Karastoyanova, Dimka (Ed.); Kazhamiakin, Raman (Ed.); Metzger, Andreas (Ed.); Pistore, Marco (Ed.)

Research Report

Workshop on Service Monitoring, Adaptation and Beyond: A workshop held at the ServiceWave 2008 conference - Proceedings

ICB-Research Report, No. 34

Provided in Cooperation with:
University Duisburg-Essen, Institute for Computer Science and Business Information Systems (ICB)

Suggested Citation: Karastoyanova, Dimka (Ed.); Kazhamiakin, Raman (Ed.); Metzger, Andreas (Ed.); Pistore, Marco (Ed.) (2009) : Workshop on Service Monitoring, Adaptation and Beyond: A workshop held at the ServiceWave 2008 conference - Proceedings, ICB-Research Report, No. 34, Universität Duisburg-Essen, Institut für Informatik und Wirtschaftsinformatik (ICB), Essen

This Version is available at:
https://hdl.handle.net/10419/58147

ICB
Institut für Informatik und
Wirtschaftsinformatik

Dimka Karastoyanova
Raman Kazhamiakin
Andreas Metzger
Marco Pistore (Eds.)

Workshop on Service Monitoring, Adaptation and Beyond

34

ICB-RESEARCH REPORT

A workshop held at the ServiceWave 2008 conference – Proceedings

UNIVERSITÄT
DUISBURG
ESSEN

ICB-Research Report No. 34

June 2009

Die Forschungsberichte des Instituts für Informatik und Wirtschaftsinformatik dienen der Darstellung vorläufiger Ergebnisse, die i. d. R. noch für spätere Veröffentlichungen überarbeitet werden. Die Autoren sind deshalb für kritische Hinweise dankbar.

The ICB Research Reports comprise preliminary results which will usually be revised for subsequent publications. Critical comments would be appreciated by the authors.

Proceedings

Edited By:

Dimka Karastoyanova,
University of Stuttgart, Germany
Karastoyanova@iaas.uni- stuttgart.de

Raman Kazhamiakin,
FBK-IRST, Trento, Italy
Mail: raman@fbk.eu

Andreas Metzger,
University of Duisburg-Essen, Germany
andreas.metzger@sse.uni-due.de

Marco Pistore,
FBK-IRST, Trento, Italy
Pistore@fbk.eu

Contact:

Institut für Informatik und
Wirtschaftsinformatik (ICB)
Universität Duisburg-Essen
Universitätsstr. 9
45141 Essen

Tel.: 0201-183-4041
Fax: 0201-183-4011
Email: icb@uni-duisburg-essen.de

ISSN 1860-2770 (Print)
ISSN 1866-5101 (Online)

Abstract

This ICB Research Report constitutes the proceedings of the first workshop on Monitoring, Adaptation and Beyond (MONA+ 2008), which was held on December 13, 2008 in Madrid, Spain. MONA+ has been collocated with the ServiceWave 2008 conference.

Table of Contents

1 Preface

The advances in modern technology and the constantly evolving requirements implied by dynamic business environments impose new challenges for engineering and provisioning service-based applications (SBAs). SBAs have to become drastically more flexible: They should be able to operate and evolve in highly dynamic environments and be able to adequately identify and react to various changes in these environments. In such a setting, adaptability becomes a key feature of SBAs as it enables these applications to continuously change themselves to satisfy new requirements and demands dictated by the environment. The ability of the application to adapt strongly relies on the presence of monitoring mechanisms and facilities. By monitoring we understand the methods and mechanisms that allow identifying, detecting, and even predicting critical events and situations that occur both in the SBA as well as in its environment and that require an SBA to change its configuration, behaviour, presentation, and so forth.

A variety of approaches and techniques addressing different forms of monitoring and adaptation have been proposed to date, but these results are still fragmented. The definition of more comprehensive and holistic approaches is crucial for delivering robust, dependable, and highly adaptable SBAs. This requires the alignment and integration of the efforts of researchers from various disciplines and research areas. It includes the "vertical" integration of disciplines such as business process management, service composition and service infrastructure, as well as "horizontal" integration, where different competencies, such as requirements engineering, design, quality assurance, and realization/implementation, should be considered along the life-cycle of SBAs.

The MONA+ workshop aimed at bringing together researchers and practitioners from different disciplines with varying research background, with the goal of trying to understand and establish the shared knowledge necessary for developing comprehensive approaches for SBA monitoring, adaptation and beyond. MONA+ has been organized as part of the ServiceWave 2008 conference, which took place in Madrid, Spain in December, 2008. It featured 5 paper presentations, one invited talk and the presentation and discussion of the S-Cube adaptation and monitoring vision. MONA+ has attracted 20 attendees. It has been sponsored by the S-Cube Network of Excellence (www.s-cube-network.eu) within the European Community's Seventh Framework Programme FP7/2007-2013.

We cordially thank Sam Guinea for accepting our invitation to give an invited talk on "Monitoring and Adaptation in Open-world Software" and extend our gratitude to all participants who contributed to the success of the workshop.

The VaMoS organizers

Dimka Karastoyanova, Raman Kazhamiakin, Andreas Metzger, Marco Pistore

2 Technical Programme

Invited Talk

Research Papers

Position Statement

Monitoring and Adaptation in Open-world Software

Sam Guinea

Politecnico di Milano Dipartimento di Elettronica e Informazione, Italy
guinea@elet.polimi.it

Abstract. The functional and non-functional requirements of today's software have sky-rocketed, and software needs to rise to the challenge and become enablers for scenarios that were previously unimaginable. Ubiquitous, context- and situational-aware applications are common, and distributed systems are the norm. Service technologies have appeared on the landscape, bringing distributed ownership to the picture, meaning designers must now wrap their heads around new and extremely complex landscapes. Software now lives in an open world. No longer can good designers constrain, at design time, the complexity and the number of problems that can arise at run time. On the contrary, the open world is made up of dynamic systems, of components and services that constantly enter or exit the field, or that change their functional and non-functional qualities without notice. Our focus shifts heavily to run-time management, and in particular to monitoring and adaptation. Continuos monitoring has been introduced since it has become impossible to solely rely on classic static validation and verification techniques. Monitoring means ”keeping an eye” on our system, to figure out, as soon as possible, if something is not going as expected. On the other hand, adaptation tries to change how a system is behaving, with the goal of optimizing quality of service. A system can be adapted as an answer to an anomalous situation, to try to ”keep things on track”, it can be pre-emptive, if the system makes use of prediction mechanisms, or it can simply be performed to take advantage of changes in the context of execution.
In the talk I will pinpoint the main problems that need to be addressed in open-world software, with a focus on monitoring and adaptation. I will briefly present the most prominent existing aproaches, and give an update on my own research agenda. Finally, I will attempt to highlight interesting research issues that remain open, and take a look at what could be our next steps going forward.

