On the Management Requirements of Web Service Compositions

Anis Charfi [1], Rainer Berbner [2], Mira Mezini [3], Ralf Steinmetz [2]

[1] SAP Research CEC Darmstadt
Darmstadt, Germany

[2] Multimedia Communication Lab, [3] Software Technology Group
Darmstadt University of Technology
Darmstadt, Germany

Abstract. Several works have addressed the management of individual Web Services. However, the specific management requirements of workflow-based web service compositions such as those specified in the BPEL have not yet been considered. In this paper, we present several management requirements in web service compositions such as discovery and selection management, SLA and policy management, middleware services management, and management of the composite service. Supporting these requirements is crucial for providing a reliable service composition with well-defined QoS properties. We also introduce web service composition management and present our vision of having dedicated tool support for it in future BPEL engines.

1 Introduction

Web services [1] that are provided by different parties can be composed to cross-organizational workflows and to value-added composite web services. Web service composition languages such as BPEL (Business Process Execution Language) [2] provide a cheap means for enterprise application integration and business-to-business integration. However, we notice that whilst web service based workflows cover the functional part of the composition (control flow, data flow, etc), the management of Quality of Service (QoS for short) properties in these compositions such as performance, availability, security, and reliability have not yet been addressed appropriately. On the other hand, research has revealed that the basic web service protocol stack is not sufficient to establish web services in real-world scenarios [4] and that considering QoS requirements is crucial for a sustainable success of web services [7] including their compositions. In fact, without any guarantee regarding QoS, no enterprise will be willing to rely on external web services within critical business processes.

In this paper, we look at several management requirements in the lifecycle of web service compositions, which are mostly not supported by current composition tools.

These requirements include discovery and selection management of partners, composition-side management of the QoS and middleware properties of the interactions with and within the composition, the management of the composite web service, the management and handling of the SLAs and policies of the services, and the management of business aspects.

We also define web service composition management (WSCM) as the management of the composite web service, the composition-side management of the composed services, and the management of the interactions that take place within and with the composition including their QoS properties. The definition of WSCM is not specific to BPEL but it works also with other composition languages. We merely assume the use of a workflow-based language to compose services that are described in terms of functionality and in terms of QoS. We will focus on BPEL because it is the standard for web service composition.

It is important to note that WSCM is different from web service management because most existing works on web service management operate at the interface level, i.e., on top of WSDL [19, 24, 30]. We look at the management requirements of web service based workflows from the implementation perspective, i.e., the perspective of the user who defines and deploys such workflows. Thus, web service composition management is a form of application level management, whereby the application is implemented in a workflow language such as BPEL.

The contribution of this paper is two-fold. First, it outlines management concerns that are crucial but mostly not supported in current web service composition tools. Second, it defines WSCM and explains how it differs from web service management in general. Our vision is that future orchestration engines should provide WSCM capabilities. How the WSCM requirements can be implemented is out of scope.

The remainder of this paper is organized as follows. Section 2 gives some background knowledge. Section 3 outlines the management requirements in web service based workflows and defines web service composition management. In Section 4, we report on related work. Section 5 concludes the paper.

2 Background

In this section, we provide some background knowledge that is relevant for understanding web service composition management.

2.1 Web Service Management

In Web Service Distributed Management (WSDM) OASIS defined a specification called Management of Web Services (MOWS) [24], which defines an additional

management interface of a web service. The management interface provides information about the identity, metrics, operational state, request processing state, etc.

In [2] web Service management is defined as an extension of enterprise application management. Following this definition, web service management has two sides: management of applications within an enterprise (*internal management*) and management of relationships with other web services across enterprises (*external management*). This distinction aims at the management *of* web services.

2.2 Web Service Composition and BPEL

Web service composition provides a means to create a value-added web service by combining existing web services. It spans two important areas: the specification by means of a composition language and the execution by means of an appropriate runtime. The specification of a web service composition consists in specifying a set of interactions between the composition and the composed web services as well as the control flow and data flow around these interactions.

