BGR+07] *Rainer Berbner, Tobias Grollius, Nicolas Repp, Julian Eckert, Oliver Heckmann, Erich Ortner, Ralf Steinmetz;* **Management of Service-oriented Architecture (SoA)-based Application Systems;** Enterprise Modelling and Information Systems Architectures, Mai 2007, S. 14-26.

Rainer Berbner, Tobias Grollius, Nicolas Repp, Julian Eckert, Oliver Heckmann, Erich Ortner, Ralf Steinmetz

Department of Computer Science, Technische Universität Darmstadt, Merckstr. 25, Darmstadt, Germany
berbner | repp | eckert | heckmann | steinmetz@kom.tu-darmstadt.de

Department of Business Information Systems, Technische Universität Darmstadt, Hochschulstr. 1, Darmstadt, Germany
grollius | ortner@winf.tu-darmstadt.de

Management of Service-oriented Architecture (SoA)-based Application Systems

Flexible business processes are a key success factor for enterprises to succeed in globalized markets. The Service-oriented Architecture (SoA) concept is very well suited to support flexible business processes and their underlying application systems because services can be composed in an efficient way to achieve a high level of agility. However, the management of SoA-founded application systems is often neglected. Thus, in this paper we conclude and extend our previous work on the enhancement of the generic SoA-based concept by additional management functionality, e.g. monitoring mechanisms and SLA management. As a validation of our concept we introduce WSQoSX, a prototypical implementation based on Web Services.

1 Introduction

In a globalized world, companies of nearly any size are confronted with a continuously changing environment. A crucial competitive factor is the ability to react quickly, flexibly, and efficiently to changes of the environment by adapting the business strategy to new conditions, e.g. [HaCh03].

If the business strategy needs to be flexible and quickly adaptable, the same requirement applies to the business processes derived from that business strategy. Furthermore, as IT applications support or execute parts of a company's business processes or even the processes as a whole, the flexibility of the processes strongly depends on the flexibility of the underlying applications and IT architecture [Oest95].

Today, companies mostly use commercial off-the-shelf software (COTS) to support or execute parts of their business processes. But instead of adapting this standard software to existing business processes, the processes are aligned to those pre-configured by the selected COTS software [Oehl05]. Even simple changes of business processes demand a tremendous customizing effort.

An approach to meet this challenge is the "Two-level-programming" paradigm [LeRo00] aiming at flexible, agile, and reliable application systems realized by a clear separation of control and business logic. Following this paradigm, changes of processes can be made with almost no effects on the business logic and vice versa. Additional flexibility of business processes can be achieved by using reference models [FeLo04]. Thus, processes do not have to be modeled from scratch, but can be derived from domain specific models. An even more effective way of process modeling is the reusability of so-called process components that can be orchestrated to business processes. However, a necessary requirement is the availability of a sufficient number of process components to realize individual processes [Lang97]. Related to the business logic additional flexibility can be achieved by following the component-based application development approach, e.g. [SzGM02; Ortn05]. The basic idea behind this concept is the composition and orchestration of reusable software components to complex applications. As a major advantage of this approach, process components designed at the process layer can be mapped onto software components. The assembly of components can be made at design or at runtime. The selection and assembly of components at runtime leads to loosely coupled application systems that provide a high degree of flexibility and agility [GrOr05].

Nowadays, the Service-oriented Architecture (SoA) paradigm is recommended as an emerging architectural blueprint that enables flexible support of business processes by orchestrating services to business processes. In the rest of the paper this approach is called SoA-based application system. However, the management of SoA-based application systems is often neglected. Thus, we propose the enhancement of a SoA by dedicated management functionalities to discover, select, assemble, and execute appropriate components for orchestrating application systems. As a proof-of-concept of our approach we present Web Service Quality of Service Architectural Extension (WSQoSX) as a prototypical implementation based on Web Service technology.

The remainder of this paper is organized as follows: In Section 2 we introduce basic technologies to improve the flexibility of application systems. The requirements on the management of SoA-based application systems are discussed in Section 3. In Section 4 we describe the implementation of an application management system based on Web Services. The paper closes with a summary and outlook on future research issues.

2 Architecture and Basic Technologies of SoA-based Application Systems

In this section, we introduce the core concepts needed to realize flexible and agile application systems as outlined above.

2.1 Flexibility of Business Processes

Flexible business processes should be modifiable in short time and with maintainable effort. These modifications are changes of the process sequence, the insertion or omission of sub processes and activities of the business process. Applying the principles proposed by a component-based software development approach to business processes promise to offer the desired flexibility and efficient process modeling. Process components are assembled to business processes. They are reusable in different contexts and offer e.g. best-practice domain-specific solutions. For further details we refer to similar approaches presented by [Aals99; MaCH03; AAD+04].

Starting from a company's business strategy, process components are identified while performing functional business process decomposition as part of an in-depth business process analysis. Figure 1 shows the result of the decomposition of a generic credit process. The credit process was decomposed into the sub processes *loan request*, *credit assessment*, *servicing* and *workout*. In our example, the sub process *credit assessment* is decomposed again. Results are the sub processes *internal rating*, *external rating* and *decision*, which itself can be decomposed. A further step of decomposition of the sub process *internal rating* leads to the activities *evaluate documents*, *evaluate employment* and *evaluate income* [BeHS05].

As shown in Figure 1 every single activity, sub process and process itself can be a process component, depending on the degree of granularity and flexibility needed and required by the business. The decision on the degree of granularity is a trade-off between reusability of the single components in different contexts and the effort to configure the process of process components.

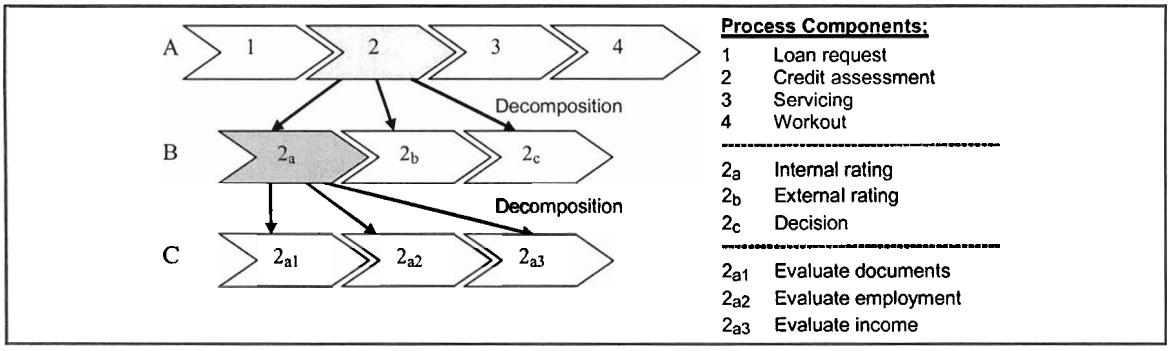


Figure 1: Decomposition of a generic credit process at the process layer [BeHS05]

2.2 Flexibility of SoA-Based Application Systems

In the previous section, we introduced how to decompose business processes using process components. If activities, sub processes or even the whole process should be computer-supported or -executed, the corresponding IT application has to be as flexible as the process itself. Applications built of loosely-coupled software components have the potential to offer this flexibility, e.g. [Ortn05].

