An approach for the Management of Service-oriented Architecture (SoA) based Application Systems

Rainer Berbner[1], Tobias Grollius[2], Nicolas Repp[1], Oliver Heckmann[1], Erich Ortner[2], Ralf Steinmetz[1]

[1]Multimedia Communications Lab – KOM
Technische Universität Darmstadt
Merckstrasse 25
64283 Darmstadt
Germany
berbner@kom.tu-darmstadt.de
repp@kom.tu-darmstadt.de
heckmann@kom.tu-darmstadt.de
steinmetz@kom.tu-darmstadt.de

[2]Department of
Business Information Systems
Technische Universität Darmstadt
Hochschulstr. 1
64289 Darmstadt
Germany
grollius@winf.tu-darmstadt.de
ortner@winf.tu-darmstadt.de

Abstract: Flexible business processes are a key success factor for enterprises to succeed in globalised markets. The Service-oriented Architecture (SoA) concept is very well suited to support flexible business processes and application systems because capabilities (in form of services) can be composed in the most efficient way to achieve a high level of agility. However, the management of SoA-founded application systems is often neglected. Thus, we present an approach that enhances the traditional SoA 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 globalised 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 and business to the new conditions, e.g. [HC03].

We thank our colleagues of the E-Finance Lab for the constructive discussions. A special thanks goes to Michael Spahn who has supported us by the implementation of WSQoSX. The work on this paper is partly sponsored by the E-Finance Lab (http://www.efinancelab.de).

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 [Oe95]. Today, companies mostly use standard software 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 standard software [Oe05]. Even simple changes of business processes demand a tremendous customizing effort. Meeting this challenge, there is an evolution that aims at flexible, agile and reliable application systems realised by a clear separation of control and business logic [LR00]. As a consequence, changes at the process layer can be made with almost no effects on the business logic and vice versa. Additional flexibility can be achieved at the process layer by using reference models [FL04]. This means that processes do not have to be modelled from scratch, but can be derived from domain specific models. An even more effective way of process modelling is the reusability of process components that can be orchestrated to business processes. However, a necessary requirement is the availability of sufficient number of process components to realize individual processes [La97]. Related to the business logic additional flexibility can be achieved by following the component-based application development approach, e.g. [Or05], [SGM02]. The basic idea behind this concept is that software components can be composed and orchestrated to complex applications. A major advantage of this approach is that process components designed at the process layer can be mapped onto software components. The assembly of components can be made at design or at run time. The selection and assembly of components at run time leads to loosely coupled application systems that provide a very high degree of flexibility and agility [GO05].

Nowadays, the Service-oriented Architecture (SoA) paradigm is recommended as an emerging architectural blueprint enabling flexible process-oriented application systems. In this paper we propose the enhancement of a SoA by dedicated management functionalities to discover, select, assemble, and execute appropriate components for orchestrating flexible component-based application systems, e.g. monitoring and Service Level Agreement (SLA) management. As a validation of our approach we present WSQoSX as a prototypical implementation based on Web Service technology.

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 ([DLP03], [SHM02], [JMS02]) or Quality of Service (QoS) aspects ([GKG03], [KKL03], [Ra03]). However, the realisation 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. [ET04].

The remainder of this paper is organised as follows: In Section 2 we introduce the basic concept of workflow technology, SoA, and component-based system engineering as well as how these approaches fit together. The management of SoA-based application systems is 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 Basic Concepts

In this section we introduce the core concepts needed to implement our management approach for SoA-based application systems.

2.1. Workflow Technology

Workflow management systems (WfMS) are used to model and control business processes (workflows). Workflow schemas are the results of workflow modelling and the basis of the control of a business process. A workflow schema describes a workflow under different perspectives, e.g. functional, informational and operational, using the language of the workflow management system [JB96]. In order to create flexible business processes in the sense that modifications like changes of the process sequence or the insertion or omission of sub processes can be made quickly and efficiently, the component-oriented approach shall be applied on workflow management: Workflow schema parts (so called process components) are assembled to a workflow schema as a complete description of the business process. For further details we refer to similar approaches presented by [MCH03], [Aa99] and [Ac04].