The Business Process Execution Language (BPEL) is a process-oriented web service composition language, in which a composite web service is implemented using a workflow process. BPEL is widely accepted by research and industry and it was recently accepted as an OASIS standard [3]. The new standard is to a large extent based on BPEL 1.1 [2]. BPEL processes are deployed on specific workflow engines, which orchestrate the invocations of the partner web services according to the process specification. The main concepts in BPEL are partners, variables, and activities.

The partners are the parties that the composite web service interacts with such as clients and other web services. A *partnerlink* is a typed connector between two WSDL port types. It specifies two roles: one role is played by the composition and the other is played by the partner [2]. The variables act as containers for the data that is exchanged between the partners as well as for the process data.

The activities are the units of work in the process. BPEL differentiates *primitive* activities and *structured* activities. Primitive activities are atomic whereas structured activities are composite. The core of a BPEL process is the set of atomic messaging activities (e.g., receive, reply, invoke), which perform interactions between the partners. The *assign* activity is used to modify the content of a variable. Structured activities such as *sequence* and *flow* contain other activities, structuring the latter according to control flow patterns such as sequential execution and concurrency.

2.3 Service Level Agreements and Policies

Service Level Agreements (SLAs) and policies are the most used means to describe the non-functional properties of web services.

SLAs are widely established to guarantee Quality of Service (QoS) between Internet Service Providers (ISP) on the network layer. Service Level Agreements (SLAs) are bilateral contracts and defined in RFC 3198 [32] as the documented result of a negotiation between a customer and a provider of a service that specifies the levels of availability, serviceability, performance, operation or other attributes of the service. A SLA contains a Service Level Specification (SLS). A SLS is a set of parameters (e.g., availability, performance, and error rate) and their values which together define the service offered to the customer. Besides the SLS, an SLA can contain pricing information, contractual information, etc. To model SLAs, we use IBM's Web Service Level Agreement (WSLA) framework [19]. WSLA is based on XML Schema and it is divided in three parts. In the section *Parties*, the organizations involved are described. Relevant parameters and the way how they are calculated are illustrated in the section *Service Descriptions*. In the section *Obligations*, Service Level Objectives (SLOs) are used to define criteria that have to be met by the provider.

Other QoS properties such as reliable messaging, security, and transactions are not supported by SLAs but they rather by policy based languages. WS-Policy [10] is a general model and XML-based syntax that can be used to express the requirements, capabilities, and preferences of web services e.g., with respect to reliability (as in WS-Reliability [27]) or security (as in WS-Security [6]). A policy is a collection of assertions. There are several domain-specific assertion languages, e.g., WS-SecurityPolicy [19] defines security assertions for integrity, confidentiality, etc.

3 Web Service Composition Management

We assume that we are building a composite service using a workflow-based web service composition language by orchestrating existing web services that are described not only in terms of their functionality (as supported by WSDL) but also in terms of QoS (as supported by SLAs and policies). Further, we assume that we have some requirements on the composite web service in terms of the QoS properties that it will guarantee to its clients.

3.1 Requirements for Web Service Composition Management

We grouped several management requirements that arise when creating and deploying web service compositions into the following categories:

3.1.1 Discovery and selection management

Service composition is generally used to solve a complex problem or implement a complex business process. The complex task of the composite web service is divided into smaller tasks that can be performed by existing services.

Discovery Management is about finding appropriate web services to build the composite web service. This activity takes place primarily at design time but can also take place at runtime. Depending on the type of service to be used, discovery is done in various ways. For instance in the case of business-to-business integration the partner organizations tell each other about the services they expose. In the case of enterprise application integration the system administrator of the enterprise knows about the services that wrap a certain application. In other cases, partner web services are discovered by searching internal and external UDDI registries. A prerequisite for an appropriate discovery is a sufficient description of partner web services.