According to [ABC+02], a component consists of different (software) artifacts. It is reusable, self-contained and marketable, provides services through well-defined interfaces, hides its implementation, and is designed to be deployed in different potential configurations. Implementing flexible process-oriented applications, every process component is assigned to a specialized component that supports or executes the tasks of the process component. A Service-oriented Architecture enables the coaction of these services.

A SoA is a specific software architecture based on services as fundamental elements for integrating and developing applications, e.g. [Barr03; Papa03; KrBS05]. They are well-defined, self-contained, and encapsulate high-level business functionality. Services can be implemented as specific software components. Services communicate with each other by sending and receiving messages and can adopt different roles. A SoA is characterized by the loosely coupling of the services involved, which is essential for flexible applications. When acting as a service provider a service publishes its interfaces that can be invoked by other services that play the role of a service requestor. Services can be replaced by other services at runtime. A SoA supports location transparency which denotes that services should have their definitions and location information stored in a repository and are accessible by a variety of clients that could locate and invoke the services independent of their location [Papa03]. To enable agile applications and processes a SoA aims at reducing complexity. This is achieved by the loosely coupling of the involved services and the decoupling of the technologies used at provider and requestor sides [KrBS05]. As a consequence, a SoA shifts the focus from technical to business requirements. From a business perspective, SoA-based application systems have the potential to achieve significant cost saving due to the reduction of maintenance costs and the fact that a SoA can be seen as an enabling framework for Business Process Outsourcing (BPO) [BeMS04; StBM04; BHMS05].

A modification of an existing business process will primarily affect the order and scope of service invocations and does not automatically result in changes of the underlying IT architecture. In order to increase flexibility, services are not connected to each other directly but using a Workflow Management System (WfMS) as an orchestration engine following the principle of a strict distinction between control and execution of business processes.

If a sub process of a business process is supported or executed by a service, the WfMS invokes the previously determined service via its interface. The WfMS provides services with required input parameters and - if applicable - receives the results of the execution in order to transfer them to following organizational resources. There are three main advantages of this approach: First, existing business processes can be rapidly adapted to a changing environment.

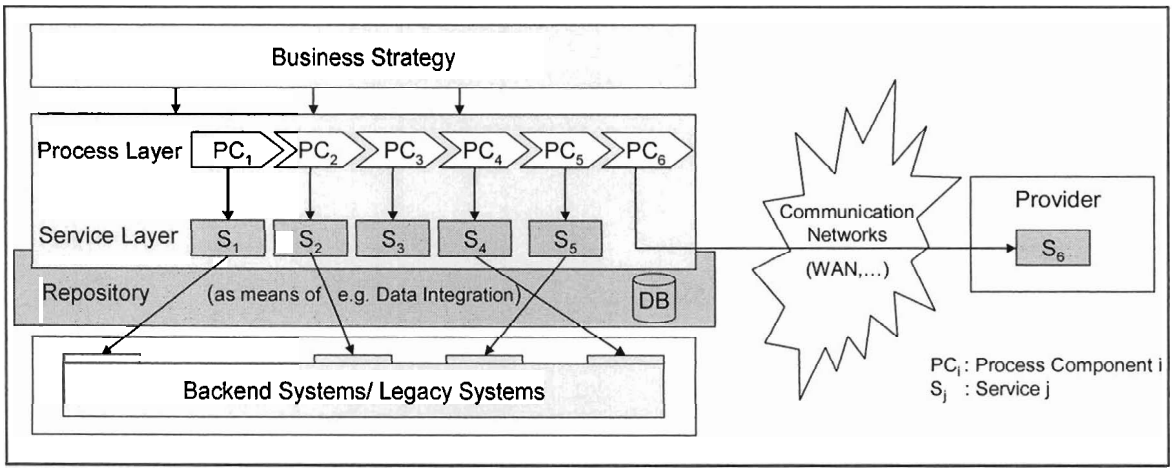


Figure 2: Service-based Business Process Outsourcing

This can be done by modifying the process sequence and recombining existing and novel process components and the executing services administered by one basic system. Secondly, it is possible to integrate legacy systems by encapsulating them with a service interface allowing a transparent usage of their functionality. Thirdly, using loosely coupled services it is possible to obtain services not only from internal service providers but from external service vendors. This supports a type of BPO called Service-based BPO which is depicted in Figure 2.

Our approach supports the reusability of process components as well as services in different business processes, which leads to further cost reductions, improved reliability and a faster development of component-based and IT-supported business processes [Nati05]. Business processes can be plugged together out of process components using a construction kit [MCL+99]. Repositories [Ortn99] can be used to manage and store those process components, information about services and mappings between them.

For the means of data integration we propose data services managing the data flow between services and backend systems. In particular, data services encapsulate and manage complex data structures. Furthermore, data services enable manipulating data in a consistent way and also provide data access to database systems. Data services can be enriched with additional functionality that checks the validity of the required data.

3 Management of SoA-based Application Systems

In the previous section the SoA concept as an architectural blueprint for enabling flexible processes and their underlying application systems is presented. However, to establish dynamic and robust application systems in real-world scenarios, application management is crucial. Thus, in this section we present requirements on a comprehensive management system for SoA-based applications.

Therefore, we have identified several components offering dedicated management functionality. Those management components as a whole form an application management system (Figure 3). Depending on the current state of the application system lifecycle, the components are separated into two groups: construction and execution components.

3.1 Discovery Component

As an initial step of the construction process of an application system, adequate services supporting the business processes have to be located by the *Discovery Component*. The most suitable ones are selected to be integrated in the application system in the subsequent step performed by the *Selection Component*.