2.2. Component-based Application Development

In the previous chapter, we briefly mentioned how to build flexible processes and control them using workflow technology. If the tasks in a process should be executed by software, the software application has to be as flexible as the processes. Applications built of loosely coupled components offer this flexibility. According to [Ac02], a component “consists of different (software) artefacts. It is reusable, self-contained and marketable, provides services through well-defined interfaces, hides its implementation and can be deployed in configurations unknown at the time of [its] development.”

To connect and control loosely coupled components, workflow technology, in particular WfMSs are well-suited 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 software component, the WfMS invokes the previously determined software component via its interface, provides it with required input parameters and - if so - receives the results of the execution in order to transfer them to following organisational resources. Binding software components to process components of a WfMS-controlled business process abolishes the restrictions for flexible process design

and changes that are related to conventional IT applications. As an additional prerequisite to flexible applications, only the selection of variants of a component but not their modification can be allowed [RR03]. Variants of a component differ slightly in particular characteristics such as functionality or interfaces. If only similar components are available opposed to a perfectly suitable component, these should not be adapted to a special usage in an application. Instead of that, a new variant of the component has to be produced and used in the planned application [GO05].

Software components as well as process components must be specified in an unified way and stored in a component catalogue. A unified specification framework was proposed by [Ac02].

2.3. Service-oriented Architecture (SoA)

There are a lot of definitions of the term software architecture, e.g. [BCK03], [Fo02]. For our purpose we prefer the comprehensive definition of Krafzig, Banke and Slama [KBS05]: “A software architecture is a set of statements that describe software components and assign the functionality of the system to these components. It describes the technical structure, constraints and characteristics of the components and the interfaces between them. The architecture is the blueprint for the system and therefore the implicit high level plan for its construction”.

A Service-oriented Architecture (SoA) is a specific software architecture based on services as fundamental elements for integrating and developing applications, e.g. [KBS05], [Pa03], [Ba03]. Services are specific software components. They are well-defined, self-contained and encapsulate high-level business functionality. Services communicate with each other by sending and receiving messages. Services can adopt different roles. 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*. A SoA is characterised by the loosely coupling of the services involved. This means that services can be replaced by other services at runtime. A SoA supports location transparency. This means that services should have their definitions and location information stored in a repository and be accessible by a variety of clients that could locate and invoke the services irrespective of their location [Pa03]. 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 [KBS05]. As a consequence, a SoA shifts the focus from technical to business requirements. From a business perspective, SoA-based processes and application systems have the potential to achieve significant cost saving due to the reduction of maintenance costs and the fact that a SoA can bee seen as an enabling framework for BPO (Business Process Outsourcing) ([BMS04], [Be05], [SBM04]). However, it is not possible to eliminate technological dependencies at all because an appropriate infrastructure (e.g. enterprise service bus) is still required. Furthermore, standardisation is a key success factor and there are still open issues in this field.

2.4. Flexible SoA-based Application Systems

Based on the results of a functional business process decomposition as part of an in-depth business process analysis in most cases it is possible to derive process components, which are composed to a complete business process and can be mapped onto software components later on (e.g. services as an integral part of a SoA).

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 of this decomposition 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”.

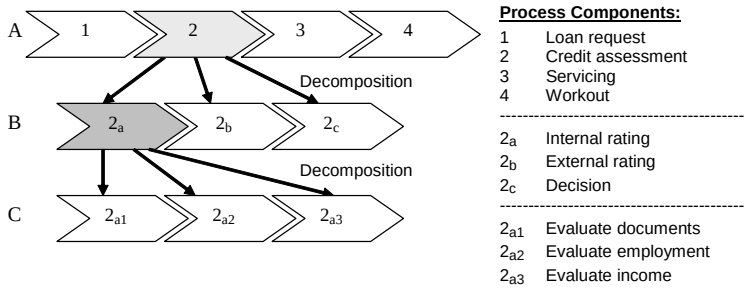

Figure 1: Decomposition of a generic credit process [BHS05]

As shown in Figure 1 every single activity, sub process and process itself can be a process component. Thus, it is possible to map both a single activity and a complete sub process to a software component, depending on the degree of granularity and flexibility needed and desired by the business. The functionality to support or implement process components can be supplied by services that are specialised software components (cp. Section 2.3).