Towards Monitoring of Key Performance Indicators Across Partners in Service Networks

Branimir Wetzstein[1], Olha Danylevych[1], Frank Leymann[1], Marina Bitsaki[2], Christos Nikolaou[2], Willem-Jan van den Heuvel[3], and Mike Papazoglou[3]

[1] Institute of Architecture of Application Systems, University of Stuttgart, Germany
lastname@iaas.uni-stuttgart.de
[2] Computer Science Department, University of Crete, Greece
lastname@tsl.gr
[3] Dep. of Information Systems and Management, Tilburg University, Netherlands
W.J.A.M.vdnHeuvel, mikep@uvt.nl

Abstract. In an ever increasing dynamic environment, companies are forced to co-operate in order to meet customer needs effectively. They set up Service Networks (SN) trying to create a win-win situation for all participants of the network. The calculation of value in an SN is based on key performance indicators (KPIs) which measure the performance of underlying cross-organizational business processes. As for the calculation of KPIs of these processes monitoring information from several paricipants is needed, in an SN it is no more sufficient for the participants to monitor just KPIs of their internal processes, e.g., by using Business Activity Monitoring technology. The participants now have to provide a set of monitoring events to the other partners in the SN. In this paper, we describe an approach to monitoring of KPIs across partners in a service network. An SN is mapped to a service choreography and a monitoring agreement is created which specifies how KPIs are decomposed to events that participants in the choreography have to provide. We present our approach based on a case study from the telecommunications domain.

1 Introduction

In today's networked economy, companies are not independent, isolated entities, but they must act in a concerted manner to survive in an ever increasing dynamic environment. Thereby, interacting companies build networks to serve their joint customers in a dynamic manner, focusing on optimizing their financial benefits at the individual and network level. Recently, Service Networks (SNs) have been proposed to model such networks and analyze and optimize company's business collaborations [1]. SN is a graph-based approach to model a business environment as a set of business partners and their relations. SNs reside on a high abstraction business level depicting partners as nodes and their offering and revenues as edges. Modeling a business landscape as SN, allows, on the one hand, calculating the value gained by a single partner when joining the collaboration network. On the other hand, an SN perspective gives the possibility to measure the value of the whole network. The value calculation is used for measuring the profitability of the SN, which can lead to adaptation of SNs, for example, through outsourcing.

Service Networks focus on cooperations between partners in terms of offerings and revenues and don't detail the concrete interactions between the partners. In addition, the dependencies between the actors in an SN don't necessarily express the temporal dependencies between the partner interactions. Each offering - each single edge - in the SN is realized through a set of complex interactions between the partners. The partner interactions are not of interest on SN level but represent one of the main concerns of the level of business processes and choreographies as part of business process management [2]. The refinement from SNs to executable processes has been motivated in [3] and first steps towards mapping of SNs to service choreographies are described in [4].

The value calculations in an SN are based on a set of Key Performance Indicators (KPIs). KPIs are business metrics which are used for measuring the performance of underlying business processes of the SN. Traditionally, companies have measured the performance of their internal processes using established concepts such as Business Performance Management and technologies like Business Activity Monitoring [5]. In the setting of an SN, this is no more sufficient. Partners now have and want to share SN relevant information of their internal processes with other partners. In order to do so the partners have to provide monitoring events or already measured metrics to the "outside", so that the overall performance of the SN can be evaluated.

In this paper, we propose a method of how to model and monitor KPIs across partners in a service network. We assume that the SN is mapped to service choreography descriptions, as described in [4]. Based on the choreography description, we describe how KPIs are decomposed to events each partner has to provide for the overall KPIs to be calculated. We introduce in this context the concept of a *monitoring agreement* which specifies the monitoring events each partner has to provide. The monitoring agreement includes partner descriptions, the events which each partner provides, and how these events are aggregated to calculate the overall KPIs of the SN. We describe the concepts based on a case study from the telecommunications domain.

This paper is organized as follows: Section 2 describes the case study we have chosen for evaluating the concepts of this work. Section 3 gives an overview of our approach. The definition of monitoring agreements is described in Section 4. Section 5 sketches aspects related to runtime monitoring. Section 6 positions our approach among existing work and finally, Section 7 concludes the paper and presents future work.

2 Case Study

The case study discussed in the following is based on the "Enhanced Telecom Operation Map" (eTOM), which is a reference model for business processes of the telecommunications industry [6]. In particular, we describe a service network that is formed in order to set up a new DSL service.

Figure 1 depicts the involved parties of the DSL installation process and the offering flows between the parties. The main actors are the Subscriber (customer) and the Service Provider, which offers DSL products to its customer. The functionalities of the Service Provider relevant to this part of the model are distributed among the Call Center, Service Agent and Field Agent. The Subscriber interacts with the Call Center to order the DSL line and to report problems or complaints. In order to make the installation, the Call

Center performs some actions (like checking availability) and forwards the request to the Service Agent. The Service Agent is responsible for the setup and configuration of the order. He finally contacts the Field Agent to perform the service installations at the customer site. The Billing Agent is responsible for setting up the monthly billing procedeure for the customer. In our case study, we further assume that Call Center, Service Agent, and Field Agent are not part of the SP company, but separate organizations that have been outsourced.

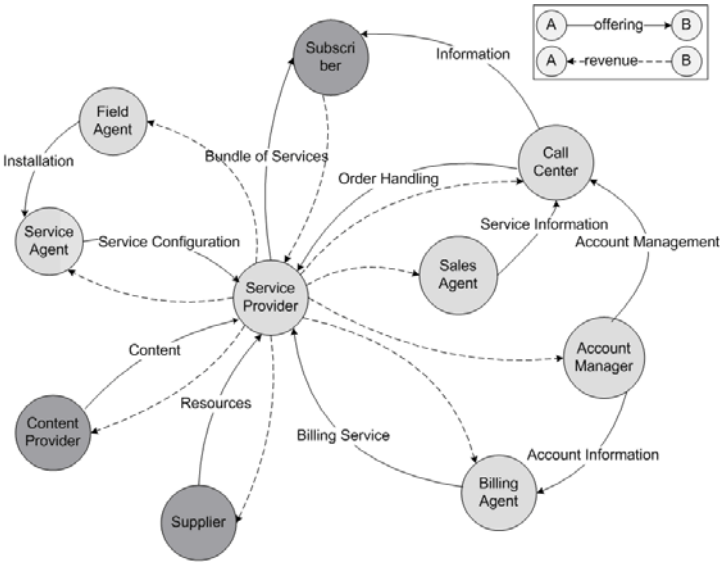

Fig. 1. The Service Network for a new DSL Service Set-Up

The value of Service Provider (SP) in the presented service network can be calculated as described in [3]. The calculation takes into account the following KPIs: the revenues, the costs the SP pays for the needed resources to provide the service, and the satisfaction of customers (subscribers). These three ”top-level” KPIs are aggregations of other finer-grained KPIs. The revenues are calculated based on the prices of the service and the number of customer orders in a certain period. The costs are a function over labor and contract rates with the Call Center and Field Agents and internal process costs and the number of employees together with some additional costs (material costs). The customer satisfaction is a function over the order fulfillment lead time, deadline adherence, price of service, number of customers that cancelled their order, number of customers that complained during the time period of order processing, and perfect order fulfilment (order processed in full and in time as requested by customer without customer complaints).

Some of the described KPIs can be calculated with information available already on the level of service network. For example labor rates and the service price are already fixed in service level agreements between the service provider and customer.

Other KPIs, the ones we focus on in this paper, are measured on BPM layer based on underlying business processes, such as order fulfillment time, deadline adherence, number of complaints etc. The Service Provider wants to measure these KPIs in a timely manner in order to be able to calculate its value in this service network. For the calculation of these KPI, he depends on information from its business partners. Assume e.g. the KPI perfect order fulfilment. In order to calculate this KPI, we need information from the Call Center (time of order receipt, and number of cusotmer complaints), Service Agent (information on whether we could install and configure the requested producted as wished by the customer), Field Agent (time of product delivery and installation at customer site). That means for the calculation of the KPI information from more than one business partner is needed. This information should also be provided in a timely manner, i.e. based on events as they happen in the process.