Assuming the business process is already decomposed into process components, the *Discovery Component* performs a search for appropriate services that offer the

functionality required by the process components. To generate highly flexible application systems, a mapping between process components and services has to be created for every process component on the lowest level of the process hierarchy. But from a business perspective it also makes sense to search for services on a higher level of the process hierarchy in order to find services supporting wider functionality.

If the search is not successful as one or more services are not available or not known to the local repository, the *Discovery Component* has to stop the construction process and inform the application system designer about the missing components, which can also be supported from external partners. In case no component could be obtained the business process has to be adapted or the missing components have to be implemented.

A precondition of discovery is the detailed specification of services to search for using a specification framework. The same specification framework can be used for the internal reference specification as well as for the service specification used by external component catalogs. Based on the reference specification, services are searched for in both internal (e.g. an enterprise repository) and external service catalogs (e.g. a service marketplace).

The comparison of the reference specification with service specifications may result in difficulties, as the characteristics can be specified by the service requestors and providers in different ways. To overcome those difficulties, the specification of all characteristics should be standardized. For this purpose a normative expert language could be used for the specification of the functionality of both process components and services.

Alternatively, semantic web technologies [LaSw99] or ontologies [Fens03] could be applied. In ontologies concepts are correlated. Thus, the search for appropriate services not only includes those, whose specification matches exactly the reference specification, but also services that are specified using synonymous or similar concepts.

3.2 Selection Component

The *Selection Component* decides which specific service is assigned to an abstract process component. For that purpose, it has to assess and compare the characteristics of the services being delivered by the *Discovery Component*.

In addition to the compliance with functional requirements, the *Selection Component* also considers the non-functional attributes. Besides costs the Quality of Service (QoS) attributes are subsumed as non-functional attributes. QoS attributes (e.g. [MaNa02; Ran03]) can be divided into runtime related QoS criteria, e.g. scalability, capacity, and performance (e.g. response time, latency, throughput), transaction support related criteria (e.g. atomicity, consistency, integrity, and durability), and security related criteria (e.g. authentication, authorization and data encryption). An additional QoS criterion which is referred to as reputation measures the average ranking given by end users, though only the end users perspective is considered. Compliance as a further QoS related criterion measures how often the service provider has delivered the offered service levels accurately. Thus, compliance represents the ability of a service provider to maintain the service level of each QoS parameter laid out in a SLA [KaKL03].

The *Selection Component* can also be triggered at runtime to switch to another service when the runtime behavior differs from the one estimated at planning time [PaKL03].

The technique applied by the *Selection Component* can be based on local optimization strategies as well as on global ones, e.g. [ZBN+04]. Local optimization selects services that are high-ranked within a specific category (e.g. performing a customer rating) whereas global optimization considers optimization criteria and constraints with regard to the workflows as a whole. When dealing with multiple QoS categories like cost, reputation or maximal execution time, we have to aggregate the QoS for each category and for each pattern which leads to a multidimensional optimization problem [JaMG05]. For a more detailed discussion of those approaches we refer to our previous work in [BHRS06].

3.3 Rating Component

The *Rating Component* calculates a ranking based on a given SLA and runtime behavior of former executions of a specific service. Considering compliance as mentioned in Section 3.2 the *Rating Component* has to build up a rating of the services and service providers, which represents the degree to which extent the requirements have been fulfilled in the past. With this rating the Service Selection Component itself can learn

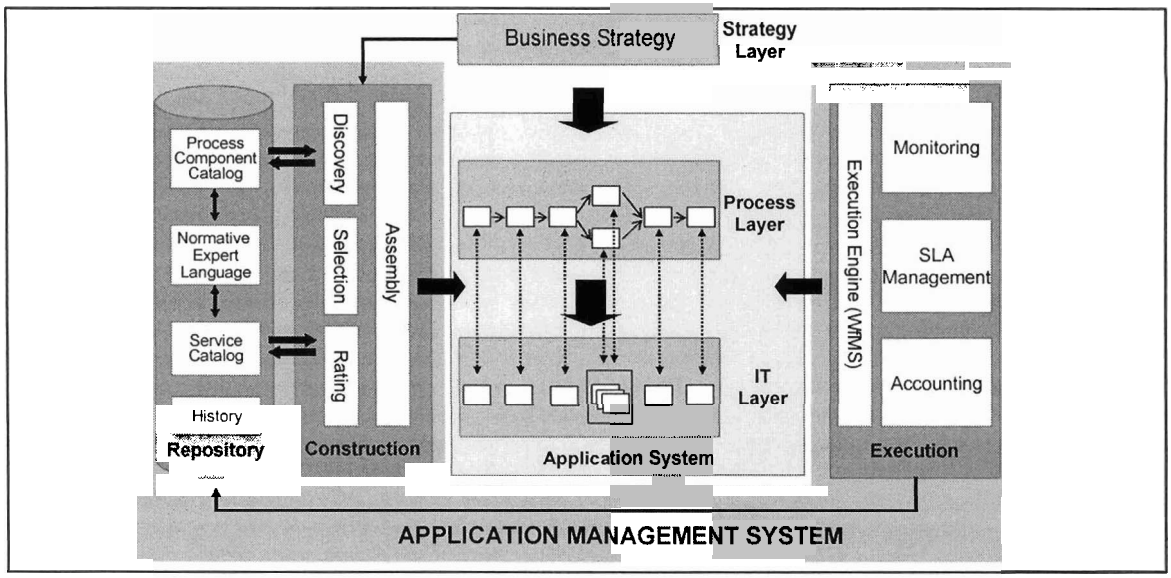


Figure 3: A management system for SoA-based applications and processes [BGR+05]

from the behavior of the past and is able to choose the best service in the future.

In a SLA measurable as well as non-measurable attributes like reputation and security are defined. Both types of criteria have to be evaluated by the *Rating Component*, although non-measurable values have to be assessed by IT experts before. For this assessment we propose an evaluation matrix [BHMS05]. After the assessment by the IT expert and the evaluation of measurable values, a score of the according service is calculated, which again is used to create a ranking of all potential services [BHMS05]. This ranking can be used as a basis for a subsequent dynamic selection of the particular services. In addition to the ranking created by the *Rating Component*, an IT expert is able to define constraints concerning further QoS requirements (e.g. "response time always smaller than 10ms").

To realize flexibility, if various users (e.g. paying users or guests) want to have access to different services, a kind of prioritization scheme is essential. The realization that the paying user has to be served first can be done by implementing a prioritization scheme. To achieve a controlled throughput coming along with a guarantee that several requests will be served and avoid that some requests will get lost, it has to be ensured that for every *m* requests of a higher priority also *n* requests of a lower priority will be processed [ShAS03].