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 WfMS as an orchestration engine (cp. Section 2.1). There are two main advantages of this approach: On the one hand, existing business processes can be rapidly adapted to a changing environment by recombining existing and novel services. On the other hand, it is possible to integrate legacy systems by encapsulating them with a service interface allowing a transparent usage of their functionality. Furthermore, using loosely coupled services it is possible to obtain services not only from internal service providers but from external service vendors supporting a certain type of Business Process Outsourcing (BPO). This concept, called Service-based BPO, is depicted in Figure 2.

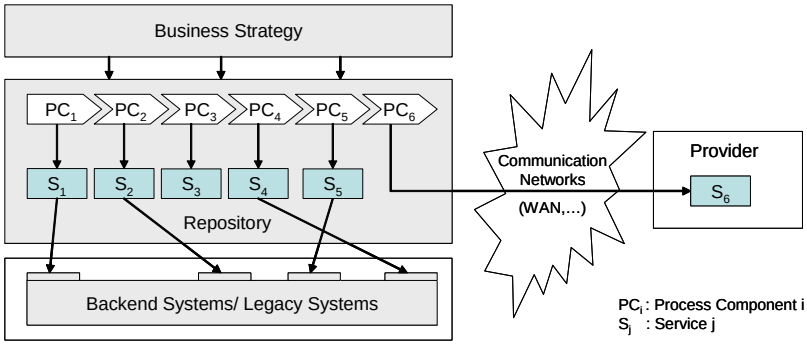

Figure 2: Service-based Business Process Outsourcing (BPO)

Our approach supports the reusability of process components in different business processes, which leads to cost reduction, improved reliability and a faster development of component-based and IT-supported business processes. Business processes can be plugged together out of process components using a construction kit [Ma99]. Repositories [Or99] can be used to manage and store those process components, information about services and mappings between them.

3 Management of SoA-based Application Systems

We present the Service-oriented Architecture (SoA) concept as an architectural blueprint for enabling flexible processes and application systems based on components. To establish flexible and reliable application systems we have designed a management system for SoA-based applications that provides additional functionality. Components offering this management functionality form an application management system (Figure 3). Depending on their main usage time in an 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 discovered. The most suitable ones are selected to be integrated in the application system in the subsequent step (selection). Assuming the business process is already decomposed into process components, the *Discovery Component* performs a search for appropriate services. 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 larger functionality.

