces.

In a roadmap to a service oriented environment the human resources factor plays a very important role. Without qualified employees a transition to SOA will most likely fail or at least be inefficient.

## 3. Conclusion and Outlook

The current strong trend towards service orientation is unlikely to change in the next few years. Even more than service orientation supporting techniques such as model driven architectures and model driven development will evolve from their draft status into fully qualified software development tools. With the availability of such tools the proliferation of service oriented concepts will increase significantly. Moreover, with the availability of service oriented commercial of the shelve software (COTS) also small and medium sized businesses can participate on a service oriented level. The augmentation of service orientation together with supporting tools will help composite applications to be as widespread as client/server applications are nowadays. Therefore it is more crucial than ever for an enterprise to stay up to date and not to miss the service orientation train. Whether there is business conducted between two arbitrary companies or not will very much rely on the

fact, if the other partner supports a service oriented environment. It has already been shown how many different sections of an enterprise are affected by the introduction of a service oriented environment. Moreover, it is most likely easier to name those departments not affected by the introduction than those affected. Hence the design and implementation must be thoroughly planned and organized.

On top of the organization the overall SOA strategy is placed, guiding the enterprises' efforts towards a functioning service oriented architecture. In order to fulfill the requirements and goals set by the strategy, a SOA roadmap is needed. In the long term only those enterprises will be successful which have a well developed and adapted SOA roadmap.

This work was supported by the Austrian Federal Ministry of Economics and Labour and the IST Sixth Framework Programme of the European Union.

## References

[1] BEA, IBM, Microsoft, SAP AG and Siebel Systems. *Business Process Execution Language for Web Services*, May 2003. Version 1.1.

[2] Dan Woods, Thomas Mattern. *Enterprise SOA, Designing IT for Business Innovation*. O'Reilly, 2006.

[3] Frank E. Redmond III. *Microsoft Distributed Component Object Model*. John Wiley and Sons Inc, 1997.

[4] B. Hofreiter and C. Huemer. Transforming UMM Business Collaboration Models to BPEL. In *Proceedings of OTM Workshops 2004*, volume 3292, pages 507–519. Springer LNCS, 2004.

[5] Ivar Jorstad, Schahram Dustdar, Do Van Thanh. A service oriented architecture framework for collaborative services. In *Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE05)*, pages 121–125. IEEE, 2005.

[6] P. Liegl, R. Schuster, and M. Zapletal. A UML Profile and Add-In for UN/CEFACT's Modeling Methodology. Master's thesis, University of Vienna, February 2006.

[7] Massimo Pezzini. Gartner Group Applied SOA Presentation: Best Practices from the Best Practitioners.

[8] Michael E. Porter. *Competitive Advantage*. Simon and Schuster, January 2004.

[9] Michael J. Earl. *Information Management: The Organizational Dimension*. Clarendon Press, March 1996.

[10] OASIS. *UDDI Version 2.04 API Specification*, July 2002. UDDI Committee Specification.

[11] Object Management Group (OMG). *Common Object Request Broker Architecture*, July 2002. Version 3.0.

[12] Object Management Group (OMG). *Unified Modeling Language Specification*, April 2005. Version 1.4.2.

[13] I. Sun Microsystems. Assessing your SOA readiness. Technical report, June 2004.

[14] UN/CEFACT Techniques and Methodologies Group. *UN/CEFACT's Modeling Methodology (UMM), UMM Meta Model - Foundation Module*, Oct. 2006. Technical Specification.

[15] Ward and Peppard. *Strategic Planning for Information Systems*. Academic Internet Publishers Incorporated, October 2006.

[16] World Wide Web Consortium (W3C). *Web Services Description Language (WSDL)*, Mar. 2001. Version 1.1.

[17] World Wide Web Consortium (W3C). *Simple Object Access Protocol (SOAP)*, June 2003. Version 1.2.

[18] World Wide Web Consortium (W3C). *Web Services Choreography Description Language*, Nov. 2005. Version 1.0, http://www.w3.org/TR/ws-cdl-10/.

[19] W. H. J. Y. Yanbing Jiang, Chunxiao Xing. On procedure strategy of constructing SOA's modeling language. In *Proceedings of the 2005 IEEE International Workshop on Service-Oriented System Engineering (SOSE05)*, pages 71–76. IEEE, 2005.