3 Overview of the Approach

In this paper, we take a top-down approach in which service networks are mapped to service choreographies and further refined to executable business processes [3]. We distinguish between three layers and between a functional and non-functional view, as shown in Figure 2.

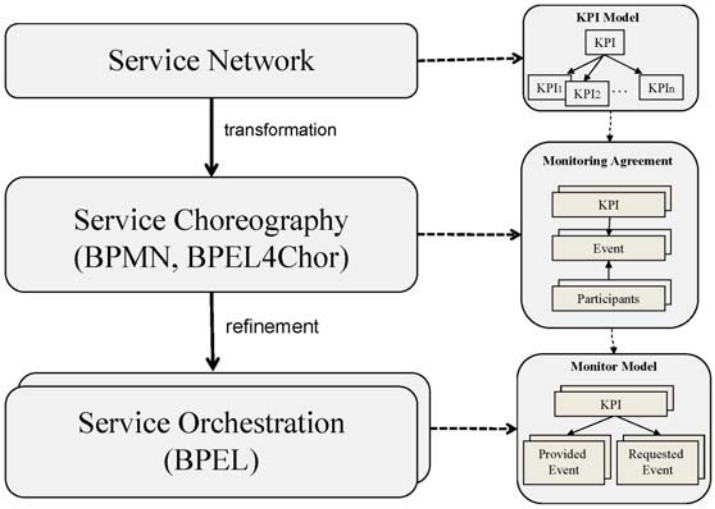

Fig. 2. Overview of the Approach

In the functional view, on the topmost layer, SNs are modeled. An SN specifies the interactions between partners on a very high level in terms of offerings and revenues. In the next step, the SN is mapped to a service choreography, which specifies the

message exchanges between the partners in the SN. This mapping can be performed semi-automatically [4]. The choreography can be first modeled in a technology-independent notation such as BPMN, and then later transformed to a technology specific choreography specification, such as BPEL4Chor [7]. On the orchestration level, each participant in the choreography implements his part of the process and exposes it to the outside as a Web service. This implementation of the process can be done in WS-BPEL.

In the non-functional view, on SN level, value calculations of the SN as a whole and of the participants in the SN are performed. Value is calculated based on KPIs, such as revenues, costs, and customer satisfaction. KPIs can again be defined based on other KPIs. For example, customer satisfaction can be defined based on the customer satisfaction index, the number of customer complaints, deadline adherence, and average order fulfillment lead time. All KPIs of the SN and their calculations are part of the *KPI model*.

In order to measure the KPIs of the SN in a timely manner, we have to specify how they are to be monitored based on operational business processes. On choreography level, the public processes involving message exchanges are modeled, serving as an agreement between partners on how they communicate together. We argue, that on this level one should also *agree* on which events each partner has to provide in order to calculate the KPIs of the KPI model. This is because on choreography level no private process information is modeled, and thus events based on public process models also should not lead to privacy issues. For example, in order to calculate the KPI *Order Fulfillment Lead Time*, the Call Center has to provide an *OrderReceived* event which contains an *order receipt date*, while the Field Agent provides the *ProductInstalled* event with the *product installation date*. The monitoring agreement specifies KPIs which are to be evaluated for the service choreography, and how they are decomposed to events each partner has to provide. The agreement involves also the definition of event formats, and monitoring mechanisms which define how the events can be retrieved at process runtime.

The monitoring agreement has two purposes for each participant in the choreography: (i) it serves as a requirements specification, on which events the participant has to provide; (ii) the monitoring agreement also specifies how KPIs are calculated based on all events provided by all partners. Based on this information, a participant can subscribe for the events of other partners, in order to be able to calculate the KPIs internally, if he wants to. Therefore, a participant creates a *monitor model*, which defines how the provided events are to be created and how the needed events from other partners are requested, and how the KPIs are calculated based on those events.

4 Definition of Monitoring Agreements

In this section, we describe how monitoring agreements between partners in an SN are created. A monitoring agreement specifies which information partners have to provide in order to enable monitoring of KPIs.

4.1 Sevice Choreography in the Case Study

Figure 3 shows an excerpt of a choreography description in BPMN for the case study. It models the business process for setting up the DSL line from customer request to

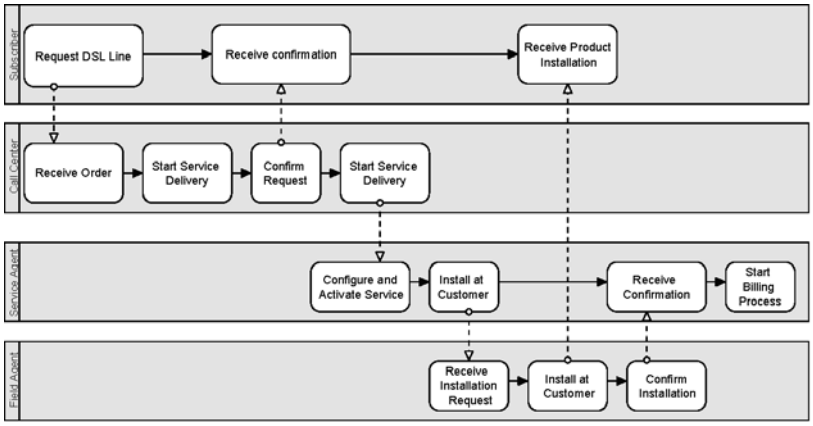

Fig. 3. BPMN Choreography Description for the Case Study SN

installing of the product at customer's site. The choreography involves four partners. The subsriber places an order in the call center. After checking its availability, the call center sends the order to the service agent, which configures the package. The field agent finally installs the product at customer site. The process is distributed between several business units and business partners. For example, we assume that Call Center, Service Agent, and Field Agent are all separate companies (business partners) which were outsourced by Service Provider.

In Table 1 we have listed a set of KPIs which are interesting in this Service Network. For each KPI, we specify its calculation, and define the events which have to be provided by the partners in the choreography. As shown in the Table, the first KPI can be provided by only one partner, the other two need events from more than one partner. Note that events have to be correlated when used in the calculation functions. In the above cases this can happen based on an *Order Identifier* which is set when a new order is received, and which is then used until shipment. This *Order Identifier* has to be part of each event and also of the messages which are exchanged between the partners. Correlation of events is an important concept when specifying the calculation of KPIs, and is well-known from Complex Event Processing (CEP) [8] which is mostly used for implementation of BAM solutions.

4.2 Specification of Monitoring Agreements

As shown in the last section, for the calculation of KPIs events from different partners are needed. In intra-organizational BAM, one also needs to model events and instrument different information systems which emit these events at runtime. Typically, these events are published to a publish/subscribe eventing infrastructure (a topic) which the BAM tool subscribes to. The difference in the SN setting is that monitoring is performed based on cross-organizational processes.

In the case of cross-partner monitoring, where different organizations are involved, one has to *agree* on events (and their content) which are provided by different partners.