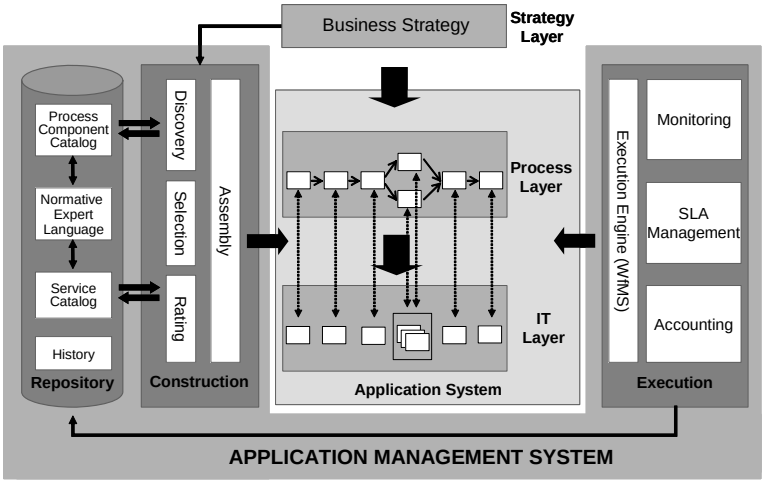

Figure 3: A management system for SoA-based applications and processes

If the search is not successful as one or more services are not available or not known to the repository, the *Discovery Component* has to stop the construction process and inform the application system designer about the missing components. Either the business process has to be adapted or the missing components have to be implemented. A precondition of searching is the detailed specification of services to search for using a reference specification. Based on the reference specification, services are searched for in both internal (e.g. an enterprise repository) and external service catalogues (e.g. a service marketplace). The comparison of the reference specification with services specifications may result into 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 standardised. For this purpose a normative expert language could be used for the specification of the functionality of process components and services. Alternatively semantic web technologies [LS99] or ontologies [Fe03] can be used.

3.2. Selection Component

The *Selection Component* decides which services are to be integrated in the application system. For that purpose, it assesses and compares the characteristics of the services being delivered by the *Discovery Component*. Besides the functional compliance, the *Selection Component* also considers the non-functional metrics, e.g. costs or availability. Assessment, comparison and decision-making methods are provided by the field of decision theory. For example, Kontio introduced the AHP (Analytical Hierarchy Process) as a decision-supporting method for component-oriented software construction [Ko95].

3.3. Rating Component

The *Rating Component* calculates a ranking based on a given SLA and runtime behaviour of former executions of the according service. In a SLA, different criteria for QoS are defined, both measurable and non-measurable values. Both types of criteria have to be evaluated by the *Rating Component*, although non-measurable values like reputation and security have to be assessed by IT experts before. This assessment can be done based on an evaluation matrix [Be05]. 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 [Be05]. This ranking can be used as a basis for a posterior dynamic selection of 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”).

3.4. Assembly Component

After determining the services, their integration in an application system has to be prepared. As mentioned before, we recommend a clear and strict distinction between the control and execution of processes. We propose using workflow technology as a basic system for the generic control function. Therefore, it is necessary to enable and ensure the interaction between the services and a workflow engine. 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 run-time 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 as *Execution Component* that is capable to manage not only software services but also other organisational resources (employees, knowledge, etc.) [JB96].

3.6. Accounting/ Billing Component

Accounting is the process of tracing information systems activities to a responsible source [ATI01] 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 *Accounting/ Billing Component* enables the realisation of different billing strategies, e.g. pay-per-use or flat fees.

3.7. Monitoring Component

At execution time the QoS parameters defined by the SLA are monitored by the *Monitoring Component*. The component analyses the data collected by the *Accounting*

component and compares them to the guaranteed metrics 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*.

3.8. SLA Management Component

SLAs (Service Level Agreements) are bilateral contracts between a customer and a service provider used to guarantee Quality of Service (QoS), especially the levels of availability, performance and serviceability of a service. Besides, a SLA can contain pricing, contractual and other information. SLAs are the result of a negotiation between a customer and a service provider. SLAs are defined in RFC 3198 [We01]. 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 necessary controlled removal of services. The *Deconstruction Component* identifies the effective dependencies between the service to be removed and other services and the consequences of removing the service for active workflow instances and the workflow schema [CL00].

3.10. Enterprise Repository System as Integration Means

Repositories are documentation systems. On a meta-language-level language artefacts, (e.g. data structures, software components, or process components) are described in a structured way. The data structure for the documentation of these language artefacts is called meta schema or documentation structure of a repository system. As an essential element of an application management system the repository includes catalogues for process and software components as well as descriptions of other organisational resources [GO05]. The domain-specific integration of the components of different categories and the component-based application systems (e.g. the overcoming of semantical heterogeneity) can be managed using a normative expert language that is also documented in the repository [Or99].

