An Architecture for a QoS driven composition of Web Service based Workflows

Rainer Berbner, Oliver Heckmann, Ralf Steinmetz
Multimedia Communications Lab - KOM
Computer Science Department
Darmstadt University of Technology
Merckstrasse 25
64283 Darmstadt
Germany
[berbner|heckmann|steinmetz]@kom.tu-darmstadt.de

Abstract

In times of globalisation and internationalisation enterprises have to establish flexible business processes to react to dynamically changing markets and to satisfy sophisticated customers. IT architectures that support the integration of legacy systems as well as the collaboration with business partners are a key success factor for flexible business processes. Thus, we present an approach to support flexible business processes by the means of a Service-oriented Architecture (SoA). Web Services as a realization of a Service-oriented Architecture gain more and more in importance to implement business processes. We address Quality of Service (QoS) as crucial for a sustainable success of Web Service based workflows. Thus, we designed and implemented a Web Service architecture with comprehensive QoS support. The selection of a particular Web Service depends on its QoS properties that must be guaranteed in Service Level Agreements (SLAs). The compliance with such SLAs is monitored by a component of the architecture as well.

1. Introduction

IT architectures within enterprises and organizations are characterised by a large amount of different legacy systems, middleware platforms, programming languages, operating systems and communication mechanisms (e.g. [25]). This heterogeneity has led to high complexity that is barely manageable. Traditional EAI (Enterprise Application Integration) solutions have often failed to overcome this heterogeneity. A report by Forrester reveals that less than 35% of integration projects have been successful within time and budget [17].

Web Services as an emerging technology based on open standards have the potential to overcome such integration problems (e.g. [2], [20]). Legacy systems expose their functionality as Web Services. Web Services can be composed to implement business processes without any effects on the underlying legacy systems. As a result enterprises can react much faster to dynamically changing markets and sophisticated customers. Only companies with a high flexibility to adapt to new conditions will survive in the long-term [5]. Furthermore it is possible to integrate Web Services offered by external Web Service providers to establish cross-organisational business processes. Thus, enterprises can focus on their core competence.

Research in the context of Web Services has revealed that the usage of WSDL (Web Service Definition Language) [37], SOAP (Simple Object Access Protocol) [38] and UDDI (Universal Description, Discovery and Integration) [36] is not sufficient to establish Web Services in real-world scenarios (e.g. [1]). Considering Quality of Service (QoS) requirements is crucial for a sustainable success of Web Services ([28], [31], [35], [44]). Without any guarantee regarding QoS, no enterprise is willing to rely on external Web Services within critical business processes. Thus, we designed and implemented an integrated architecture for a QoS driven composition of Web Service-based workflows that is introduced in this paper. Our research (e.g. [6], [34]) takes place within the E-Finance Lab[1]; the E-Finance Lab is a an industry-academic partnership between Frankfurt and Darmstadt Universities and partners Accenture, Deutsche Bank, FinanzIT, IBM, Microsoft, Postbank, Siemens, T-Systems, DAB bank, IS Teledata and VR-NetWorld. Goal of the E-Finance Lab is to develop scientific yet managerial methods for rearranging the business processes of the financial service industry.

The rest of this paper is structured as follows: In Chapter 2, we discuss work related to our research and how our research is different. An approach how business processes can be mapped onto the underlying IT architecture by the means of a Service-oriented Architecture (SoA) is described in Chapter 3. This approach allows the integration of legacy systems as well as the usage of external services. In Chapter 4 we introduce our integrated Web Service architecture that provides a comprehensive QoS support. The implementation is shortly presented in Chapter 5. In Chapter 6 we summarise our work and give an outlook of our future research activities.

[1] http://www.efinancelab.de/

2. Related work

A Web Service is defined as *a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web Service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols* [39]. WSDL, SOAP and UDDI form the Web Service core standards. 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. [2]).

BPEL4WS (Business Process Execution Language for Web Services) was introduced by IBM, BEA and Microsoft to model business processes that consist of Web Services. The BPEL4WS specification [3] distinguishes between executable process models and abstract process descriptions. The latter specifies the message exchange between the involved business parties without revealing inner details. This considers the fact that enterprises do not want to let their partner know how they do business. The BPEL4WS specification allows the dynamical binding of Web Services at runtime. However, criteria that determine what Web Service should actually be invoked are neglected. QoS is not considered in this context.

The WSMF (Web Service Modelling Framework) aims at the overcoming of semantic and syntactical heterogeneity in the context of Web Service composition [11]. QoS related issues are not discussed within this framework.

In [19], [27], [30] Web Service based workflows are addressed, but without a deep discussion of QoS. In [7], [8], [32] the search of suitable Web Services and their QoS behaviour are discussed. The focus lies on the semantic description of Web Services by means of ontologies as an assumption to find proper Web Services. In this context the QoS behaviour is addressed as an import search criterion. [31] presents the prioritisation of incoming Web Service invocations as an approach to realise QoS for Web Services. QoS criteria for Web Services are introduced in [15], [28], [44]. However, the technical support by an architecture is not discussed deeply.