KPI	Metric Calculation	Partner. Provided Event
# of Received Orders	count(OrderReceived)	Call Center.OrderReceived
Order Fulfillment Lead Time	t(ProductInstalled) - t(OrderReceived)	Call Center.OrderReceived Field Agent.ProductInstalled
Perfect Order Fulfillment	Order Fulfilment Time $<$ 14 days & ServiceConfigured.status = ”in full” & not(ReceivedCustomerComplaint)	Call Center.OrderReceived Field Agent.ProductInstalled ServiceAgent.ServiceConfigured Call Center.ReceivedComplaint

Table 1. KPIs and their Calculation

Note that in this case there are also several non-technical aspects involved, such as privacy issues. Companies want to restrict insight into their internal processes as much as possible. On the other hand, a certain degree of openess and monitoring support, might be part of a service offering. Discussion of these non-technical issues is out of scope of this paper. As we base our monitoring agreement, and therein specified events, on service choreographies which consist of public process descriptions, privacy issues are minimized. However, we do not restrict ourselves to choreographies; if participants want (or need) to provide events which go beyond information contained in the choreography, they are free to do so.

Figure 4 shows the main concepts needed for the specification of a monitoring agreement. A *monitoring agreement* is specified for a *service choreography* description which contains a set of *participants*, and for each participant a public *process model* which defines its behavior as part of the choreography. If BPEL4Chor is used as a service choregraphy language, the process model is specified in (an abstract profile of) WS-BPEL. The *monitoring agreement* defines a set of *indicators*. An *indicator* is defined based on a *function* which calculates an indicator value based on already defined indicators or based on *events* which are provided by *participants*. Note that we support the special case that no indicators but only events are specified in an agreement. That case is needed if one only wants to track the progress of a process instance (which is signaled by events). Functions can contain boolean, arithmetic and aggregate operators, among others. An *event* contains a set of *properties* which can be arbitrary data items, consisting of a name and a type. An *event* definition can contain a reference (not shown in the Figure) to a process element (process, activity, variable) thus specifying where in the process the event is emitted. In addition to the specified concepts, one needs to specify how the events can be obtained at process runtime, e.g. by publishing and subscribing to a topic (see Section 5).

Listing 5 shows an excerpt of the monitoring agreement for the KPI Order Fullfilment Lead Time as defined in Table 1. It is calculated based on two events provided by the Call Center and Field Agent. The monitoring agreement document references the

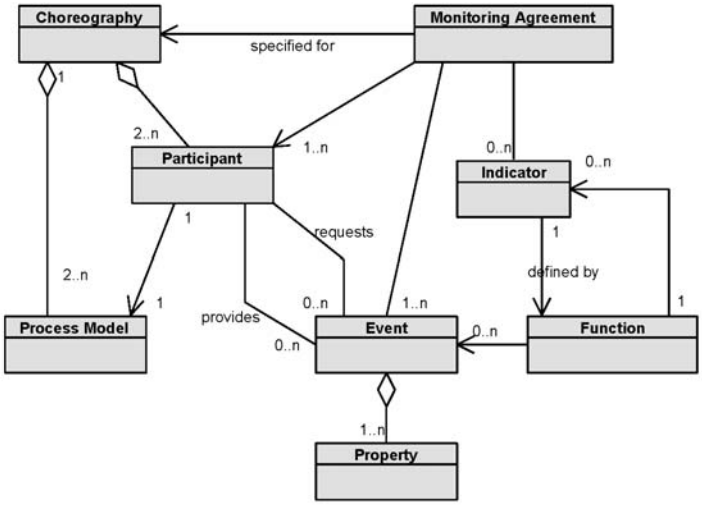

Fig. 4. Overview of the Main Monitoring Agreement Concepts

choreography description and the participants defined in the choreography description. The calculation specification uses a predefined function *duration* and correlates on the *orderId* which is a property of both events.

5 Monitoring Architecture

After creation of the choreography description and the monitoring agreement, each partner implements its internal process according to the choreography, and implements his part of the monitoring agreement. Figure 6 shows a high-level view of the monitoring architecture.

Each partner implements its role in the choreography description, e.g., by refining the abstract BPEL description from the BPEL4Chor descirption to an executable BPEL service orchestration. This service orchestration implements its role in the choreography. In the same manner, each partner has to implement its part of the monitoring agreement, thus providing events to the other partners and receiving events from the other partners.

As we focus on BAM as performance measurement technology, we assume that each partner has an internal BAM implementation. Based on the monitoring agreement, partners have to make sure that they provide events to the outside. As shown in Figure 6, a possible solution is to establish a publish/subscribe topic (a.k.a. publish/subscribe channel), e.g. based on WS-Notification, which is used by all partners in the choreography. The partner thus has to generate events, which might involve prior instrumentation of services and systems, and publishes it to the topic. All partners which are interested in this event and have accordingly subscribed to it, receive this event.

Note that the implementation of the publish/subscribe channel can be hosted by one of the partners or also by a third party. The channel is also only responsible for routing

```
<monitoringAgreement>
  <serviceChoreography name="eTomChor:eTom-Choreography"/>
  <participants>
        <participant name="eTomChor:CallCenter">
         <providedEvents>...</providedEvents>
         <requestedEvents>...</requestedEvents>
        </participant>
        <participant name ="eTomChor:FieldAgent">
     <providedEvents>...</providedEvents>
    </participant>
  </participants>
  <indicators>
   <indicator name="Order Fulfillment Lead Time" unit="hours">
    <duration>
     <event name="ProductInstalled" property="installationDate" />
     <event name="OrderReceived" property="receiptDate" />
     <correlation>
      <equal>
       <event name="ProductInstalled" property="orderId" />
       <event name="OrderReceived" property="orderId" />
      </equal>
     </correlation></duration></indicator>
  </indicators>
  <events>
   <event name="OrderReceived">
    <property name="orderId" type="xsd:string"/>
    <property name="receiptDate" type="xsd:date"/>
   </event>
   <event name="ProductInstalled">...</event>
  </events>
</monitoringAgreement>
```

Fig. 5. Monitoring Agreement for Order Fullfilment Lead Time (simplified)

of events, but not for correlation and aggregation of events for the calculation of KPIs. In this architecture, this is performed by each partner on its own.

6 Related Work

Business activity monitoring approaches in the context of monitoring of KPIs of business processes focus on intra-organizational processes. There exist several research approaches [5, 9] and products [10] which deal with evaluation of process metrics in near real time and their presentation in dashboards. They all have in common that events are emitted as the process is executed, collected by a process monitor and evaluated in near real time. Some solutions focus on monitoring of BPEL processes [9], while others are more general and support an extensible architecture via event adapters [10]. To the best of our knowledge there is no work yet which considers monitoring of KPIs of service networks or service choreographies in a cross-organizational scenario.