At first, the discovery management component of the web service composition tool has to consider the functionality (business match) and find web services that match the port type of each partner (e.g., make an offer for CPUs). The business match is based only on the syntax and it can be improved by using some semantic web services technology such as OWL-S [28]. Besides the functional match, the non-functional properties of a web service are another important criterion in web service discovery. For describing the non-functional QoS criteria of web services, SLAs and policies are the mostly used means. Such QoS descriptions allow the selection of a particular web service to be driven by QoS concerns.

When creating a new web service it is often the case that there are certain QoS requirements on that new service. For instance, the creator of the composite service may require a response time of 2 ms to be guaranteed. Inferring QoS criteria for the individual services to be composed in terms of their SLAs and policies is a complex task that requires tool support. Another example is that the creator of a composite web service may have some transaction requirements that need to be translated to requirements on the individual web services. In the case of BPEL, the programmer may specify that a certain *sequence* with nested *invoke* activities has to be transactional (e.g., using a deployment descriptor [12] or policies that are attached to the process [13]). In such a case, appropriate tool support is required to restrict the selection of partners to web services with support for WS-AtomicTransaction.

In other scenarios, the discovery and selection of partners may pose certain non-functional requirements on the composite service. For instance, assume that we build a service in an intra-organizational setting, which provides one operation that checks for product availability and places an order if the required product is available. If the applications wrapped up by the availability service and the order service require authentication it is then necessary to make the composite service require authentication as well to have the authentication data that will be passed to each composed service.

In addition to the static specifications of QoS properties, the history, i.e., the runtime behavior, of a web service is sometimes necessary for the selection. For instance, if the SLA of some service specifies an availability of 80% then selection management has to gather information about each call, i.e., it has to record the run-time behavior

of web services to decide which web service should be invoked. Therefore, the history of web services executions should be stored in a database. Selection based on SLAs, policies, and history can be combined.

3.1.2 Management of the composite web service

In several composition languages including BPEL, the composition is exposed as a new web service, which needs to be managed like any other web service. Typical management concerns in web services are lifecycle management, the startup and the shut down of the service, the number of instances, the ability of the service to provide information about itself, its identity, its current load, the number of messages it is currently processing, its dependencies on other services, error handling, forwarding of errors to some third party, etc. Since the composite web service is implemented using a workflow process (e.g., a BPEL process) we need to understand what does each management concern means in terms of constructs of the workflow language, e.g., what is the relationship of service instances to process instances what is the number of messages the composite service is processing at a certain point in time.

As the lifecycle is rather implicit in BPEL, when we deploy a process we start the composite service but the process might have not started yet. The service can be shutdown by undeploying the process. Moreover, the current load of the composite service can be inferred from the number of process instances[1].

Fault handling is another important management issue in composite web services, e.g., if an error occurs during the execution of an operation of the composite service, it is important to identify the source of that error (whether it is a process error, an error in one of the composed services, an error in the client messages, a network error, etc). Some kind of process debugger (i.e., a tool that shows the execution state of each process instance and the values of each variable) would be very helpful to identify the source of the fault so that user can fix it and then redeploy the process.

To establish a reliable composition several QoS properties of the composite web service have to be addressed such as:

- **Availability**: Availability of a composite web service means the probability that all web services involved in the composition are available when invoked by the workflow process. A web service is considered available if it is able to respond to a request within a defined time interval.
- **Performance**: It can be measured by the throughput and the response time. Throughput means the number of requests that can be processed during a defined time slot. In the case of BPEL processes throughput depends not only on the number of process instances that can run simultaneously but also on the number of

[1] Some BPEL engines already provide this information in their management module.

messages that one process instance can consume[2] and on whether BPEL-specific concepts such as message correlation are used. In fact, the same process instance with correlation may be able to process multiple client requests (e.g., one for logging in, one for searching for a product, one for placing an order, etc).