Service Level Agreements (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 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 [41]. SLAs are used for different types of services, e.g. pure network connectivity services with network service providers or e.g. Web Services with Web Service providers. A SLA contains a Service Level Specification (SLS). A SLS is a set of parameters and their values which together define the service offered to the customer. Besides the SLS a SLA can contain pricing, contractual and other information. In [9], [16] early approaches using SLAs in the context of Web Services are introduced.

Compared to this work we present an architectural approach and its prototypical implementation with a comprehensive QoS support:

- External Web Service providers have to subscribe at a portal to register their Web Services.
- QoS properties of the Web Services have to be guaranteed by the provider in form of SLAs.
- The selection of a particular Web Service is performed at runtime. Furthermore this selection depends on the QoS properties of the Web Service and rules defined by decision makers within the enterprise.
- The compliance of the Web Service execution with the corresponding SLA is supervised at runtime.

How QoS can be realised at the service provider or at the network layer is out of scope in this paper. For this, we refer to e.g. [13], [29].
Besides QoS there are further open research issues in the area of Web Services (e.g. security and transaction processing [2]) we do not address within this paper.

3. Decomposition of business processes by means of a Service-oriented Architecture (SoA)

In organisation theory a business process is defined as *a set of one or more interconnected activities which collectively realize a business objective or policy goal, normally within the context of an organisational structure defining functional roles and relationships* [42]. Business processes that are supported by information systems are labelled as workflows [18].

To achieve optimal support by the underlying IT architecture, we first decompose business processes conceptually into activities (so-called process patterns). For this, we collaborate with the business process owners in the organisation. The decomposition is continued until the process patterns have the optimal granularity. The optimal granularity is determined by the business context and the functional handling as well as the optimal support by the information systems.
Figure 1 demonstrates this approach by means of a generic credit process. The credit process is decomposed into the process patterns loan request, credit

assessment, servicing and workout. As an example, the process pattern credit assessment is decomposed again, namely into the process patterns internal and external rating and decision. If we have a closer look to the internal rating, this activity is decomposed into three process patterns that are responsible for the evaluation of the required documents as well as loan requestor's income and his employment.

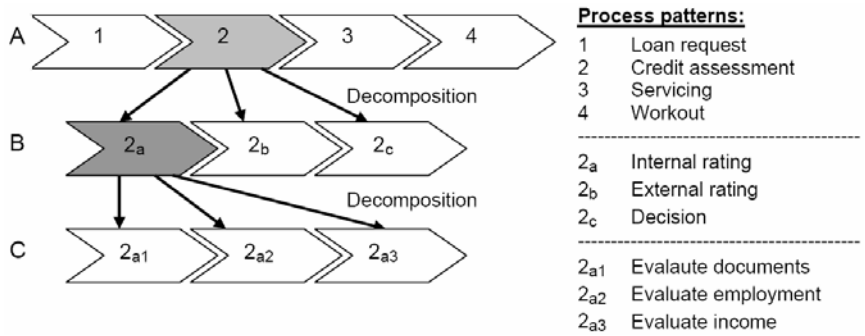

Figure 1: Decomposition of a generic credit process

The functionality of these process patterns is provided by so-called services. In this context a service is defined as a self-contained software component that provides a well-defined functionality [43]. An architectural approach based on services is called Service-oriented Architecture (SoA). A SoA is characterised by the loosely-coupling of the services involved. This means that services can be replaced by other services at runtime. Services communicate with each other by sending and receiving messages [26].

Figure 2 shows the mapping of conceptually modelled process patterns onto services. Those services encapsulate the functionality of legacy systems. The advantages of this approach are twofold: One the one hand the further operation of existing legacy systems is possible. On the other hand, despite of monolithic legacy applications, business processes can be modified by recombining existing and upcoming services in a flexible manner. As a consequence, a modification of business processes mainly affects the order of the different service invocations, but not the underlying legacy systems. Thus, our approach follows the paradigm of a *two-level programming*, which means that there is a clear distinction between the business process modelling and the implementation itself [18]. In this context the modelling of the workflow (the definition of the data and control flow), is called *programming in the large*, whereas the implementation of the abstract process patterns is named as *programming in the small*.

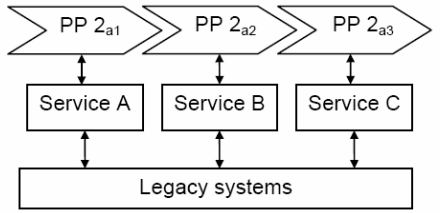

Figure 2: Mapping of process patterns onto services

Because services are loosely-coupled, this architectural approach allows the integration of services offered by external service providers. Compared to a traditional Business Process Outsourcing (BPO), not the complete process has to be outsourced, but only a few activities. We call this Service-based BPO. This approach allows a BPO with higher flexibility.