Service Level Agreements (SLA) are similar to our problem in that they involve monitoring in a cross-organizational setting. Thereby mostly two partners, the service consumer and the service provider, agree on certain service QoS, typically technical characteristics such as availability and response time. The SLA specifies also how the agreed QoS levels are to be monitored and what happens in case of violations. Web Service Level Agreement (WSLA) [11] is an approach to modeling and monitoring

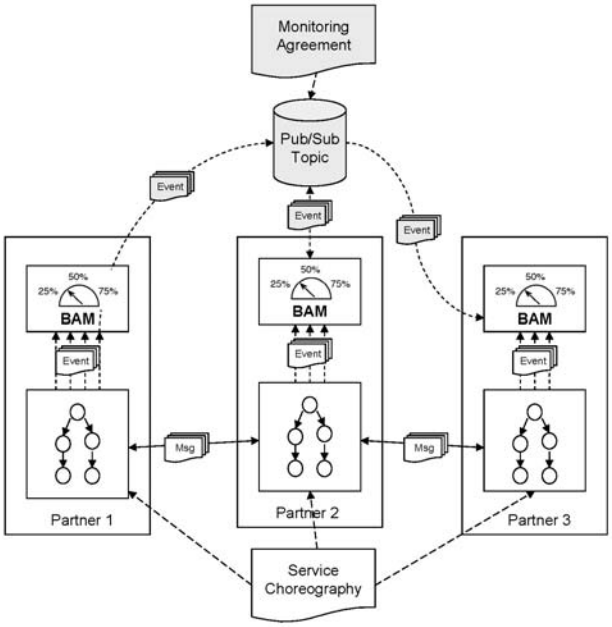

Fig. 6. Monitoring Architecture

of SLAs in the context of Web services. A WSLA-based agreement specifies involved parties, SLA parameters and objectives they agree on, the underlying metrics including measurement directives, and penalties in case of violations. The commonalities with monitoring in our context are that in an SLA partners also *agree* on metrics and how they are to be monitored. In our case, we also need an agreement on metrics and measurements between partners. However, in our case the focus is on monitoring (of potentially more than two partners) and not on guaranteeing certain KPI values. We are also mostly interested in monitoring of process metrics, not in low level QoS metrics, and in particular deal also with cross-partner metrics, which require event correlation, and which are not being dealt with in frameworks such as WSLA.

7 Conclusions and Future Work

In this paper we have presented an approach to monitoring of KPIs in service networks. We have motivated the approach based on a case study, showing that partners have to provide monitoring events to the outside for calculation of KPIs when participating in service networks. Thereby, KPIs are decomposed to events each partner has to provide. We have introduced the concept of a monitoring agreement which is specified on the level of service choreographies. The monitoring agreement describes the calculation of the KPIs based on monitoring events and the obligations of each partner concerning the

provision of those events. Finally, we have sketched a possible monitoring architecture, which is based on a publish-subscribe infrastructure used by all partners.

Our future work includes refining and implementing the framework presented in this paper. We want to base the implementation of the framework on BPEL4Chor for the specification of choreography descriptions and WS-BPEL for implementing orchestrations. The realization includes specifying the monitoring agreement metamodel in detail, including its linkage to BPEL4Chor choreography descriptions, and the semi-automatic generation of a WS-BPEL based monitoring implementation. The monitoring solution should provide both near real time monitoring in BAM fashion and retrieval of monitoring information on demand.

Acknowledgments The research leading to these results has received funding from the European Communitys 7th Framework Programme under the Network of Excellence S-Cube Grant Agreement no. 215483.

References

1. Caswell, N.S., Nikolaou, C., Sairamesh, J., Bitsaki, M., Koutras, G.D., Iacovidis, G.: Estimating Value in Service Systems: A Case Study of a Repair Service System. IBM Systems Journal **47**(1) (2008) 87–100
2. Weske, M.: Business Process Management: Concepts, Languages, Architectures. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2007)
3. Bitsaki, M., Danylevych, O., Van den Heuvel, W.J., Koutras, G., Leymann, F., Mancioppi, M., Nikolaou, C., Papazoglou, M.: An Architecture for Managing the Lifecycle of Business Goals for Partners in a Service Network. In: ServiceWave2008. (2008)
4. Bitsaki, M., Danylevych, O., Van den Heuvel, W.J., Koutras, G., Leymann, F., Mancioppi, M., Nikolaou, C., Papazoglou, M.: Model Transformations to Leverage Service Networks. In: Proceedings of the 4th International Workshop on Engineering Service-Oriented Applications (WESOA 2008), Springer-Verlag (2008)
5. Jeng, J.J., Schiefer, J., Chang, H.: An Agent-based Architecture for Analyzing Business Processes of Real-Time Enterprises. In: Proceedings of the 7th International Conference on Enterprise Distributed Object Computing (EDOC '03), Washington, DC, USA, IEEE Computer Society (2003) 86
6. TeleManagement-Forum: Enhanced Telecom Operations Map (eTOM) - The Business Process Framework For The Information and Communications Services Industry (2003)
7. Decker, G., Kopp, O., Leymann, F., Weske, M.: BPEL4Chor: Extending BPEL for Modeling Choreographies. In: ICWS, Salt Lake City, USA (2007)
8. Luckham, D.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley Professional (2002)
9. Roth, H., Schiefer, J., Schatten., A.: Probing and Monitoring of WSBPEL Processes with Web Services. Proceedings of the The 8th IEEE International Conference on E-Commerce Technology (CEC-EEE'06) (2006)
10. Wahli, U., Avula, V., Macleod, H., Saeed, M., Vinther., A.: Business Process Management: Modeling Through Monitoring Using WebSphere V6.0.2 Products. IBM, International Technical Support Organization (2007)
11. Keller, A., Ludwig, H.: The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. J. Netw. Syst. Manage. **11**(1) (2003) 57–81

Monitoring Adaptable SOA-Systems using SALMon

Marc Oriol, Jordi Marco, Xavier Franch, David Ameller

Universitat Politècnica de Catalunya, Spain
{moriol,jmarco,franch,dameller}@lsi.upc.edu

Abstract. Adaptability is a key feature of Service-Oriented Architecture (SOA) Systems. These systems must evolve themselves in order to ensure their initial requirements as well as to satisfy arising new ones. In SOA Systems there are a lot of dependencies between services, but each service is an independent element of the system. In this situation it is necessary not only ensuring that the system fulfils its requirements but also that every service satisfies its own requirements, and dynamically adapting the system when some of them cannot be ensured. In this paper we propose a SOA system, named Service Level Agreement Monitor (SALMon), for monitoring and adapting SOA Systems at run time. SALMon is based on monitoring the services for detecting Service Level Agreement (SLA) violations. The SALMon architecture is composed of three types of components: Monitors, which are composed of measure instruments themselves; the Analyzer, which checks the SLA rules; and the Decision Maker that performs corrective actions to satisfy SLA rules again. These three types of components are mostly technology-independent and they act as services inside of a SOA system making our architecture very scalable and comfortable for its purpose.

1. Introduction

Service-Oriented Architecture (SOA) has become one of the most successful architectural styles used for the development of software systems. The main characteristic of this architecture is the construction of software solutions based on a group of services that communicate with each other. However, this high coupling also implies a strong dependency among the different components of the SOA system. A failure of a service could imply the malfunction or failure of the whole system.

The emerging research challenge, then, is how we can ensure that the components which are using these services are able to offer the same benefits and accomplishing the user's requirements in case that one of these services fails.

In this context, being able to build self-adaptive SOA systems is a major undertaking. Self-adaptive SOA systems are those which are able to change dynamically the services they use in order to keep fulfilling the Quality of Service (QoS) requirements stated in Service Level Agreements (SLA). Self-adaptive SOA systems demand having several alternative services to use in case that a service is not working properly.

The construction of this kind of SOA systems requires tool support for (1) monitoring services to continuously know their QoS, (2) determine when the SLA is

being violated, and (3) finally take the decision of using an alternative service with the same or similar functionality.

In this paper, we present SALMon, a SOA system itself that uses a monitoring technique to provide runtime QoS information that is needed to detect and eventually correct SLA violations. SALMon is still under development, therefore the prototype we present here should be considered as ongoing research. Notice that although the architecture and interfaces of SALMon are technologically independent so that the tool could be used to monitor any kind of services of a SOA system, we will focus in this first preliminary version on monitoring web services.