The response time is the sum of transmission time and processing time and can be measured as the time required for processing a request. In BPEL the processing time is the time period from the point where a SOAP message with a matching a startable *receive*[3] arrives at the engine to the point where a response SOAP message matching the same operation is sent using a *reply* activity.

One major challenge in composition management is performance modeling and performance measurement of the composite service [14]. To analyze composite services and plan the workflow control, network calculus can be used to describe the worst-case performance behavior of a composite service. By addressing capacity planning, resource usage becomes more and more important. Performance modeling and measurement are crucial to ensure that the execution of the composite service remains feasible and SLA violations due to overload are avoided.

– **Error rate:** The error rate specifies the number of processing errors within a particular time interval. The error rate of the composite web service can be calculated based on the error rates of each partner web service whilst taking into account the number of interactions with each partner. When we define the error rate for a composite service that is implemented in BPEL, we have to differentiate errors that are generated by the process, errors that are thrown by partner services using fault messages, errors caused by faulty client messages, and errors due to network failures.

In addition, other QoS properties of the composite web service have to be managed such as security, reliable messaging, and transactions (cf. Section 3.1.4).

3.1.3 SLA and policy management

After the selection phase, we have a *required policy* (resp. SLA) for each partner (that was probably inferred from the non-functional requirements on the composite service) and a *published policy* (resp. SLA) for the selected service.

Moreover, the composite Web Service may have two different policies: one that is published to clients (server-side policy) and another that is used for interactions with the composed services (client-side). As the partner services may specify options in their policies, e.g., that they support either algorithm *a* or *b* for encryption, policy management should allow the process deployer to specify parameters that drive the decision on which option to choose. When such a client-side composition policy

[2] This can be inferred from the number of *receive* and *onMessage* activities in the process.
[3] A startable receive activity is an activity that leads to the creation of a new process instance.

exists an effective policy [10] has to be calculated using that policy and the published policy of the partner service.

Appropriate policy management should also check whether the published policy of a service has changed. In such a case, it should signal any mismatch to the user. Further, the policy management component is supposed to know about the middleware capabilities at the composition side, i.e., if a requirement of the partner after a policy update cannot be supported then policy management should throw an error.

SLA and Policy management is about handling all these SLAs and policies of the partner services and the composite service. Ideally, one would like to see a list of policies (required, published, effective) and SLAs (required, published) for each composed service and each interaction.

SLA Management should also monitor the composite service to check if the originally defined SLA is supported. It may turn out that this SLA should be modified after a certain period of time (e.g., because of the performance of a partner service that cannot be replaced, e.g., when that service is a wrapper around an internal application). Further, the SLA descriptions of the selected partner web services have to be monitored from the composition side by collecting execution data for each interaction with that service to check whether the SLA has been violated.

In addition, some means are needed to define how the composition runtime should behave in the case of SLA violation (e.g., notification of service provider and service consumer, sending an e-mail to an administrator, selecting an alternative service and in that case what selection strategy to follow, etc)

Based on SLAs, rankings for partner web services can be calculated for each service category (e.g., delivery web service) [7, 8, 9]. This ranking can be later used as a foundation for the dynamic selection of a particular web service. Furthermore, IT experts can define additional rules to exclude web services that do not satisfy certain minimal QoS requirements e.g., “Do not admit web services with a response time longer than 10 ms”.

3.1.4 Management of middleware concerns

There are several middleware requirements in web service compositions [12], which can be supported by WS-* based middleware services for security, transactions, reliable messaging, etc. Due to place limitations we focus only on security as a representative for the other middleware services.

Several security concerns arise in a composite web service such as the authentication of the composition in front of its partner services. That is, appropriate tool support is needed to specify the data (e.g., user name and password, binary keys) that should be passed to the security middleware before interacting with a partner. The security

middleware will then use that data to process the SOAP message according to the WS-* specifications for security such as WS-Security and WS-Trust.