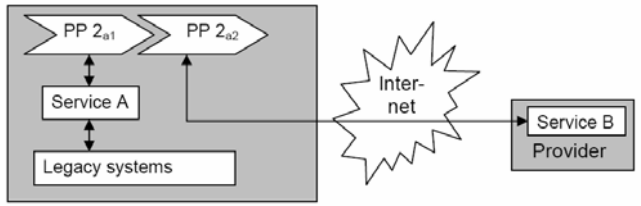

Figure 3: Service-based Business Process Outsourcing (BPO)

This architectural approach supports the reusability of process patterns in different business processes. Advantages of reusability are cost reduction, reliability and a faster development process [33]. For this, abstract process patterns and their implementations can be stored in repositories [24]. As a consequence business processes can be composed by combining process patterns out of a construction kit [22]. This approach becomes even more effective if these process patterns can be customised by parameters. This allows additional flexibility and adaptability of business processes. Furthermore this architectural approach brings together standardisation and individualisation, because the combination of standardised components allows the creation of individualised business processes.

4. A Web Service architecture with comprehensive QoS support

In Chapter 3 we demonstrated the strength of a Service-oriented Architecture (SoA) to integrate internal legacy systems as well as external services. Web Services as an implementation of the SoA concept gain in importance to establish business processes. However, there are a lot of concerns related to Web Services (e.g. [1]). A crucial issue is that quality of service is often neglected although it is obvious that enterprises are not willing to rely on external Web Services if there is no guarantee about their quality of service. In the future there will be a lot of Web Services that provide the same functionality. Thus, the non-functional QoS properties of a Web Service will become more important.

Thus, we present a Web Service architecture (Figure 4) with comprehensive QoS support. The decision, which particular Web Service among a group of Web Services with similar functionality is invoked, depends on its QoS properties. Furthermore our approach makes use of a portal and Service Level Agreements (SLAs). The compliance of the Web Service execution with these SLAs is monitored by the architecture as well.

Next we introduce the components of this architecture and their interaction.

An enterprise can publish invitations to bid for certain Web Services (e.g. Web Services for credit rating) at its web portal. Providers that offer suitable services have to subscribe to the portal and register their Web Services. Additionally, they have to reference a Service Level Agreement (SLA). These SLAs contain binding information related to QoS criteria. Our architecture supports the QoS criteria discussed in [12], [15], [21], [35]:

- Availability of a Web Service means the probability that a certain Web Service is available when invoked by a client. A Web Service is considered as available if it is able to respond to a request within a defined time interval.
- Performance as a generic term 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. The response time is the sum of transmission time and processing time and can be measured as the time required processing a request.
- The error rate specified the number of processing errors within a particular time interval.
- Security is a comprehensive criterion. Therefore it is assessed by the sub criteria authenticity, authorisation (of the participating partners) and data encryption.

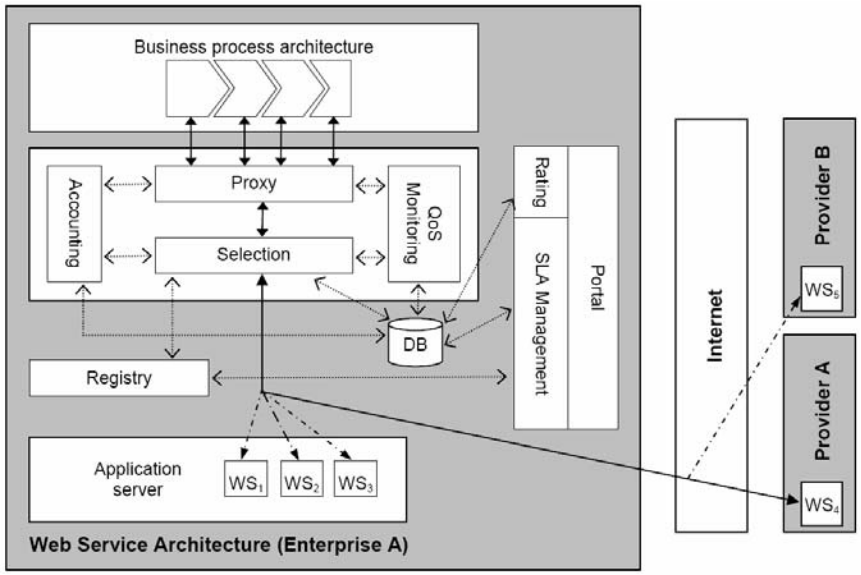

Figure 4: Architecture

- The reputation of a Web Service and its providers aims at the past experiences with a certain Web Service and its provider. This criterion also considers external references of the provider.

These criteria have to be guaranteed in a SLA. The SLAs are handled by the *SLA Management component*. After the registration of the SLA at the enterprise portal, the SLA is parsed by the *SLA Management component* and relevant information about guarantees is stored in a database. This information will be used by the *QoS Monitoring component* to detect potential SLA violations.

The evaluation and assessment of the QoS criteria proposed above is performed by the *Rating component.* The *Rating component* creates a ranking based on the SLAs. For this, the *Rating component* evaluates the criteria defined within the SLAs. The non-measurable values like reputation and security have to be assessed by IT experts. Furthermore IT experts can define rules to exclude Web Services that does not satisfy certain minimal QoS requirements. An example for such a rule can be: “Do not admit Web Services with a response time longer than 10 ms”. Web Services that pass this stage are integrated by the *Rating component* in the internal registry. This registry contains links to all Web

Services that comply with the defined constraints. These Web Services can be made available by internal business units or external service providers. Internal Web Services are favoured to encapsulate the functionality of legacy systems [20].