The rest of the paper is structured as follows: first we provide a framework for metrics definition based on previous works and the second part is dedicated to the details of SALMon architecture. Finally there is a section for the conclusions.

2. Quality Attributes and Metrics

In our previous work [1], we identified the quality attributes of services building a quality model [2]. Our approach is compliant with the ISO/IEC 9126-1 standard [4] and remarkably we have added some subcharacteristics related to non-technical issues following the advices given in [5]. This model was developed during our participation in a ITEA European project, SODA (Services Oriented Devices & Delivery Architectures, www.soda-itea.org), in which we had the responsibility of identifying and classifying the characteristics needed for defining the quality of Web services.

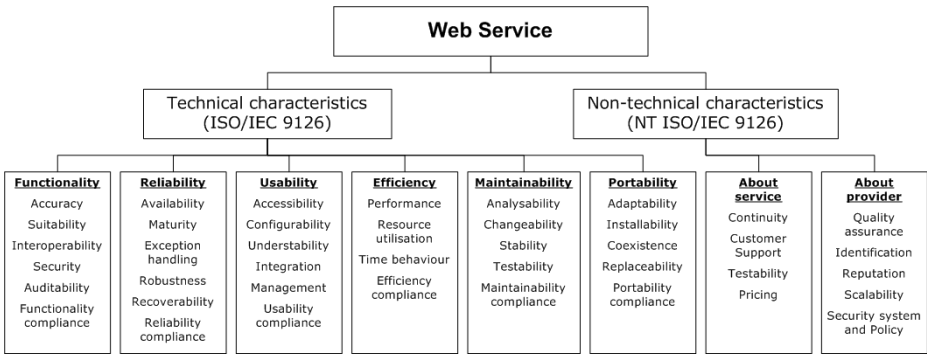

Figure 1: Web Service Quality Characteristics.

In the proposed quality model (Figure 1) we have identified several characteristics. Notice that, for instance, one characteristic is Efficiency and one of its subcharacteristics is the Time Behavior. But Time Behavior itself is not a single measurable concept, therefore we need to define attributes to decompose this subcharacteristic. The attributes are normally dependent on what we want to measure. In our case, since we are focusing on Web services, Response Time and Execution Time are good examples of measurable attributes for Time Behavior.

Time behaviour

Time behaviour is the capability of the software product to provide appropriate response and processing times and throughput rates when performing its function, under stated conditions. In SALMon, we are interested in two particular measurable attributes:

- Response Time: It measures the time that a Web Service takes to give a basic response.
- Execution Time: It measures the time that a Web Service takes to execute a certain job (a method, a process...).

Availability

The availability is the degree to which the system is operable and in a committable state. The user might want to ensure that a service is available.

Accuracy

Accuracy is the capability of the software product to provide the right or agreed results or effects with the needed degree of precision. In this case, the user could want to monitor a concrete functionality of the web service, in this way, we talk about functionality test.

The attributes Response Time and Availability are attributes that belong to the web service since they are related to the whole service. If a web service is not available, none of all its operations will be available. If the response time of the web service is increased (e.g., because of a high demand on the service), the execution time of all the operations of the web service will be affected accordingly. On the contrary, Execution Time and Accuracy are attributes that belong to concrete operations of the web service.

Once these four attributes have been identified, the next step is determining their concrete metrics that will be subject of measure. In Table 1 we present these metrics.

		Metric	Description
Web Service's attributes	Availability	Current availability	It measures the current availability of a Web Service all the time or in slots of time.
		Accumulative availability time	It measures in percentage how much time a Web Service has been available since it has been monitored for a first time.
		Accumulative unavailability time	It measures in percentage how much time a Web Service has not been available since it has been monitored for a first time.
		Average recovery time	It measures in seconds the average time that a Web Service needs to be available again (It considers unavailability).
	Response time	Current response time	It measures the current response time in milliseconds to access to a Web Service.
		Minimum response time	It measures which is the lowest response time in milliseconds to access to a Web Service.
		Maximum response time	It measures which is the maximum response time in milliseconds to access to a Web Service.
		Average response time	It measures which is the average response time in milliseconds to access to a Web Service.
Operation's attributes	Execution Time	Current execution time	It measures the current execution time in milliseconds that a Web Service takes to execute a job. (response+execution)
		Minimum execution time	It measures which is the minimum execution time in milliseconds that a Web Service takes to execute a job.
		Maximum execution time	It measures which is the maximum execution time in milliseconds that a Web Service takes to execute a job.
		Average execution time	It measures which is the average execution time in milliseconds that a Web Service takes to execute a job.
	Test functionality	Current functionality compliance	It measures the current functionality compliance of a Web Service.
		Parameter Accuracy factor	It divides the number of accepted correct type parameters passed to a Web Service method by the number of expected parameters for this method (1 is better).
		Result Accuracy factor	It divides the number of correct type results returned by a Web Service method by the number of expected results of this method (1 is better).
		Fault factor	It divides the number of faults that a Web Service method had generated by the total times that this method had been executed (1 is worse)

Table 1: Metrics defined over the quality attributes measured in SALMon.

An important distinction is between basic metrics and derived metrics. Basic metrics are those which must be monitored to obtain their values. Examples of basic metrics are Current Response Time or Current Availability. Derived metrics are those which can be obtained from a set of basic metrics. For example, the Average Response Time is a derived metric since it can be obtained through the set of Current Response Times in an interval of time. Another example is Recovery Time Failure which is also a derived metric from the basic metric Current Availability. This distinction is important since given the values of a basic metric, there is no interaction with the monitored service to obtain the values of the corresponding derived metrics.

3. Platform Architecture Overview

The architecture of SALMon is a Service Oriented Architecture. Our platform is composed by the following services (see Fig. 2): Analyzer, Decision Maker and Monitor. Another type of module relevant to the SALMon architecture are the Measure Instruments, which are part of the Monitor service. SALMon also uses a service to store the monitoring data and a service for authentication and authorization.

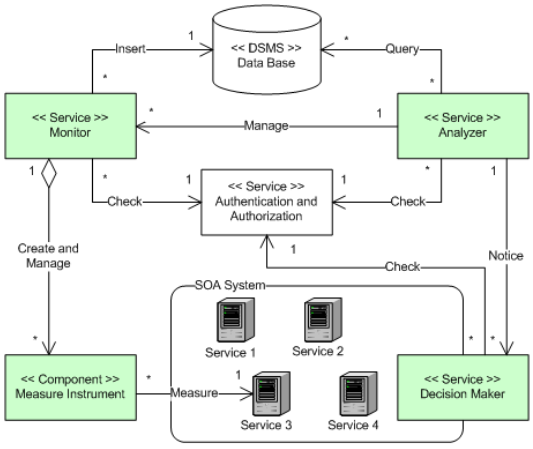

Figure 2: Platform Architecture

In the rest of the section we will present this services and components in detail:

- Monitor service. It uses Measure Instruments to get the information about QoS. Measure Instruments are components generated by the Monitor to communicate with the services in order to get the monitoring information that is relevant for computing the chosen metrics. This information is stored in a data base and is also rendered to the Analyzer.

- Analyzer service. It is responsible of checking for SLA violations in concrete SOA Systems: when a violation is detected, it is notified to the Decision Maker service of the affected SOA System. To attain its goal, the Analyzer manages a Monitor service.