4 WSQoSX: A Prototypical Implementation using Web Service Technology

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

4.1. Web Service Technology

Web Services can be regarded as a SoA-based technology. A Web Service is defined a self-contained, modular business application that has open, Internet-oriented, standard-based interfaces [Be04]. The Web Service core standards are formed by WSDL (Web Service Definition Language), SOAP (Simple Object Access Protocol) and UDDI (Universal Description, Discovery and Integration). 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. [Al04]).

4.2. QoS related to Web Services

To establish robust business processes it is crucial that requirements on the business process layer can be mapped onto the underlying IT architecture. Changes in business processes (e.g. new business requirements) have to be reflected in the underlying IT architecture and the involved Web Services. Derived from those business requirements the Quality of Service (QoS) parameters can be defined. As a consequence, the IT architecture needs modules that explicitly manage Quality of Service. QoS related to Web Services refers to the non-functional properties of a Web Service, e.g. [MN02]. Enterprises are not willing to rely on Web Services hosted by external service providers if there is no guarantee about their QoS needs. The most important QoS parameters are ([GKG03], [KKL03]) are availability, throughput, response time and error rate. Additionally, there are as well non-measurable QoS attributes like security and reputation. However, the management of these QoS metrics is often neglected. Thus, we introduce our prototypical implementation based on Web Service technology that offers a comprehensive QoS-aware management support.

4.3. WSQoSX – An Application Management System based on Web Services

As a proof-of-concept, a prototype of an application management system supporting QoS in a Web Services based Service-oriented Architecture was developed at Technische Universität Darmstadt. The application management system called WSQoSX (Web Service Quality of Service architectural Extension) can be seen as a partial implementation of the concepts presented in section 3. The implementation uses BPEL4WS (Business Process Execution Language for Web Services) 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 and services. 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 metrics, prepares it for further processing and stores the information inside the repository of the application management system. SLAs are modelled based on IBM's Web Service Level Agreement (WSLA) framework ([KL02], [Lu03]).

- *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 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 realised by sending a message to the *Selection Component* to terminate the bad-performing 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 4.

After receiving an HTTP request containing the Web Service URL and the SOAP message sent by the BPEL engine (Step 1 in Figure 4), 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. 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).

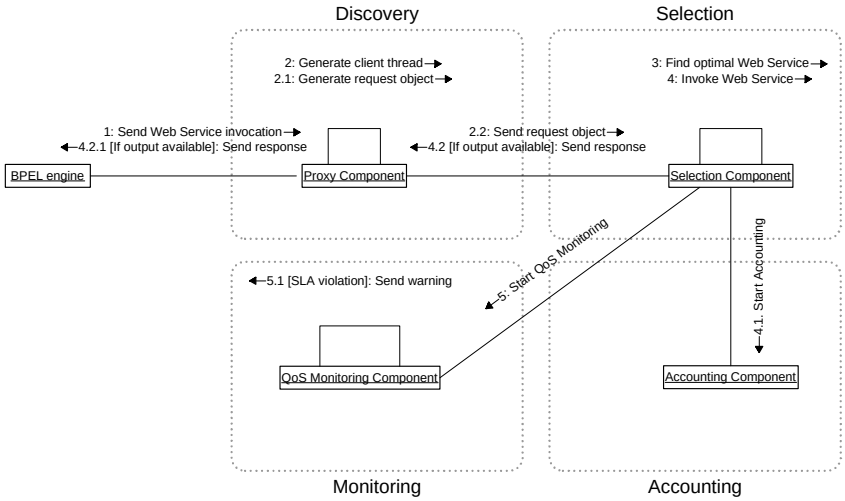

Figure 4: Interaction of the different components

5 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, e.g. 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.

References

[Aa99] van der Aalst, W. M. P.: Flexible Workflow Management Systems: An Approach Based on Generic Process Models. Berlin, 1999.

[Ac02] Ackermann, J. et al.: Standardized Specification of Business Components. Gesellschaft für Informatik, Bonn, 2002.