There are also confidentiality and integrity requirements for the interactions with partners that can be enforced using a WS-Security based middleware service, which encrypts, signs, and adds user credentials to the messages of messaging activities. Appropriate tool support should allow the user to see the security properties of each interaction in the composition and also to pass the necessary parameters needed by the security service for enforcing that property. As there is a relationship between the security properties of an interaction and the security policies of the involved parties, the management of middleware concerns is related to policy management.

The composite web service could also have authentication requirements on its clients, i.e., it mandates incoming client requests to provide some claims; messages without appropriate user claims will be ignored. This can be the case when the client has to pay a fee for using the service. Authentication is then used to associate a contract (including a pricing model) with the client. There are further security issues such as trust, federation, secure conversations, and privacy that need to be managed and configured, e.g., if some of the partner services can be grouped into a trust domain, then the process would not have to authenticate itself separately in front of each partner (i.e., some kind of single sign-on can be introduced).

There are other middleware concerns [12] in composite web services such as persistence, transactions, and notification, etc. For each concern, tool support is needed to see the middleware properties of each interaction (i.e., messaging activity in BPEL) and also other process activities (e.g., a transactional *sequence* in BPEL) as well as the parameters passed to the middleware services to enforce these properties

3.1.5 Management of business aspects

Several business aspects have to be dealt with in Web Services such as enforcing legal contracts between the partners, accounting, billing, etc.

Accounting is the process of tracing information systems activities to a responsible source [5] usually conducted by the service provider as a foundation for charging and billing. In the context of web service composition, there are two forms of accounting (as being the provider of the composite web service, and as being the client of the partner web services). Logging and tracing are accounting activities with the purpose of keeping track of which requests and responses have been sent to or received from clients and partner web services including the respective data.

Billing is concerned with the bills that should be given to the clients of the composite service and also the bills between the composition and the composed services. The web service composition may have to pay a fee for using a partner web service based on different pricing models, i.e., pay-per-use or volume-based rates. At the composition side, the management module should collect the necessary data and statistics

about the usage of partner web services. This can be helpful to assign costs to internal business units according the cause of the costs. Additionally, the service requestor (i.e., the composition) can make use of accounting information to check the provider's invoice. Since the composition itself is a web service, which could charge clients a fee also according to various pricing models, the management module should correlate contracts and usage statistics to produce a bill to the client.

3.2 Definition of Web Service Composition Management

We define Web Service Composition Management (WSCM) as *the management of the composite Web Service, the composition-side management of the composed services, and the management of the interactions that take place within and with the composition including their QoS properties*. In a broader sense, it includes the supporting activities that are needed to provide a reliable Web Service composition with well-defined QoS properties such as:

- the discovery and selection of appropriate services to build the composition,
- the management of interactions with and within the composition in terms of SLAs, policies, middleware properties, and business aspects,
- the management of the composite web service

It is important to note that we look at the composite Web Service from the implementation perspective, i.e., the workflow process definition is available to us and not only the interface definition of the composite web service. This perspective is different from the one taken by general web service management approaches [19, 24, 30]. The latter assume only a WSDL interface and no knowledge about the internal implementation of the web service.

Web services and web service compositions can be managed from the technical perspective and also from the business perspective [15]. From the technical perspective, web service compositions are considered as distributed computing systems. From the business perspective, they represent business processes. Thus, WSCM is positioned between traditional systems and network management on the one side, and business process management on the other side [15].