- Decision Maker service. It selects the best treatment to solve the incidences detected by the Analyzer in a concrete SOA system. Each Decision maker is related with one (and only one) SOA System.

3.1. Monitor

The Monitor service is composed of several Measure Instruments for the same SOA System. Measure Instruments are components used to get all the *basic metrics* of the selected quality attributes. Derived metrics will be obtained from them. These components are responsible of bringing the measures to the Monitor, which has the responsibility of maintaining this information updated. The update process is an iterative call to each Measure Instrument in different intervals of time, saving the results in a database.

Since our current approach on monitoring services is intrusive[1], Measure Instruments have the responsibility to minimize the number of interactions performed with the monitored service.

The Monitor needs the information of the service to monitor service's metrics (Response Time and Availability) and also information of the operations to monitor operation's metrics (Execution Time and Functionality Test).

To monitor the metrics of a Service, service details such as *url* and *port* are needed. The time interval between measures to the service is also needed and some services might require a user and a password.

In order to minimize the interaction with the monitored service, all service-metrics share the same time interval between calls and use the same measure instrument. Therefore, if we want to measure the basic metrics current availability and current response time of a service, the same measure instrument is in charge of measure both metrics in the same call.

To monitor the metrics of an operation, further information details are needed. In particular, the name of the operation and at least one valid SOAP message request. To test the accuracy of the operation, we need also to have the knowledge to determine whether if a response message is valid or not. To do so, we use patterns of correct SOAP message responses. If the message response meets the pattern, we say that the response is a valid message, otherwise we say that it's invalid. This approach however, cannot state if the data given in the response is reliable but if it is well-structured and consistent accordingly to the request.

3.2. Analyser

The Analyzer manages Monitors and checks for SLA violations in concrete SOA systems. When a violation is detected it is notified to the Decision Maker of the affected SOA system. In general an Analyzer can handle multiple SOA systems using one Monitor and one Decision Maker for each one. The use of Decision Maker services is optional but if they are not used, the SALMon user is limited to monitoring and SLA violation detection.

The SLA can be configured manually with the interface provided by the Analyzer or automatically with a SLA standard document for each service (e.g., WSLA [3] for

[1] Intrusive (or Active) monitoring approach is also known as Testing.

the case of Web services). We understand SLA as a set of conditions that must be true in some time interval. A condition is composed of the evaluated metric, a relational operator and a value for the comparison (i.e. “Current Response Time < 100ms” is a condition that must be true for the specified service during the specified time interval).

Defining time intervals is important since some conditions are relevant in a specific interval of time or date. For instance, it could be possible that a service is required to be available in a specific timetable, but we could agree that this service can be temporally unavailable in a scheduled time for maintaining purposes.

The Analyzer is also responsible to compute the desired derived metrics from basic metric values stored by the monitor service.

3.3. Decision Maker

The Decision Maker service has a repository of treatments and alternative services for a concrete SOA system. It will automatically select and execute the best treatment for the reported incidences.

Because the kind of job of this service and for security reasons, it is preferred to place the service in the concrete SOA system where it is working.

The Decision Maker has the following responsibilities:

- To take actions when something goes wrong in the SOA system.
- To write reports of the incidences with the taken actions.

3.4 Users

There are two kinds of user in the SALMon architecture, normal user and administrator user. Note that the kind of user is set for each service, for example one user could be administrator of an Analyzer and a Decision Maker but only with normal access to a Monitor service.

The normal users will be limited to the finality of each service while the administrators have extra functionalities: management of the access to the service, establishment of restrictions in services (e.g.. set the maximum number of Measure Instruments in a Monitor), and set the interconnection between services.

The user of the Analyzer service must be authenticated before start working with it, this user must be authorized to use the Monitor services. If the user wants to use Decision Maker services, he/she must also have enough rights in each of the monitored SOA systems. Measure instruments are property of one monitor so they don’t need authentication.

To be able to use SALMon the user will need to have at least a normal user level in one Analyzer and in one Monitor. The Decision Maker is optional but if the user has no access to the Decision Maker or the SOA system has no Decision Maker service, it will be limited to monitoring.

3.5 Use cases

As we can see in Fig. 3, SALMon is composed of 12 groups of use cases. We may distinguish them whether if they belong to a normal user or to the administrator. In the diagram we have also two virtual actors: Analyzer Engine is responsible of checking SLA while Monitor Engine is responsible of measuring the metrics of the web services. A normal user will interact directly with Analyzer service, which will in turn, communicate with the appropriate monitor.

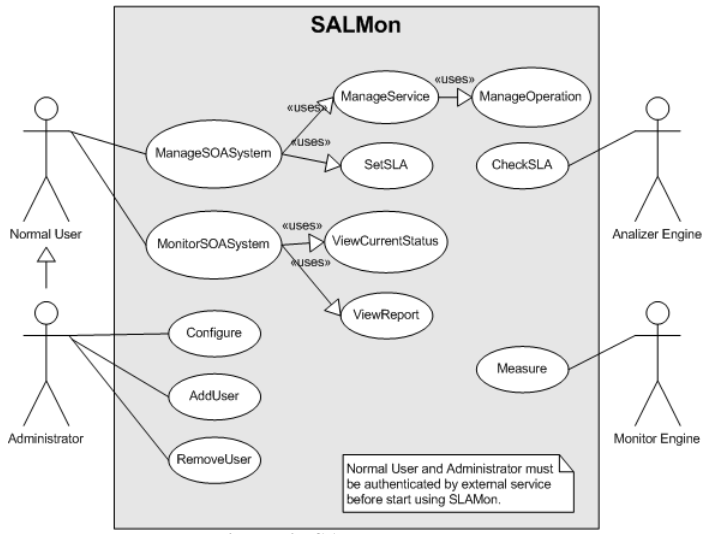

Figure 3: SALMon use cases

3.6 Data Model

In Figure 4 we show the data model of SALMon. To build this data model we have extracted some ideas from the State of the Art report of monitoring web services developed by the NESSI group [6]

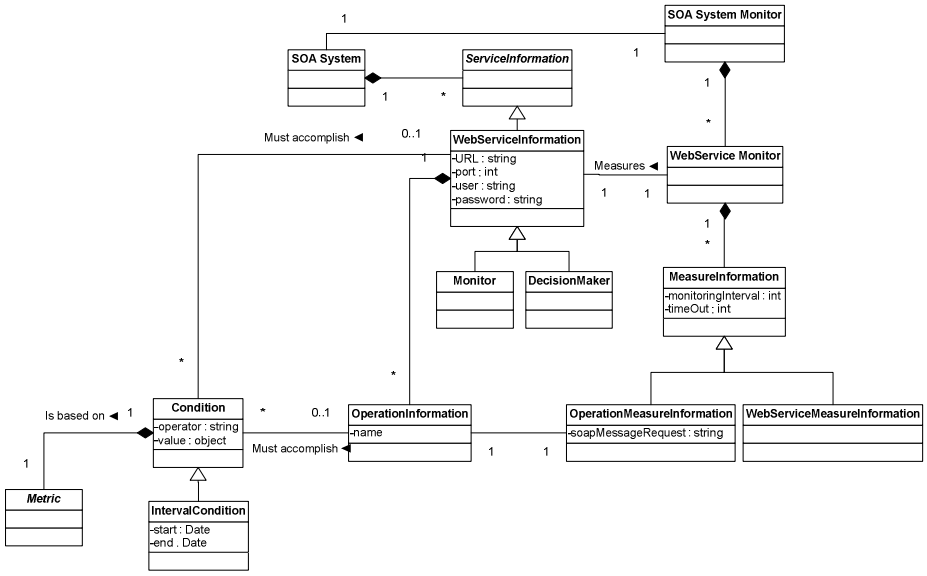

Figure 4: Data Model of SALMon

4. Conclusions

In the context of SOA systems, dynamic changes are needed in order to keep fulfilling the QoS requirements stated in SLAs. We have presented SALMon, a SOA System which is able to monitor services, check their SLA and eventually take decisions in order to support self-adaptation of SOA systems.