3.4 Assembly Component

After determining the appropriate internal and external services, their integration in an application system has to be prepared. Based on the services chosen by the *Selection Component*, the *Assembly Component* manages the construction of the execution plan. Furthermore, the *Assembly Component* provides wrapping services to overcome any kind of heterogeneity, e.g. by converting data types of input or output message elements.

3.5 Execution Component

After constructing the application system, the main task of an application management system at runtime is to control the execution of the business process and to invoke the participating services in the correct sequence, at the right time and providing them with the required input - if necessary - via wrapping services. This task could be assigned to a WfMS behaving as an *Execution Component* that is capable to manage not only software services but also other organizational resources (employees, knowledge, etc.) [JaBu96].

3.6 Accounting Component

Accounting is the process of tracing application systems activities to a responsible source [Atis01] usually conducted by the service provider as a foundation for

charging and billing. Accounting activities aim at keeping track of which requests and responses have been sent to or received from partner services. The cost of the used service, which is dependent on the demands of QoS, is an essential criterion when deciding the pricing of a web service.

The *Accounting Component* enables the realization of different billing strategies. To introduce several pricing mechanisms the *Accounting Component* should support the following approaches: In traditional Application Service Provider (ASP) scenarios the pay-as-you-go model is used for pricing of services, where the customer has to pay a subscription fee plus a transaction based charge [MaSV04]. In general, we can identify some basic charging models as the flat rate model, where the user has unlimited access to the service and the pay-per-use model, where the user has to pay a fee for each invocation of the service. Besides these models there are the one-off payment model, where the user can use the service for its lifetime after a certain payment and the lease-based model, where the user can invoke the service after a certain payment any number of times within a given time period [SMS+03].

Three pricing policies can be distinguished: commoditized services, channelized services and customized services [MaSV04]. The pricing models for these categories named by the authors are a subscription-based pricing model, a transaction-based pricing model and a risk-based pricing model.

3.7 Monitoring Component

At execution time the QoS parameters defined by the SLA are monitored by the *Monitoring Component*. The component analyzes the data collected by the *Accounting Component* and compares them to the guaranteed characteristics defined in the SLA. In case of deviations between SLA and measured data the provider of the particular service as well as the service requestor are notified. Furthermore, not only notifications to service provider and requestor are sent in case of non SLA compliant Web Services, but bad-performing services can be automatically substituted by other services with the same functionality sending a message to the *Selection Component*.

To monitor the QoS attributes a customizable SLA monitoring engine can be used [SMS+02]. While monitoring the QoS attributes during the service execution a replanning process can be triggered, if there is a high likelihood for a SLA violation or if the deviation between the actual and the estimated QoS is very high.

The replanning mechanism is able to trigger the *Selection Component* selecting different services to satisfy the QoS requirements. In crucial situations the tradeoff between a possible improvement by using a replanning mechanism and the replanning overhead has to be balanced [CPEV05]. In [BHRS06] we propose an extremely fast performing heuristic based replanning mechanism, which ensures that the execution of a Web Service workflow remains feasible, valid and optimal.

3.8 SLA Management Component

SLAs are bilateral contracts between a customer and a service provider used to guarantee a special QoS for a service, especially the levels of availability and performance [KeLu02]. Besides, a SLA can contain pricing, contractual and other information. SLAs are the result of a negotiation between a customer and a service provider. In RFC 3198 a conformed SLAs definition to our approach can be found [WSS+01].

In our approach, SLAs are handled by the *SLA Management Component*. This component parses a given SLA and extracts the relevant information about guarantees to provide the *Monitoring Component* with.

3.9 Deconstruction Component

The *Deconstruction Component* manages the controlled removal of services. The *Deconstruction Component* identifies the effective dependencies between the service to be removed and other services and evaluates the consequences of removing the service for active workflow instances and the workflow schema [CrLa00].

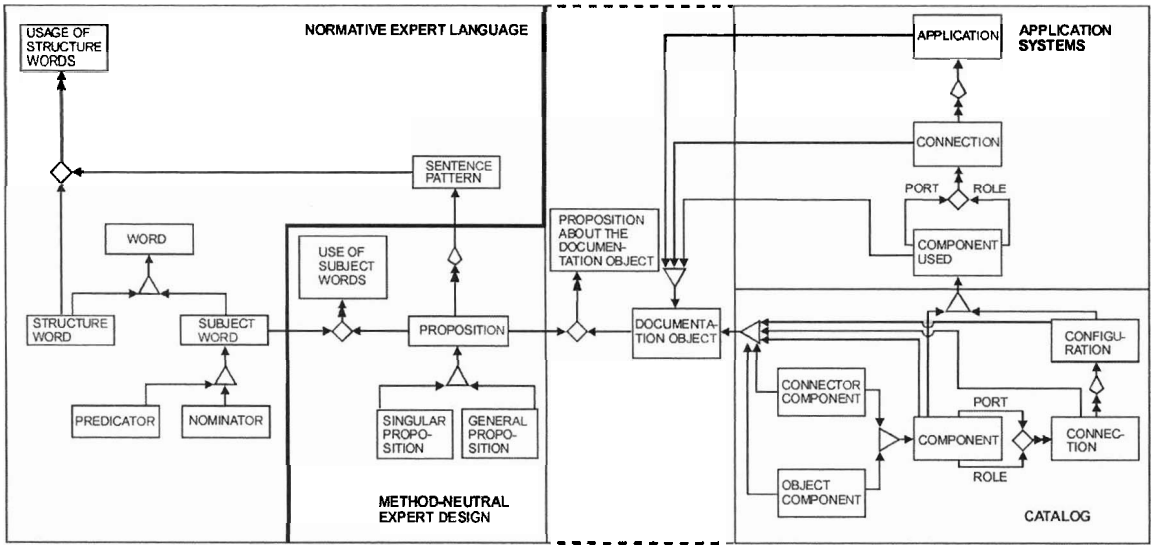


Figure 4: Meta schema (documentation structure) of a universal repository [Ortn05]

3.10 Enterprise Repository System as Integration Means

Repositories are documentation systems. On meta-language-level artifacts, e.g. data structures, software components, or process components are described in a structured way. The data structure for the documentation of these language artifacts is called meta schema or documentation structure of a repository system.

As an essential element of an application management system the repository includes catalogs for process and software components as well as the descriptions of other organizational resources [GrOr05]. Figure 4 shows a part of the meta schema of an enterprise repository, including the documentation structure of the software component catalog and the structure for documenting the usage of the software components in application systems. An example of a sophisticated specification framework for software components is presented in [Over04].