When considering BPEL, the WSCM requirements mentioned so far can be supported by a WSCM module that will be hopefully part of future BPEL design-time and runtime tools as shown in Fig. 1. The nature of the WSCM requirements and their dependency on workflow-level details make supporting them necessarily a task of a component that is part of the orchestration engine because only the engine has knowledge about the workflow constructs and their execution state. This tool should show the different partners in the composition and allow the user to define criteria for their discovery and selection as well as criteria for selecting other services if some

QoS assurances are not met. The SLA and policy management view of the WSCM tool shows the SLAs and policies for each party that is involved in the composition and also the effective policy for each interaction. Further, it should provide information on the real QoS properties for each interaction via messaging activities in each process instance. The middleware concerns view shows the middleware properties of all interactions with clients and partners as well as the middleware properties of non-messaging BPEL activities such as *sequence* and *scope*. The business aspects view shows contracts and also accounting and billing information. The most important view of that component is definitely the one concerned with the management of the composite web service. It includes the policies and SLAs of that service, shows values for each QoS parameters such as availability and performance, and several server-side measurements for its SLA.

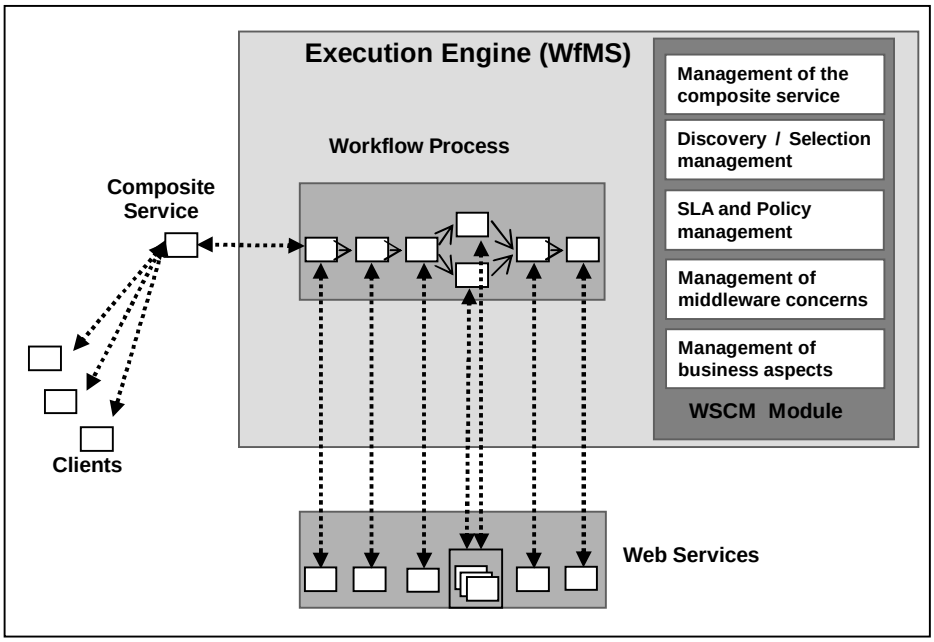

Figure 1. Architecture of a Service Composition Engine with a WSCM Module

4 Related Work

A lot of research has been done in the area of web service management (WSM) from the application management perspective [16], [21], [30]. OASIS proposes the "Web Services Distributed Management" specification that addresses the management of IT resources by defining web services interfaces (management through web services) [25], [26] and the management of web services by defining messages, events state

properties [24]. However, these specifications do not address the management of web services compositions at all.

In [1] Web Service Management is defined as an extension of enterprise application management, which can be seen as the task of monitoring and controlling applications in an enterprise so that they can be resilient to failures, configurable to changing needs of the business, accountable for billing and auditing, capable of performing under varying workloads, and secure to intended or unintended attacks [1]. Following this definition, Web Service Management has two sides: management of applications within an enterprise (internal management) and management of relationships with other web services across enterprises (external management). The external web service management is characterized by a limited visibility and control over portions of the application. In that work the management of service compositions from the composer's view is not discussed.

The *Web Service Offerings Language (WSOL)* discussed in [30] supports the management of web services as well as the management of web service compositions. So this work comes close to our own. However, we believe that it is more beneficial to use widely-accepted standards, such as BPEL, instead of designing new languages.