The *Proxy component* and the *Selection component* dynamically invoke Web Services considering QoS attributes of the registered Web Services. The *Proxy component* receives the Web Service invocations of the BPEL4WS process and forwards them to the *Selection component*. The *Selection component* chooses the Web Service with the highest score calculated by the *Rating component* on basis of the SLAs. Then the *Selection component* executes the particular Web Service. When a Web Service is invoked by the *Selection component*, the accounting mechanism is started as well. The *Accounting component* tracks start and end time of a Web Service invocation as well as potentially occurring errors. Accounting is *the process of tracing information systems activities to a responsible source* [4] and is usually conducted by the service provider as a foundation for charging and billing. However, the approach we propose in this paper enables accounting at the client side as well. This can be helpful to assign costs to internal business units according the cause of the costs. Additionally, the service requestor can make use of accounting information to check the provider's invoice.

The *QoS Monitoring component* controls the compliance of the Web Service execution with the SLA. Thus, it evaluates the measurements conducted by the *Accounting component.* The measurements have to be compared to the guaranteed metrics within the SLA. If there are any violations of the SLA, the provider of the particular Web Service as well as the service requestor are notified. Furthermore the *QoS Monitoring component* can initiate that a bad-performing Web Service whose availability break down, is substituted by another Web Service with the same functionality. For this, it sends a message to the *Selection component* to terminate the bad-performing Web Service and to start another one.

5. Implementation

The components of our Web Service architecture are implemented in Java. As application server we use Apache Tomcat 5.0. Apache Axis 1.1 is employed as SOAP engine and MySQL acts as a database server.

First of all, the Web Service provider has to register at the portal of the user (Figure 5). After the registration the provider is able to subscribe its Web Services (Figure 6) according to pre-defined categories (e.g. credit rating). A corresponding SLA hast to be referenced as well.

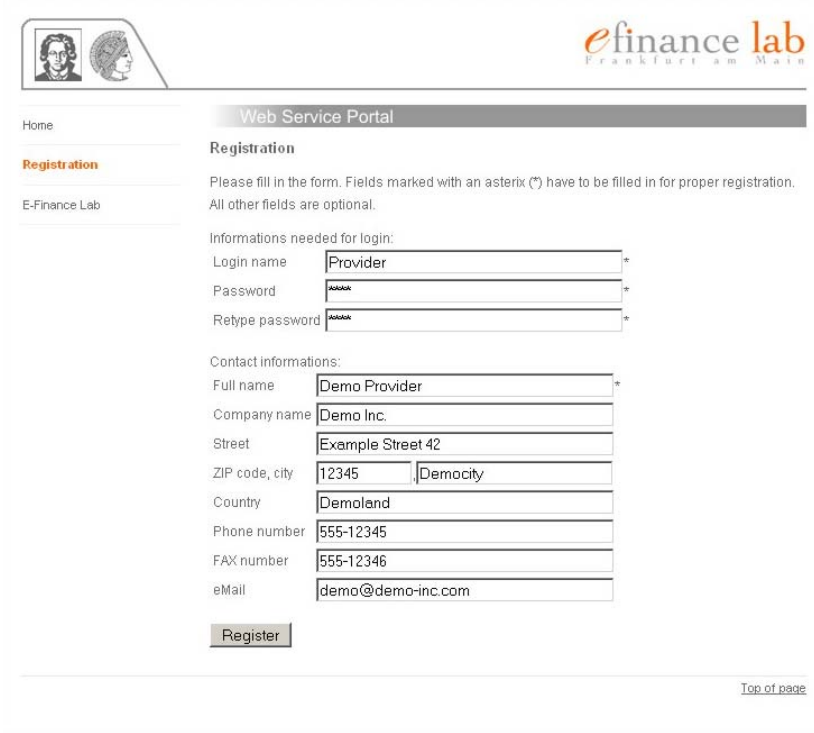

Figure 5: Registration of the provider at the portal

The SLA is evaluated by the *SLA Management component*. In this context information about the Web Services, e.g. provider name, name of the Web Service, guaranteed QoS metrics and the validity, are extracted and stored in the database.

To model Service Level Agreements we use IBM's Web Service Level Agreement (WSLA) framework ([10], [16]). WSLA is based on XML Schema [40] and divided in three parts: In the section *Parties* the organisations 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.