The application-specific (semantical) integration of the components of different categories and the component-based application systems can be managed using a normative expert language that is also documented in the repository [Ortn99].

4 WSQoSX: A Prototypical Implementation of an Application Management System

In this section we introduce the prototypical implementation of a SoA-based application management system presented in Section 3 using Web Service technology.

4.1 Web Service Technology

Web Services can be regarded as a SoA-enabled technology. A Web Service is defined as a self-contained, modular business application that has open, Internet-oriented, standard-based interfaces [BCC+04]. The Web Service core standards are formed by Web Service Definition Language (WSDL), Simple Object Access Protocol (SOAP) and Universal Description, Discovery and Integration (UDDI). WSDL is used as interface definition language, SOAP as communication protocol and UDDI as a repository to publish and search for particular Web Services, e.g. [ACKM04].

QoS related to Web Services refers to the non-functional properties of a Web Service, e.g. [MaNa02]. Enterprises are not willing to rely on Web Services hosted by external service providers, if there is no guarantee about their QoS needs. However, the management of QoS attributes is often neglected. Thus, we introduce our prototypical implementation based on

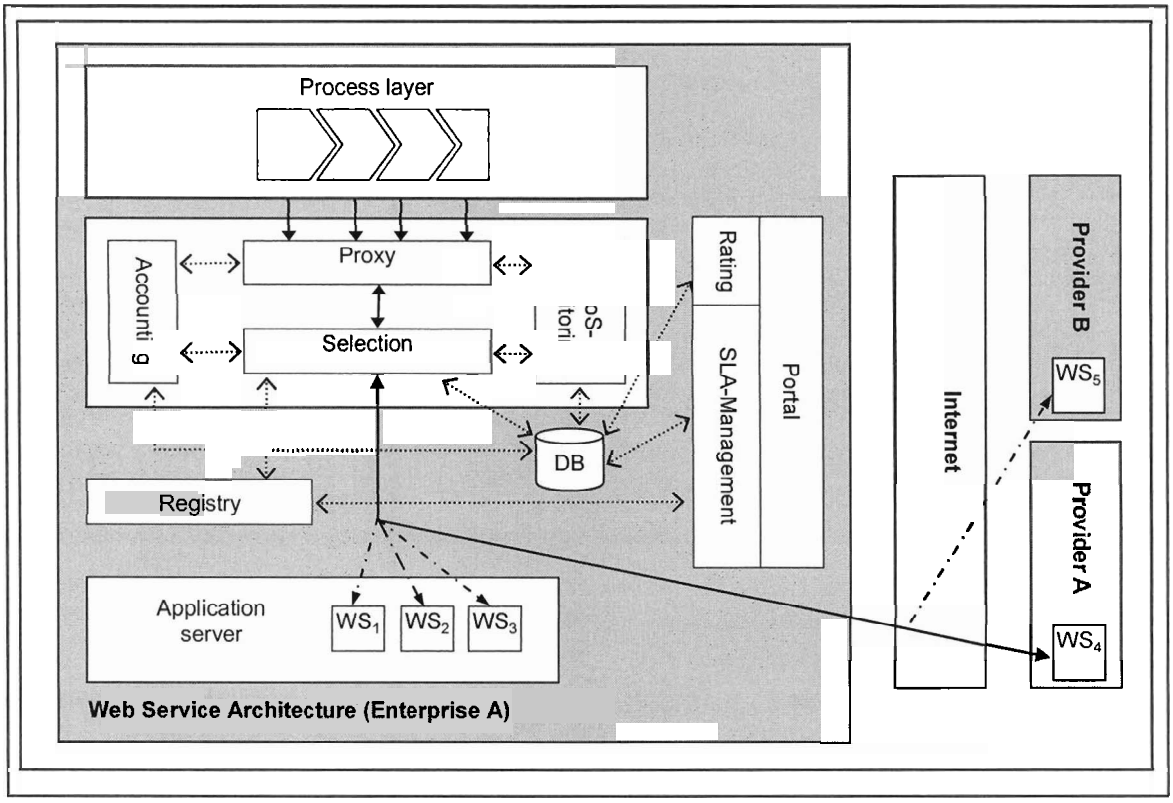


Figure 5: Web Service Portal of WSQoSX [BeHS05]

Web Service technology that offers a comprehensive QoS-aware management support.

4.2 Implementation

As a proof-of-concept, a prototype of an application management system supporting QoS in a Web Services based SoA was developed at Technische Universität Darmstadt. The application management system called Web Service Quality of Service architectural Extension (WSQoSX) can be seen as a partial implementation of the concepts presented in Section 3. The implementation uses the Business Process Execution Language for Web Services (BPEL4WS) to execute processes but is not restricted to BPEL4WS. To realize those concepts and to achieve the required QoS, different components were designed and implemented.

- *Portal-based Web Service Registration*: The provider of a service has to register at the web front end (portal) of the application management system in order to announce information about his business as demonstrated in Figure 5. After the registration the provider is allowed to publish his Web Services according to pre-defined categories (e.g. credit rating). Furthermore, a corresponding SLA has to be referenced as well, which defines the core QoS parameters of the offered services. At the time a potential user of the service agrees to this offer a contract between provider and user is established.
- *SLA Management*: A SLA Management Component is responsible for the evaluation of submitted SLAs. The component extracts information about Web Services, e.g. provider name, name of the Web Service and guaranteed QoS attributes, prepares it for further processing and stores the information inside the repository of the application management system. SLAs are modeled

based on IBM's Web Service Level Agreement (WSLA) framework [KeLu02; LKD+03].