In [22], BPEL is extended with capabilities for performance measurements (e.g., logging and auditing). However, there is no complete support for the management requirements presented in our paper. Several other works such as [17], [18], [29] have considered QoS related non-functional properties but none of them took into consideration management issues of compositions of web services.

In [31], the authors present a Web Service Management Layer (WSML), which is placed in between the client application and the external web services to offer generic management functionality using aspects, e.g., billing, accounting, security and transactional support. Furthermore, the WSML proposed in [31] allows dynamically selecting and integrating web services at runtime based on rules and policies. However, there is no integration of this concept into a composition language and no focus on the management of the composition. A similar approach to the one adopted by WSML can be used together with AO4BPEL [11] to implement a WSCM layer.

The Web Service Agent Framework (WSAF) [22] achieves service selection taking into account the preferences of service consumers as well as the trustworthiness of providers. Policies are used by providers and consumers as a formal description of the offered or needed QoS. Due to possible discrepancies between the formally offered and the real QoS, service selection relying only on provider policies may lead to suboptimal service selections. To optimize service selection, the trustworthiness of provider policies has to be taken into account. Agents are used as service proxies to select services which propose the best fit between expressed offers and needs in consideration of the trustworthiness of policies. During execution agents monitor the QoS and calculate the deviation between the offered and the real QoS as a measure

for the trustworthiness of the policy, which influences further service selections. This work is also not concerns with the management requirements in service composition.

5 Summary

In this paper, we illustrated several management requirements in web service compositions and defined web service composition management. Our definition is not specific to one composition language and BPEL was taken as an example for illustration because it is the standard. We also explained why and how managing a composite web service is different from the general web service management. Moreover, we argued that state of the art BPEL engines are lacking support for composition management but hopefully future composition tools such as BPEL orchestration engines will provide support for the WSCM requirements discussed in this paper.

References

1. Alonso, G., Casati, F., Kuno, H., Machiraju, V.: Web Services. Concepts, Architectures and Applications. Springer-Verlag, Berlin (2003).
2. Andrews, T. et al: Business Process Execution Language for Web Services. Version 1.1. http://www.ibm.com/developerworks/library/ws-bpel. (2003).
3. Arkin A. et al: Web Services Business Process Execution Language 2.0, Public Review Draft, August 2006.
4. Alonso, G.: Myths around Web Services. Bulletin of the Technical Committee on Data Engineering. Vol. 25, No. 4. IEEE (2002) 3-9.
5. ATIS Committee: Accountability. http://www.atis.org/tg2k/_accountability.html (2001).
6. Atkinson, B. et al: Web Services Security (WS-Security). http://www.ibm.com/developerworks/library/ws-secure/ (2002).
7. Berbner, R., Heckmann, O., Steinmetz, R.: An Architecture for a QoS driven composition of Web Service based Workflows. In Proc. of Networking and Electronic Commerce Research Conference - NAEC 2005 (Lake Garda, Italy, Oct. 2005).
8. 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: Proc. of Enterprise Modeling and Information Systems Architectures (EMISA 2005), (Klagenfurt, Austria) 208-221.
9. Berbner, R.; Grollius, T.; Repp, N., et al.: Management of Service-oriented Architecture (SoA) based Application Systems. In: Enterprise Modelling and Information Systems Architectures - An International Journal, 2 (2007) 1, S. 14-25.
10. Box, D. et Al: Web Services Policy Framework (WS-Policy). http://www.ibm.com/developerworks/library/ws-policy (2004).
11. Charfi, A., Mezini, M.: AO4BPEL: An Aspect-Oriented Extension to BPEL. World Wide Web Journal, published online March 2007, Springer
12. Charfi, A, Schmeling B, Heizenreder A., Mezini M, Reliable, Secure and Transacted Web Service Composition with AO4BPEL. Proc. of ECOWS 06, pp. 23-34. IEEE.
13. Charfi, A. Khalaf R., Mukhi N., QoS-aware Web Service Compositions Using Non-Intrusive Policy Attachment to BPEL, Proc. of ICSOC 07, LNCS 4749, pp. 582-593, Springer, Vienna, Austria 2007,