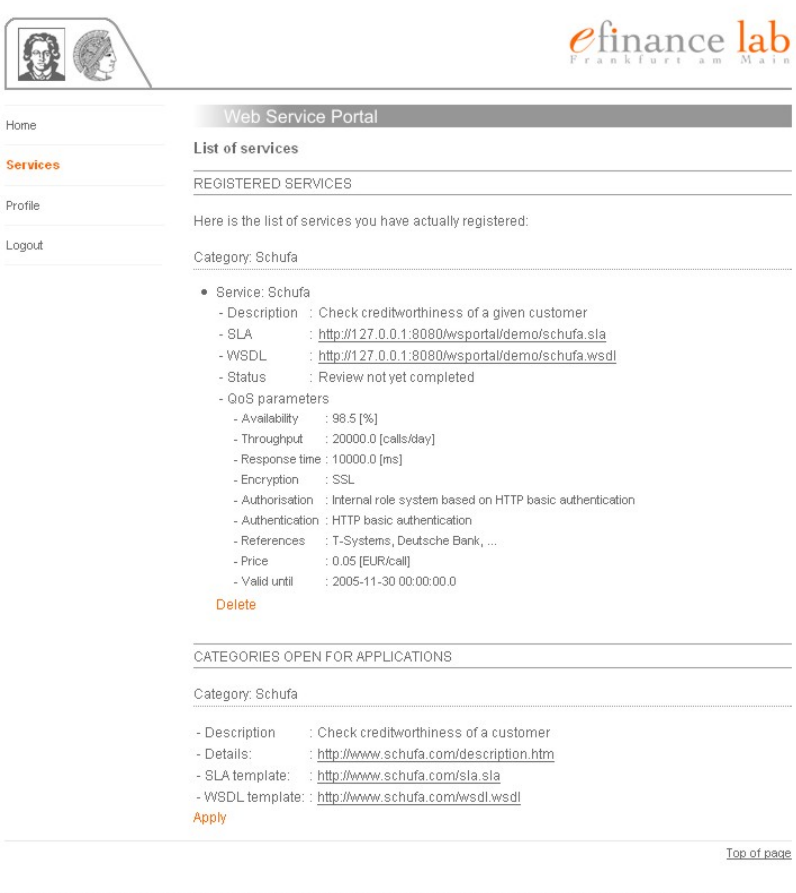

Figure 6: List of the registered Web Services and categories

Figure 7 shows an excerpt of a SLA for a Credit Rating Web Service. Within this SLA an availability of 99.5% is assured and the validity of the Web Service is defined as well.

```
<ServiceDefinition name="CreditRatingService">
  <ServiceLevelObjective name=”SLO_For_Availability”>
       <Obliged>Company</Obliged>
       <Validity>
            <Start>2004-08-01</Start>
            <End>2004-08-31</End>
       </Validity>
       <Expression>
            <Predicate type=”Greater”>
               <SLAParameter>Availability</SLAParameter>
               <Value>99.5</Value>
            </Predicate>
       </Expression>
       <EvaluationEvent>NewValue</EvaluationEvent>
  </ServiceLevelObjective>
</ServiceDefinition>
```

Figure 7: Excerpt of a SLA for a Credit Rating Web Service

The Web Service composition is realised in BPEL4WS (Business Process Execution Language for Web Services), because this specification has become a de facto standard [23]. For the execution of the BPEL4WS process we use the BPEL4WS engine provided by IBM AlphaWorks [14]. Figure 8 illustrates the invocation of a Credit Rating Web Service modelled by BPEL4WS. In BPEL4WS this is realised by the <invoke> construct.

```
<invoke name="CreditRating"
     partner="Bank1"
     portType="ns1:ratingWS"
     operation="validCustomer"
     inputVariable="creditRequest"
     outputVariable="creditResponse">
       …
</invoke>
```

Figure 8: Invocation of a Credit Rating Web Service by the BPEL4WS process

However, in our approach the BPEL4WS engine does not directly invoke a certain Web Service (e.g. a Credit Rating Web Service), but the *Proxy component* instead. This is realised by modifying the WSDL file when the Web Service is registered at the portal in a manner that the physical address of the particular Web Service (e.g. http://www.schufa-company-

a.de/web/services/Schufa) is substituted by the address of the *Proxy component* (e.g. http://localhost:8081/axis/WSproxy). This modification of the physical Web Service address is performed by the *SLA Management component* automatically. The *Proxy component* acts as a server waiting with a daemon thread listening on a port (e.g. 8081) for incoming Web Service invocations that arrive as http requests. These http requests contain the SOAP message for the invocation of the particular Web Service. When an incoming Web Service invocation arrives on the particular port, the daemon thread starts a new client thread to process the inbound invocation of the BPEL4WS engine.

Figure 9 shows an excerpt of a http request to invoke a Credit Rating Web Service with the URL „/wsproxy/Schufa“ and the SOAP parameter „_surname“ and „_firstname“.

```
POST /wsproxy/Schufa HTTP/1.0 SOAPAction: ""
User-Agent: Axis/1.1 Host: 192.168.12.200:8081
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="…" …>
   <soapenv:Body>
      <m:validCustomer xmlns:m="…" …>
        <_surname xsi:type="xsd:string">Miller</_surname>
        <_firstname xsi:type="xsd:string">Peter</_firstname>
      </m:validCustomer>
   </soapenv:Body>
</soapenv:Envelope>
```

Figure 9: http request for the invocation of a Credit Rating Web Service

First of all, the client thread reads the incoming http request (e.g. Web Service URL and the SOAP message) and stores this data in a request object generated for this purpose. The request object is passed on to the *Selection component* by the client thread of the *Proxy component*. The task of the *Selection component* is selecting a suitable Web Service in compliance with the pre-defined rules by the user (cp. Chapter 4). This process consists of the following steps:

- The internal registry is browsed to find the appropriate Web Service category (e.g. credit rating) by evaluating the URL (e.g. /wsproxy/creditRating).
- Based on this category, a so-called *Target* is created. This *Target* is declared as: <Protocol>:<Web Service category>:<Search criterion; Constraint >. If the Credit Rating Web Service with highest score without any constraint should be invoked, the *Target* is modelled as followed: <DB>:<creditProcess.rating>:<maxScore, noConstraint>
- To determine the optimal Web Service, the *Target* is handed on to a SearchFactory generated for this purpose. The SearchFactory delivers an object according the specified protocol. This protocol depends on the registry that stores and manages the Web Services. Due to simplicity it is possible to use a SQL database. Our approach supports UDDI as registry as well. In this case the protocol has to be specified as *UDDI*.
- The *Selection component* receives the object with the endpoint URL (i.e. the physical address) of the best suitable Web Service according to the ranking calculated by the *Rating Component*. Using this endpoint URL the *Selection component* invokes the particular Web Service. At the same time the *Accounting component* is started as well.
- If the Web Service returns an output, this output will be sent as a response to the client thread by the *Selection component*. The client thread forwards this response to the BPEL4WS engine. After this step the Web Service invocation is completely finalised and the client thread is terminated.
- Due to the tracked data, the *QoS Monitoring component* checks if there are any SLA violations. In this case a warning is generated and the provider of the particular Web Service is notified.

The interaction of the participating architectural components described in this section is shown as a collaboration diagram in Figure 10.

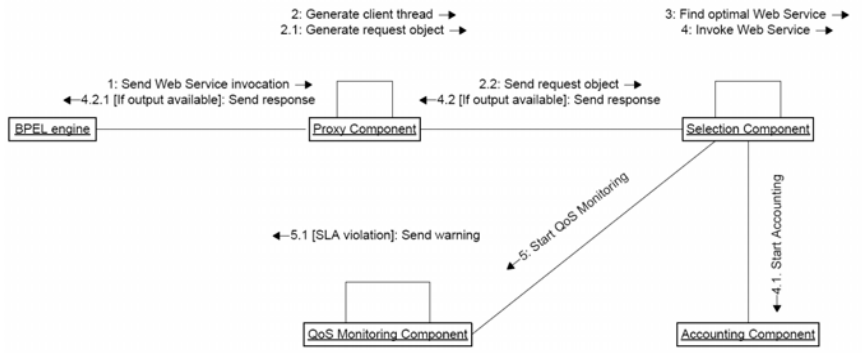

Figure 10: Interaction of the participating architectural components

6. Conclusion and Outlook

In the era of globalisation and sophisticated customers flexible business processes are crucial for the long-term success of an enterprise. Web Services as

realisation of a Service-oriented Architecture (SoA) gain in importance not only to integrate legacy systems, but also to establish flexible, cross-organisational business processes. The adoption of Web Services in critical business processes depends on the manner how Web Services guarantee Quality of Service (QoS).

Thus, we present an integrated Web Service architecture with comprehensive QoS support. This architecture supports the assessment of Web Services to assure that only those Web Services will be used in critical business processes that satisfy the requirements defined by the user. The selection and execution of a certain Web Service depends on its QoS properties that must be guaranteed in Service Level Agreements (SLAs). The compliance with such SLAs is monitored by a component of the architecture as well. Furthermore we described our prototypical implementation of the Web Service architecture presented in this paper.

Our further research activities will focus on the performance evaluation of the Web Service architecture we propose in this paper. Additionally, we will research on self-configuration of business processes. Self-configuration of business processes aims at business processes that combine process patterns to reach a certain business goal (e.g. assessment of a loan request) according to a particular business context (e.g. certain clients and past experiences).

References

[1] Alonso, G., “Myths around Web Services”, in: Bulletin of the Technical Committee on Data Engineering, vol. 25, no. 4, 3-9 (2002).

[2] Alonso, G., Casati, F., Kuno, H., Machiraju, V., “Web Services. Concepts, Architectures and Applications”, Springer Berlin (2003).

[3] Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., Weerawarana, S., “Business Process Execution Language for Web Services”, version 1.1, http://www.ibm.com/developerworks/library/ws-bpel (2003).

[4] ATIS Committee. “Accountability”, http://www.atis.org/tg2k/_accountability.html (2001).

[5] Becker, J., Kugeler, M., Rosemann, M. (Eds.), “Process Management. A Guide for the Design of Business Processes”, Springer Berlin (2003).

[6] Berbner, R., Mauthe, A., Steinmetz, R., "Supporting dynamic E-Finance Processes (in German)", in: Horster, P. (Ed.), "Elektronische Geschaeftsprozesse 2004", Klagenfurt, Austria, 44-54 (2004).

[7] Cardoso, J., Sheth, A., “Semantic e-Workflow Composition”, Technical Report 02-004, LSDIS Lab, Computer Science Department, University of Georgia, Athens, USA (2002).