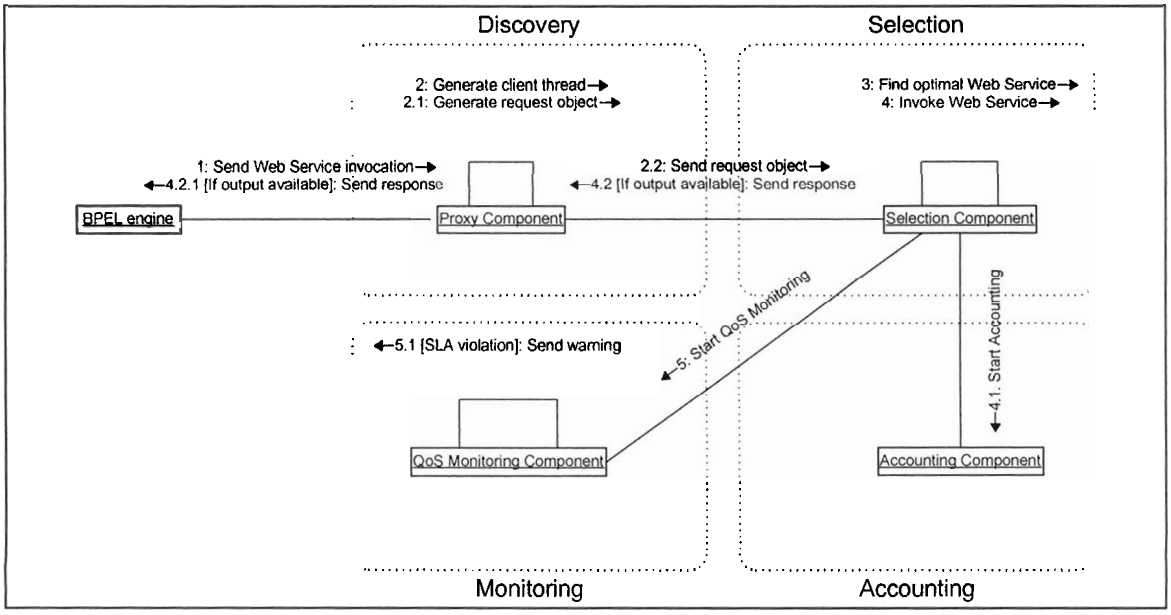
- *Dynamic Web Service Selection*: The dynamic selection of Web Services is conducted by the *Selection Component*. It invokes Web Services after comparison of the services QoS attributes stored in the repository by the *Rating Component*. In our prototype invocations of Web Services are created by a BPEL4WS engine executing a given process description. Furthermore, the accounting mechanism is started as well while invoking the service through the *Selection Component*. The *Accounting Component* tracks every event needed for settlement like start and end time of Web Service usage (e.g. in case of time-based accounting) as well as potentially occurring errors.

- *Dynamic replacement of bad-performing or erroneous Web Services at runtime*: Based on information gathered by the *QoS Monitoring Component* at runtime, the system is able to substitute Web Services not fulfilling the SLAs requirements during the execution of a process. This is realized by sending a message to the *Selection Component* to terminate the bad-performing or erroneous Web Service and to start another Web Service with the same functionality as far as such a service is known to the application management system.

The BPEL4WS engine does not invoke a Web Service (e.g. a credit rating Web Service) directly. Web Service invocation is managed by a *Proxy Component* instead. To facilitate this kind of indirection, the *SLA Management Component* automatically substitutes the physical address of the Web Service for the address of the *Proxy Component* at the time of service registration at the portal. Acting as a server waiting for Web Service invocations in form of SOAP messages encapsulated in HTTP requests, the *Proxy Component* works as a dispatcher in this architecture. Whenever a Web Service invocation is received on a dedicated port of the server, it starts a client thread, which itself is responsible to process the actual invocation made by the BPEL4WS engine and to route back the results of the invocation. The interaction of the participating architectural components is depicted in Figure 6.

After receiving a HTTP request containing the Web Service URL and the SOAP message sent by the Business Process Execution Language (BPEL) engine (Step 1 in Figure 6), the *Proxy Component* creates both a client thread (Step 2) and a request object based on the data received (Step 2.1). The client thread of the *Proxy Component* will be closed after the execution of the Web Service.

Figure 6: Interaction of the different components



The request object is passed to the *Selection Component* by the client thread of the *Proxy Component* (Step 2.2). The *Selection Component* is responsible for the retrieval of a suitable Web Service using the results calculated by the *Rating Component*. Based on these calculations the *Selection Component* chooses the best suitable Web Service and is now able to invoke the Web Service (Step 4). For accounting purposes, the *Accounting Component* is started (Step 4.1). Every possible output of the Web Service will be sent by the *Selection Component* as a response to the client thread of the *Proxy Component*, which itself will forward the output to the BPEL4WS engine (Step 4.2.1).

During the execution of the Web Service, the *QoS Monitoring Component* checks for violations of the given SLA (Step 5). In case of a violation, a warning is created in order to inform the provider of the particular Web Service (Step 5.1).

5 Related Work

Research in the area of SoA and SoA-based technologies (e.g. Web Services) has been mainly focused on certain issues, e.g. SLA management [JiMS02; SaDM02; DaLP03] or Quality of Service (QoS) aspects, e.g. [GoKG03; KaKL03; Ran03]. However, the comprehensive realization of flexible and reliable application systems by dedicated management support has been either not addressed or there are no prototypical implementations that could proof the feasibility of the proposed concepts, e.g. [EsTo04].

The Web Service Offerings Language (WSOL) discussed in [TPP+03] supports the management of web services as well as the management of web service compositions. The Web Service Offerings Infrastructure (WSOI), which enables monitoring of WSOL-enabled Web Services and dynamic manipulation of their classes of service, has been developed to demonstrate the usefulness of WSOL [TMPE04]. So this work comes close to our own. Nevertheless, from a business perspective it is more beneficial to use the de facto standard BPEL instead of designing new languages and specifications.

In [Mc03] BPEL is extended with capabilities for performance measurements (e.g. logging and auditing). However, this work discusses only one of the facets of Web Service management. It does not address e.g. SLA and policy management in BPEL processes.

6 Summary and Outlook

In this paper we presented an approach for managing SoA-based application systems. For this purpose, we extended the basic SoA concept with additional management functionality like monitoring mechanisms and SLA management. As a proof-of-concept we introduced WSQoSX, a prototypical implementation of our approach.

Our further research activities aim at extending the concepts and architectural approaches introduced in this paper as well as completing the development of WSQoSX. Furthermore, we are researching on a simulation environment for the evaluation of the underlying concepts. Especially, we focus on QoS-aware selection mechanisms for Web Services based on optimization problems.

Acknowledgement

We thank our colleagues of the E-Finance Lab for the constructive discussions. A special thank goes to Michael Spahn who has supported us implementing WSQoSX.

The work on this paper is partly sponsored by the E-Finance Lab (http://www.efinancelab.com).

References

[AAD+04] Acker, H.; Atkinson, C.; Dadam, P.; Rinderle, S.; Reichert, M.: Aspekte der komponenten-orientierten Entwicklung adaptiver prozess-orientierter Unternehmenssoftware. In: 1st Joint Conference on Architectures, Components, and Applications (AKA 2004). Gesellschaft für Informatik, Bonn 2004.