14. Eckert, J ; Pandit, K ; Repp, N ; Berbner, R ; Steinmetz, R Worst-Case Performance Analysis of Web Service Workflows In: 9th International Conference on Information Integration and Web-based Application & Services (IIWAS); Jakarta, Indonesia (2007).
15. Esfandiari, B., Tosic, V.: Requirements for Web Service Composition Management. In Proc. of 11th HP-OVUA Workshop (Paris, France 2004).
16. Farrell, J.A., Kreger, H.: Web services management approaches. IBM SYSTEMS JOURNAL. Vol. 41, No 2. (2002) 212-227.
17. Gouscos, D., Kalikakis, M., Georgiadis, P.: An Approach to Modeling Web Service QoS and Provision Price. In Proc. of 4th International Conference on Web Information Systems Engineering Workshops - WISEW 2003. IEEE. (Rom, Italy 2003) 121-130.
18. Kalepu, S., Krishnaswamy, S., Loke, S.W.: Verity: A QoS Metric for Selecting Web Services and Providers. In Proc. of 4th International Conference on Web Information Systems Engineering Workshops - WISEW 2003. IEEE. (Rom, Italy 2003) 131-139.
19. Kaler, C., Nadalin A. (Eds.). Web Services Security Policy Language (WS-SecurityPolicy), Version 1.1, July 2005
20. Keller, A., Ludwig, H.: The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. Journal of Network and Systems Management, Vol. 11, No. 1. (2003), 57-81.
21. Machiraju, V., Sahai, A., van Moorsel, A., Web Services Management Network. 2002, hp labs. Technical Report HPL-2002-234 (2002) 1-17.
22. Maximilien, E. M., Singh, M. P., Toward autonomic web services trust and selection. Proc. of the 2nd international conference on Service oriented computing, ICSOC 04, (New York, NY, USA) 212-221.
23. McGregor, C.: A Method to extend BPEL4WS to enable Business Performance Measurement. Technical Report No. CIT/15/2003. University of Western Sydney (2003).
24. OASIS, Web Services Distributed Management: Management of Web Services (WSDM-MOWS 1.0). http://docs.oasis-open.org/wsdm/2004/12/wsdm-mows-1.0.pdf (2004).
25. OASIS, Web Services Distributed Management: Management Using Web Services (MUWS 1.0) Part 1. http://docs.oasis-open.org/wsdm/2004/12/wsdm-muws-part1-1.0.pdf
26. OASIS, Web Services Distributed Management: Management Using Web Services (MUWS 1.0) Part 2. http://docs.oasis-open.org/wsdm/2004/12/wsdm-muws-part2-1.0.pdf
27. OASIS Web Services Reliable Messaging (WSRM) TC, Web Services Reliability (WS-Reliability) version 1.1 (2004).
28. The OWL Services Coalition: OWL-S: Semantic Markup for Web Services. Technical White Paper. http://www.daml.org/services/owl-s/1.0/owl-s.html (2003).
29. Ran, S.: A model for Web Services discovery with QoS. ACM SIGecom Exchanges. Vol. 4, No. 1. (2003) 1-10.
30. Tosic, V., Pagurek, B., Patel, K., Esfandiari, B., Ma, W., Management Applications of the Web Service Offerings Language (WSOL). Proc. of 15th International Conference on Advanced Information Systems Engineering - CAiSE. (Velden, Austria 2003) 468-484.
31. Verheecke, B., Cibrán, M.A.: AOP for Dynamic Configuration and Management of Web Services. In: Proc. of International Conference on Web Services - ICWS-Europe 2003, (Erfurt, Germany 2003) 137–151.
32. 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, RFC 3198. http://rfc3198.x42.com (2001).