[8] Cardoso, J., Sheth, A., Miller, J., “Workflow Quality of Service”, in: Proceedings of the International Conference on Enterprise Integration and Modeling Technology and International Enterprise Modeling Conference (ICEIMT/IEMC 2002), Valencia, Spain, 303-311 (2002).

[9] Castellanos, M., Casati, F., Dayal, U., Shan, M.-C., “Intelligent Management of SLAs for Composite Web Services”, in: Proceedings of 3rd International Workshop on Databases in Networked Information Systems (DNIS), Aizu, Japan, 158-171 (2003).

[10] Dan, A., Ludwig, H., Pacifici, G., “Web Services Differentiation with Service Level Agreements”, IBM developerworks, http://www-106.ibm.com/developerworks/webservices/library/ws-slafram (2003).

[11] Fensel, D., Bussler, C., “The Web Service Modeling Framework WSMF”, in: Electronic Commerce: Research and Applications, vol. 1, no. 2, 113-137 (2002).

[12] Gouscos, D., Kalikakis, M., Georgiadis, P., “An Approach to Modeling Web Service QoS and Provision Price”, in: Proceedings of the 4th International Conference on Web Information Systems Engineering Workshops (WISEW 2003), Rome, Italy, 121-130 (2003).

[13] Heckmann, O., “A System-oriented Approach to Efficiency and Quality of Service for Internet Service Providers”, PhD thesis, Technische Universitaet Darmstadt, http://elib.tu-darmstadt.de/diss/000522/ (2004).

[14] IBM AlphaWorks, “IBM Business Process Execution Language for Web Services Java Run Time (BPWS4J)”, http://www.alphaworks.ibm.com/tech/bpws4j (2004).

[15] Kalepu, S., Krishnaswamy, S., Loke, S. W., “Verity: A QoS Metric for Selecting Web Services and Providers”, in: Proceedings of the 4th International Conference on Web Information Systems Engineering Workshops (WISEW 2003), Rome, Italy, 131-139 (2003).

[16] Keller, A., Ludwig, H., “The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services”, in: Journal of Network and Systems Management, vol. 11, no. 1, 57-81 (2003).

[17] Koetzle, L., Rutstein, C., Liddell, H., Buss, C., Nakashima, T., “Reducing Integration's Cost”, Forrester Research Report, http://www.forrester.com/go?docid=11981 (2001).

[18] Leymann, F., Roller, D., “Production Workflow, Concepts and Techniques”, Prentice Hall Upper Saddle River (2000).

[19] Leymann, F., Roller, D., “Web Services: Business Processes in a Web Services World. A quick overview of BPEL4WS”, IBM DeveloperWorks, http://www-106.ibm.com/developerworks/webservices/library/ws-bpelwp (2002).

[20] Leymann, F., Roller, D., “Using flows in information integration”, IBM Systems Journal, vol. 41, no. 4, 732-742 (2002).

[21] Mani, A., Nagarajan, A., “Understanding Quality of Service for Web Services”, IBM DeveloperWorks, http://www-106.ibm.com/developerworks/webservices/library/ws-quality.html (2002).

[22] Malone, T. W., Crowston, K., Lee, J., Pentland, B. T., Dellarocas, C., Wyner, G. M., Quimby, 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, vol. 45, no. 3, 425-443 (1999).

[23] OASIS, “OASIS Web Services Business Process Execution Language (WSBPEL)”, Technical Committee, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel (2003).

[24] Ortner, E., “Repository Systems”, Informatik Spektrum, vol. 22, no. 4, 235-251 (1999).

[25] Papazoglou, M. P., van den Heuvel, W. J., “Leveraging legacy assets”, in: Papazoglou, M. P., Spaccapietra, S., Tari, Z. (Eds.), “Advances in Object-Oriented Modeling”, MIT Press, Cambridge, MA, 131-160 (2000).

[26] Papazoglou, M. P., “Service-Oriented Computing: Concepts, Characteristics and Directions”, in: Proceedings of 4th International Conference on Web Information Systems Engineering (WISE 2003), Rome, Italy, 3-12 (2003).

[27] Preuner, G., Schrefl, M., “Integration of Web Services into Workflows through a Multi-Level Schema Architecture”, in: Proceedings of the 4th IEEE International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS 2002). Newport Beach, USA, 51-60 (2002).

[28] Ran, S., “A model for Web Services discovery with QoS”, in: ACM SIGecom Exchanges, vol. 4, no. 1, 1-10 (2003).

[29] Schmitt, J. B., “Heterogeneous Network Quality of Service Systems”, Kluwer Academic Publishers (2001).

[30] Schmidt, R., “Web Services Based Architectures to Support Dynamic Inter-organizational Business Processes”, in: Proceedings of the International Conference on Web Services - Europe (ICWS-Europe 2003), Erfurt, Germany, 123–136 (2003).

[31] Sharma, A., Adarkar, H., Sengupta, S., “Managing QoS through Prioritization in Web Services”, in: Proceedings of the 4th International Conference on Web Information Systems Engineering Workshops (WISEW 2003), Rome, Italy, 140-148 (2003).