[Aals99] van der Aalst, W. M. P.: Flexible Workflow Management Systems: An Approach Based on Generic Process Models. In: Springer-Verlag, Berlin, Heidelberg 1999, pp. 186-195.

[ABC+02] Ackermann, J.; Brinkop, F.; Conrad, S.; Fettke, P.; Frick, A.; Glistau, E.; Jaeckel, H.; Kotlar, O.; Loos, P.; Mrech, H.; Ortner, E.; Raape, U.; Overhage, S.; Sahm, S.; Schmietendorf, A.; Teschke, T.: Standardized Specification of Business Components. Gesellschaft für Informatik, Bonn 2002.

[ACKM04] Alonso, G.; Casati, F.; Kuno, H.; Machiraju, V.: Web Services. Concepts, Architectures and Applications. Springer-Verlag, Berlin, Heidelberg 2004.

[Atis01] ATIS, C.: Accountability. http://www.atis.org/tg2k/_accountability.html, 2001, (19.08.2005).

[Barr03] Barry, D. K.: Web Services and Service-Oriented Architecture: The Savvy Manager's Guide. Morgan Kaufmann Publishers,San Francisco, USA 2003.

[BCC+04] Bellwood, T.; Capell, S.; Clement, L.; Colgrave, J.; Dovey, M. J.; Feygin, D.; Hately, A.; Kochman, R.; Macias, P.; Novotny, M.; Paolucci, M.; Riegen, C. v.; Rogers, T.; Sycara, K.; Wenzel, P.; Wu, Z.: UDDI Version 3.0.2. http://uddi.org/pubs/uddi_v3.htm, 2004, (13.05.2005).

[BeHS05] Berbner, R.; Heckmann, O.; Steinmetz, R.: An Architecture for a QoS driven composition of Web Service based Workflows. In: Networking and Electronic Commerce Research Conference (NAEC 2005). Riva Del Garda, Italy 2005.

[BeMS04] Berbner, R.; Mauthe, A.; Steinmetz, R.: Unterstützung dynamischer E-Finance-Geschäftsprozesse. In: Konferenz Elektronische Geschäftsprozesse 2004 (EGP 2004). Syssec Verlag, Klagenfurt, Austria 2004, pp. 44-54.

[BGR+05] Berbner, R.; Grollius, T.; Repp, N.; Heckmann, O.; Ortner, E.; Steinmetz, R.: An approach for the Management of Service-oriented Architecture (SoA) based Application Systems. In: Enterprise Modelling and Information Systems Architectures (EMISA 2005). Klagenfurt, Austria 2005.

[BHMS05] Berbner, R.; Heckmann, O.; Mauthe, A.; Steinmetz, R.: Eine Dienstgüte unterstützende Web Service-Architektur für flexible Geschäftsprozesse. In: Wirtschaftsinformatik 47 (2005) 4, pp. 268-277.

[BHRS06] Berbner, R.; Heckmann, O.; Repp, N.; Spahn, M.: An Approach for Replanning of Web Service Workflows. In: Forthcoming in: Americas Conference on Information Systems (AMCIS). Acapulco, Mexico 2006.

[CPEV05] Canfora, G.; Penta, M. D.; Esposito, R.; Villani, M. L.: QoS-Aware Replanning of Composite Web Services. In: IEEE International Conference on Web Services (ICWS 2005). IEEE, Orlando, USA 2005, pp. 121-129.

[CrLa00] Crnkovic, I.; Larsson, M.: Component Configuration Management. In: 5th Workshop on Component Oriented Programming. Blekinge, Sweden 2000.

[DaLP03] Dan, A.; Ludwig, H.; Pacifici, G.: Web service differentiation with service level agreements. http://www-106.ibm.com/developerworks/webservices/library/ws-slafram/, 2003, (14.05.2005).

[EsTo04] Esfandiari, B.; Tosic, V.: Requirements for Web Service Composition Management. In: 11th HP OpenView University Association Workshop (HP-OVUA 2004), Paris, France 2004.

[FeLo04] Fettke, P.; Loos, P.: Referenzmodellierungsforschung. In: Wirtschaftsinformatik 46 (2004) 5, pp. 331-340.

[Fens03] Fensel, D.: Ontologies - A Silver Bullet for Knowledge Management and Electronic Commerce. Springer-Verlag, Berlin, Heidelberg 2003.

[GoKG03] Gouscos, D.; Kalikakis, M.; Georgiadis, P.: An Approach to Modeling Web Service QoS and Provision Price. In: 4th International Conference on Web Information Systems Engineering Workshops (WISEW'03). IEEE, Rome, Italy 2003, pp. 121-130.

[GrOr05] Grollius, T.; Ortner, E.: A Concept of Configuring Flexible Application Systems for Business and Administration Processes. In: 4th International Conference on Information Systems Technology and its Applications (ISTA'05). Gesellschaft für Informatik, Bonn 2005, pp. 196-199.

[HaCh03] Hammer, M.; Champy, J.: Rengineering the Corporation: A Manifesto for Business Revolution. HarperBusiness, New York, USA 2003.

[JaBu96] Jablonski, S.; Bussler, C.: Workflow Management. International Thomson Computer Press, London 1996.

[JaMG05] Jaeger, M. C.; Mühl, G.; Golze, S.: QoS-Aware Composition of Web Services: An Evaluation of Selection Algorithms In: OTM Conferences. 2005, pp. 646-661.

[JiMS02] Jin, L.-j.; Machiraju, V.; Sahai, A.: Analysis on Service Level Agreement of Web Services. HP Laboratories, Palo Alto, USA 2002.

[KaKL03] Kalepu, S.; Krishnaswamy, S.; Loke, S. W.: Verity: A QoS Metric for Selecting Web Services and Providers. In: 4th International Conference on Web Information Systems Engineering Workshops (WISEW'03). IEEE, Rome, Italy 2003, pp. 131-139.

[KeLu02] Keller, A.; Ludwig, H.: The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. IBM Research Report, 2002.

[KrBS05] Krafzig, D.; Banke, K.; Slama, D.: Enterprise SOA. Service-Oriented Architecture. Best Practices. Prentice Hall, Upper Saddle River, USA 2005.

[Lang97] Lang, K.: Gestaltung von Geschäftsprozessen mit Referenzprozessbausteinen. Gabler Verlag, Wiesbaden 1997.

[LaSw99] Lassila, O.; Swick, R. R.: Resource Description Framework (RDF) Model and Syntax Specification. http://www.w3.org/TR/PR-rdf-syntax/, 1999, (26.08.2005).