[Ac04] Acker, H. et al.: Aspekte der komponentenorientierten Entwicklung adaptiver prozessorientierter Unternehmenssoftware. 1st joint conference on architectures, components, and applications (AKA 2004), Augsburg, 2004.
[Al04] Alonso, G. et al.: Web Services. Concepts, Architectures and Applications., Springer Verlag, Berlin, Heidelberg, 2004.
[ATI01] ATIS Committee: Accountability. 2001; http://www.atis.org/tg2k/_accountability.html, accessed on 19.08.2005.
[Ba03] Barry, D. K.: Web Services and Service-Oriented Architecture: The Savvy Manager's Guide. Morgan Kaufmann Publishers, 2003.
[BCK03] Bass, L.; Clements, P.; Kazman, R.: Software Architecture in Practice. Addision-Wesley, 2003.
[Be04] Bellwood, T. et al.: UDDI Version 3.0.2. OASIS, 2004; http://uddi.org/pubs/uddi_v3.htm, accessed on 13.05.2005.
[Be05] Berbner, R. et al.: Eine Dienstgüte unterstützende Web Service-Architektur für flexible Geschäftsprozesse. Wirtschaftsinformatik, vol. 47, no. 4, 2005; pp. 268-277.
[BHS05] Berbner, R.; Heckmann, O.; Steinmetz, R.: An Architecture for a QoS driven composition of Web Service based Workflows. Networking and Electronic Commerce Research Conference (NAEC 2005), Riva Del Garda, Italy, 2005.
[BMS04] Berbner, R.; Mauthe, A.; Steinmetz, R.: Unterstützung dynamischer E-Finance-Geschäftsprozesse. Konferenz Elektronische Geschäftsprozesse 2004 (EGP 2004), Klagenfurt, Österreich, 2004.
[CL00] Crnkovic, I.; Larsson, M.: Component Configuration Management. Blekinge, 2000.
[DLP03] Dan, A.; Ludwig, H.; Pacifici, G.: Web service differentiation with service level agreements. IBM, 2003; http://www-106.ibm.com/developerworks/webservices/library/ws-slafram/, accessed on 14.05.2005.
[ET04] Esfandiari, B.; Tosic, V.: Requirements for Web Service Composition Management. 11th HP-OVUA 2004 Workshop, Paris, 2004.
[Fe03] Fensel, D.: Ontologies - A Silver Bullet for Knowledge Management and Electronic Commerce. Springer-Verlag, Berlin, Heidelberg, 2003.
[FL04] Fettke, P.; Loos, P.: Referenzmodellierungsforschung. Wirtschaftsinformatik, vol. 46, no. 5, 2004; pp. 331-340.
[Fo02] Fowler, M.: Patterns of Enterprise Application Architecture. Addision-Wesley, 2002.
[GKG03] Gouscos, D.; Kalikakis, M.; Georgiadis, P.: An Approach to Modeling Web Service QoS and Provision Price. 4th International Conference on Web Information Systems Engineering Workshops (WISEW'03), Rom, Italien, 2003.
[GO05] Grollius, T.; Ortner, E.: A Concept of Configuring Flexible Application Systems for Business and Administration Processes. 4th International Conference on Information Systems Technology and its Applications, Palmerston North, 2005.
[HC03] Hammer, M.; Champy, J.: Rengineering the Corporation: A Manifesto for Business Revolution. HarperBusiness, New York, 2003.
[JB96] Jablonski, S.; Bussler, C.: Workflow Management. International Thomson Computer Press, London, 1996.
[JMS02] Jin, L.-j.; Machiraju, V.; Sahai, A.: Analysis on Service Level Agreement of Web Services. HP Laboratories, Palo Alto, USA, HPL-2002-180, 2002.
[KBS05] Krafzig, D.; Banke, K.; Slama, D.: Enterprise SOA. Service-Oriented Architecture. Best Practices. Prentice Hall, Upper Saddle River, USA, 2005.
[KKL03] Kalepu, S.; Krishnaswamy, S.; Loke, S. W.: Verity: A QoS Metric for Selecting Web Services and Providers. 4th International Conference on Web Information Systems Engineering Workshops (WISEW'03), Rom, Italien, 2003.
[KL02] Keller, A.; Ludwig, H.: The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. IBM Research Report, 2002.