[32] Sivashanmugam, K., Miller, J., Sheth, A., Verma, K., “Framework for Semantic Web Process Composition”, Technical Report 03-008, LSDIS Lab, Computer Science Department, University of Georgia, Athens, USA (2003).

[33] Sommerville, I., “Software Engineering”, 6. Edition, Addison-Wesley Munich (2001).

[34] Steinmetz, R., Berbner, R., Martinovic, I., "Web Services supporting flexible Business Processes in the Financial Service Industry“ (in German), in: Sokolovsky, Z.; Loeschenkohl, S. (Eds.), "Industrialisierung der Finanzwirtschaft", Gabler Wiesbaden (2005).

[35] Tian, M., Gramm, A., Naumowicz, T., Ritter, H., Schiller, J., “A concept for QoS Integration in Web Services”, in: Proceedings of the 4th International Conference on Web Information Systems Engineering Workshops (WISEW 2003), Rome, Italy, 149-155 (2003).

[36] UDDI Spec Technical Committee, “UDDI Version 3.0.1”, Technical Committee Specification, http://uddi.org/pubs/uddi_v3.htm (2003).

[37] W3C, “Web Services Description Language (WSDL) 1.1”, W3C Note, http://www.w3.org/TR/wsdl20 (2001).

[38] W3C, “SOAP Version 1.2”. W3C Recommendation, http://www.w3.org/TR/soap12/ (2003).

[39] W3C, “Web Services Architecture Requirements”, W3C Working Group Note, http://www.w3.org/TR/wsa-reqs/ (2004).

[40] W3C, “XML Schema Part 0: Primer Second Edition”, http://www.w3.org/TR/xmlschema-0/ (2004).

[41] 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 (2001).

[42] Workflow management Coalition, “Workflow standard - Terminology glossary”, Technical Report WFMC-TC-1011, Workflow Management Coalition, Version 2.0 (1996).

[43] Yang, J., Papazouglou, M. P., van den Heuvel, W. J., “Tackling the Challenges of Service Composition in E-Marketplaces”, in: Proceedings of 12th International Workshop on Research Issues in Data Engineering: Engineering E-Commerce/E-Business Systems (RIDE 2002), San Jose, USA, 125-133 (2002).

[44] Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., Sheng, Q. Z., “Quality Driven Web Services Composition”, in: Proceedings of the 12th International Conference on World Wide Web, Budapest, Hungary, 411-421 (2003).

Author information

Rainer Berbner. Rainer Berbner received his Master degree (Dipl.) from the University of Technology Darmstadt in Joint Business and Computer Sciences (Wirtschaftsinformatik) in 2003. Since June 2003 he is a researcher at Multimedia Communications Lab (KOM) of Prof. Steinmetz at the Darmstadt University of Technology.
Rainer Berbner researches in the area of IT architectures supporting distributed business processes with a special focus on Web Services. In this context he is interested in QoS-enabled Web Services. His research activities contribute to the E-Finance Lab project (http://www.efinancelab.de).

Oliver Heckmann. Oliver Heckmann finished his Ph.D. in computer science with the title “A System-oriented Approach to Efficiency and Quality of Service for Internet Service Providers” in 2004. Since then he is research group leader of the Multimedia Distribution & Networking groups of the Multimedia Communications Lab (KOM) at the Darmstadt University of Technology. He is working for the eFinance Lab (www.efinancelab.de) and the Internetökonomie (www.internetoekonomie.uni-frankfurt.de) research projects. From 2000 to 2004 he was a researcher at the same lab working in the M3I (www.m3i.org) and letsqos (www.letsqos.de) research projects.
His research areas are Internet Service Providers, IT Architectures and Peer-to-Peer Networking. His special focus is quality of service, efficiency and scalability in these environments.

Ralf Steinmetz. Ralf Steinmetz received the M.Sc. (Dipl.-Ing.) degree and the Ph.D. (Dr.-Ing.) degree, working in the area of Petri nets and concurrent programming languages with focus on communications, from the Darmstadt University of Technology, both in electrical engineering, in 1982 and 1986, respectively. He was research engineer at Philips and IBM. He became responsible for the establishment, definition and realization of multimedia projects at IBM, Heidelberg. In January 1993 he was nominated as IBM "Chief Designer" for IBM Germany. From 1997 to 2001 he has been the Director of the GMD (German - National Research Center for Information Technology) institute IPSI (Integrated Publications- and Information Systems Institute) in Darmstadt. In 1999 he founded the httc (Hessian Telemedia Technology Competence Center) with focus on applied networked multimedia services being applied to education projects.
He is an ICCC Governor, IEEE Fellow, and ACM Fellow.
Since 1996 Ralf Steinmetz is Professor at the Departments of "Electrical Engineering and Information Technology" and "Computer Science" at the Darmstadt University of Technology and head of the "Multimedia Communications" (KOM) Lab. His research interests are the areas of multimedia distribution, multimedia networking and multimedia semantics.
In 2003, the E-Finance Lab was established with 3 professors of economics of the Johann Wolfgang Goethe-University of Frankfurt and Ralf Steinmetz from the Darmstadt University of Technology.