[LeRo00] Leymann, F.; Roller, D.: Production Workflow, Concepts and Techniques. Prentice Hall, Upper Saddle River, USA 2000.

[LKD+03] Ludwig, H.; Keller, A.; Dan, A.; King, R. P.; Franck, R.: Web Service Level Agreement (WSLA) Language Specification. http://www.research.ibm.com/wsla/WSLASpec V1-20030128.pdf, 2003, (2006-05-10).

[MaCH03] Malone, T. W.; Crowstone, K.; Herman, G. A.: Organizing Business Knowledge. The MIT Press, Cambridge, USA 2003.

[MaNa02] Mani, A.; Nagarajan, A.: Understanding quality of service for Web services. http://www-106.ibm.com/developerworks/webservices/library/ws-quality.html, 2002, (15.04.2005).

[MaSV04] Mathew, G. E.; Shields, J.; Verma, V.: QoS Based Pricing for Web Services. In: (Hrsg.): WISE 2004 Workshops. LNCS 3307. Springer-Verlag, Berlin, Heidelberg 2004, pp. 264-275.

[Mc03] McGregor, C.: A Method to extend BPEL4WS to enable Business Performance Measurement. . School of Computing & Information Technology, University of Western Sydney, Sydney, Australia 2003.

[MCL+99] Malone, T. W.; Crowston, K.; Lee, J.; Pentland, B. T.; Dellarocas, C.; Wyner, G. M.; uimby, J.; Bernstein, A.; Herman, G. A.; Klein, M.; Osborn, C. S.; O'Donnell, E.: Tools for Inventing Organizations: Toward a Handbook of Organizational Processes. In: Management Science 45 (1999) 3, pp. 425-443.

[Nati05] Natis, Y. V.: Applied SOA: Conquering IT Complexity Through Software Architecture. Gartner Research, Stamford, USA 2005.

[Oehl05] Oehler, K.: Business Engineering bei der Einführung betriebswirtschaftlicher Standardsoftware - Auswirkungen einer serviceorientierten Architektur. In: HMD-Praxis der Wirtschaftsinformatik 241 (2005) pp. 35-44.

[Oest95] Österle, H.: Business Engineering: Prozess- und Systementwicklung. Springer-Verlag, Berlin, Heidelberg 1995.

[Ortn05] Ortner, E.: Component-Based Application Architecture for Enterprise Information Systems. In: Data Management in a Connected World 2005. Springer-Verlag, Berlin, Heidelberg 2005, pp. 181-200.

[Ortn99] Ortner, E.: Repository Systems. In: Informatik-Spektrum 22 (1999) 4, pp. 235-251.

[Over04] Overhage, S.: UnSCom: A Standardized Framework for the Specification of Software Components. In: Weske, M., Liggesmeyer, P. (Hrsg.): 5th Annual International Conference on Object-Oriented and Internet-Based Technologies, Concepts, and Applications for a Networked World, NODe 2004. Springer-Verlag, Berlin, Heidelberg 2004, pp. 169-184.

[PaKL03] Padowitz, A.; Krishnaswami, S.; Loke, S. W.: Towards Efficient Selection of Web Services. In: 2nd International Joint Conferences on Autonomous Agents and Multi-agent Systems. New York, USA 2003.

[Papa03] Papazoglou, M. P.: Service -Oriented Computing: Concepts, Characteristics and Directions. In: 4th International Conference on Web Information Systems Engineering (WISE'03). IEEE, Rome, Italy 2003, pp. 3-12.

[Ran03] Ran, S.: A Model for Web Services Discovery with QoS. In: ACM SIGecom Exchanges 4 (2003) 1, pp. 1-10.

[SaDM02] Sahai, A.; Durante, A.; Machiraju, V.: Towards Automated SLA Management for Web Services. http://www.hpl.hp.com/techreports/2001/HPL-2001-310R1.pdf, 2002, (14.05.2003).

[ShAS03] Sharma, A.; Adarkar, H.; Sengupta, S.: Managing QoS through prioritization in Web services. In: 4th International Conference on Web Information Systems Engineering Workshops (WISE 2004). IEEE, Brisbane, Australia 2003, pp. 140-148.

[SMS+02] Sahai, A.; Machiraju, V.; Sayal, M.; Moorsel, A. P. A. v.; Casati, F.: Automated SLA Monitoring for Web Services. In: 13th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management: Management Technologies for E-Commerce and E-Business Applications. Springer-Verlag, Berlin, Heidelberg 2002, pp. 28-41.

[SMS+03] Silver, G.; Miller, J. A.; Sheth, A.; Myers, J.; Maduko, A.; Jafri, R.: Modeling and Simulation of Quality of Service for Composite Web Services. In: 7th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2003) Orlando, USA 2003.

[StBM04] Steinmetz, R.; Berbner, R.; Martinovic, I.: Web Services zur Unterstützung flexibler Geschäftsprozesse in der Finanzwirtschaft. In: (Hrsg.): Handbuch Industrialisierung der Finanzwirtschaft. Gabler-Verlag, Wiesbaden 2004, pp. 641-654

[SzGM02] Szyperski, C.; Gruntz, D.; Murer, S.: Component Software. Addison-Wesley, Upper Saddle River, USA 2002.

[TMPE04] Tosic, V.; Ma, W.; Pagurek, B.; Esfandiari, B.: Web Services Offerings Infrastructure (WSOI) – A Management Infrastructure for XML Web Services. In: IEEE/IFIP Network Operations and Management Symposium (NOMS 2004). Seoul, South Korea 2004, pp. 817-830.

[TPP+03] Tosic, V.; Pagurek, B.; Patel, K.; Esfandiari, B.; Ma, W.: Management Applications of the Web Service Offerings Language (WSOL). In: 15th International Conference on Advanced Information Systems Engineering (CAISE 2003). Velden, Austria 2003, pp. 468-484.

[WSS+01] Westerinen, A.; Schnizlein, J.; Strassner, J.; Scherling, M.; Quinn, B.; Herzog, S.; Huynh, A.; Carlson, M.; Perry, J.; Waldbusser, S.: Terminology for Policy-Based Management. Network Working Group, RFC 3198, 2001.

[ZBN+04] Zeng, L.; Benatallah, B.; Ngu, A.; Dumas, M.; Kalagnanam, J.; Chang, H.: QoS-aware Middleware for Web Service composition. In: IEEE Transactions on Software Engineering 30 (2004) 5, pp. 311-328.