[Ko95] Kontio, J.: OTSO: A Systematic Process for Reusable Software Component Selection. University of Maryland, College Park, USA, 1995.
[La97] Lang, K.: Gestaltung von Geschäftsprozessen mit Referenzprozessbausteinen. Gabler Verlag, Wiesbaden, 1997.
[LR00] Leymann, F.; Roller, D.: Production Workflow, Concepts and Techniques. Prentice Hall, Upper Saddle River, USA, 2000.
[LS99] Lassila, O.; Swick, R. R.: Resource Description Framework (RDF) Model and Syntax Specification. 1999; http://www.w3.org/TR/PR-rdf-syntax/, accessed on 26.08.2005.
[Lu03] Ludwig, H. et al.: Web Service Level Agreement (WSLA) Language Specification. IBM Corporation, 2003; http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf, accessed on 14.05.2005.
[Ma99] Malone, T. W. et al.: Tools for Inventing Organizations: Toward a Handbook of Organizational Processes. Management Science, vol. 45, no. 3, 1999; pp. 425-443.
[MCH03] Malone, T. W.; Crowstone, K.; Herman, G. A.: Organizing Business Knowledge. The MIT Press, Cambridge, USA, 2003.
[MN02] Mani, A.; Nagarajan, A.: Understanding quality of service for Web services. IBM developerWorks, 2002; http://www-106.ibm.com/developerworks/webservices/library/ws-quality.html, accessed on 15.04.2005.
[Oe05] Oehler, K.: Business Engineering bei der Einführung betriebswirtschaftlicher Standardsoftware - Auswirkungen einer serviceorientierten Architektur. HMD-Praxis der Wirtschaftsinformatik, vol. 241, no., 2005; pp. 35-44.
[Oe95] Österle, H.: Business Engineering: Prozess- und Systementwicklung. Springer-Verlag, Berlin, Heidelberg, 1995.
[Or05] Ortner, E.: Component-Based Application Architecture for Enterprise Information Systems. Data Management in a Connected World 2005, 2005.
[Or99] Ortner, E.: Repository Systems. Informatik Spektrum, vol. 22, no. 4, 1999; pp. 235-251.
[Pa03] Papazoglou, M. P.: Service -Oriented Computing: Concepts, Characteristics and Directions. Fourth International Conference on Web Information Systems Engineering (WISE'03), Rome, Italy, 2003.
[Ra03] Ran, S.: A model for Web Services discovery with QoS. ACM SIGecom Exchanges, vol. 4, no. 1, 2003; pp. 1-10.
[RR03] Ravichandran, T.; Rothenberger, M. A.: Software Reuse Strategies and Component Markets. Communications of the ACM, vol. 46, no. 8, 2003; pp. 109-114.
[SBM04] Steinmetz, R.; Berbner, R.; Martinovic, I.: Web Services zur Unterstützung flexibler Geschäftsprozesse in der Finanzwirtschaft. in Handbuch Industrialisierung der Finanzwirtschaft. Gabler Verlag, Wiesbaden, 2004.
[SGM02] Szyperski, C.; Gruntz, D.; Murer, S.: Component Software. Addison-Wesley, Upper Saddle River, 2002.
[SHM02] Sahai, A.; Durante, A.; Machiraju, V.: Towards Automated SLA Management for Web Services. HP Laboratories, Palo Alto, USA, 2002; http://www.hpl.hp.com/techreports/2001/HPL-2001-310R1.pdf, accessed on 14.05.2003.
[We01] Westerinen, A. et al.: RFC 3198 - Terminology for Policy-Based Management. 2001; http://www.faqs.org/rfcs/rfc3198.html, accessed on 12.07.2005.