The SALMon architecture has been designed to support any kind of services. To do so, SALMon have general interfaces which can be applied to any type of service. However, we are focusing our first implementation in monitoring web services since they are the most common type of service used in a SOA system. As future work we plan to support monitoring of multiple types of services using the same monitor with different kinds of Measure Instruments, so we will be able to monitor an entire heterogeneous SOA system.

We have identified that some metrics are easy to monitor (e.g., response time), but others (e.g., accuracy) need a more complex method to measure them. In this area, we are focusing in how to determine accuracy metrics precisely. Our approach presented is using patterns of correct responses.

Finally, our current monitoring strategy can be labeled as active measurement, it means that we are establishing a connection to the monitored service. This method has its benefits but it is not always the best choice because it could interfere with the obtained QoS measurements, for this reason we plan to build measure instruments capable to work according to conservative strategies which won't need to establish connections but require to be placed nearer in the client or the service network.

Acknowledgements

This work has been supported by the research projects ADICT, TIN2007-64753, MCyT, Spain, and SODA FIT-340000-2006-312. Marc Oriol has a FPI grant bound to the project TIN2007-64753.

References

[1] D. Ameller, X. Franch. "Service Level Agreement Monitor (SALMon)". Composition-Based Software Systems, 2008. ICCBSS 2008.
[2] International Organization for Standarization. ISO Standard 8402: Quality management and quality assurance-Vocabulary, 1986.
[3] M.P. Papazoglou. "Service-Oriented Computing: Concepts, Characteristics and Directions". In Proceedings of the Fourth International Conference on Web Information Systems Engineering, p.3, December 10-12, 2003.
[4] International Organization for Standarization. ISO/IEC Standard 9126: Software Engineering – Product Quality, part 1. 2001.
[5] J. P. Carvallo, X. Franch, C. Quer. "Managing Non-Technical Requirements in COTS Components Selection". RE 2006.
[6] NESSI Group. "NESSI Open Framework – Reference Architecture, State of the art report". 2008

A quality model for service monitoring and adaptation *

Cinzia Cappiello[1], Kyriakos Kritikos[1], Andreas Metzger[2], Michael Parkin[4], Barbara Pernici[1], Pierluigi Plebani[1], and Martin Treiber[3]

[1] Politecnico di Milano – Dipartimento di Elettronica e Informazione, Italy
{kritikos, pernici, cappiell, plebani}@elet.polimi.it
[2] University of Duisburg-Essen – Software Systems Engineering, Germany
Andreas.Metzger@sse.uni-due.de
[3] Vienna University of Technology – Distributed Systems Group, Austria
m.treiber@infosys.tuwien.ac.at
[4] Tilburg University – Department of Information Systems and Management, Austria
m.s.parkin@uvt.nl

Abstract. Adaptive Service Based Applications (SBAs) can become a reality with the advent of sophisticated monitoring and adaptation mechanisms. In this paper, the main focus is on defining quality and how it can be exploited by these monitoring and adaptation mechanisms. To this end, we propose a quality model for SBAs and their infrastructure, new techniques for predicting quality and different types of quality-based adaptation actions for SBAs.

1 Introduction

Based on the Service Oriented Architecture (SOA), Service Based Applications (SBAs) can be built from simple or complex services based on functional and non-functional requirements usually provided by a user. This construction of SBAs is usually performed either at design-time or run-time with the help of a service composition engine. As it is already known, the design-time construction of SBAs has the limitation that the constructed complex service can fail in many ways: one of its main component services may not be available as it has reached its capacity limit or it has produced erroneous output or it has failed or the network connecting this service with the outer world is not available. For the above reasons, the run-time construction of SBAs is preferred as it can solve the above problems, for instance, by substituting the faulty service with another one offering the same functional and similar quality capabilities of the faulty one. However, substituting a faulty service with a new one does not always solve the problem. Sometimes it is preferable to re-execute the faulty service with the same or new input parameters or to compensate this service with a compensation action defined within the service management interface in case we are talking about transactional services. Moreover, in case the faulty service is substituted with a new one, maybe the remaining execution plan has to be changed in order to still satisfy the user functional and quality requirements or violate the quality requirements in the smallest possible way.

*The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube) and the Italian FIRB Project TEKNE

1.1 Quality

QoS research gained a lot of attention during the last years in the field of Service Oriented Computing (SOC). This is due to the role that QoS can play in the life-cycle of services [1]. In particular, QoS can be used for filtering and selecting between functionally equivalent services [1], for the design time and runtime selection of component services of a composite service and for adapting services when their promised service level is violated [2]. As QoS is actually a set of non-functional properties, several QoS taxonomies along with their QoS attributes have been proposed. Sabata et. al. [3] present a taxonomy for the specification of QoS in distributed systems. In their approach, the taxonomy is a hierarchical structure that is divided into two major classifications: *metrics* and *policies*. Metrics such as performance (divided into *timeliness*, *precision*, and *accuracy*) measure quantifiable QoS attributes. Policies provide strategies to cope with changing situations and define renegotiation strategies. The work presented in [4] proposes a quality extension to UDDI that encapsulates QoS information which distinguishes four basic classes of QoS attributes, namely *runtime related* QoS, *transaction support related* QoS, *configuration management and cost related* QoS and *security related* QoS. Truong et. al. [5] introduce a framework for monitoring Grid services with respect to QoS attributes. The authors define an extensive, hierarchical QoS classification schema that consists of four main categories, namely *cost*, *dependability*, *configuration* and *performance*.

Basic sets of QoS attributes are discussed in several papers. In [6] a basic set of QoS attributes include *availability*, *accessibility*, *integrity*, *performance*, *reliability*, *regulatory*, and *security*. Zeng. et. al. [7] introduce five major quality criteria for atomic services: *execution price*, *execution duration*, *reputation*, *reliability*, and *availability*. The authors use these criteria in a linear programming approach to select optimal execution plans for composite services. In PAWS [2], the needed functionalities for selecting services, negotiating and maintaining quality through adaptation are discussed.

As it can be seen from the above analysis, there are many service quality models that focus on SOC or Grid Computing but there is no standard and rich service quality model that can be used across multiple scientific disciplines. Moreover, all quality-based service composition and adaptation approaches, that take advantage of these quality models, stay only at the service level and neglect the infrastructural one, while they also perform re-active adaptation trying to repair the problem after it is created and sensed.

1.2 Self-Adaptation

In addition to run-time responses to failures of the service-based application (see above), the highly dynamic environment under which services operate imposes new challenges for engineering and provisioning SBAs. SBAs have to be *flexible* and *adaptable*. By *flexible* we mean that the SBA should be able to change its behavior according to variable execution contexts, while by *adaptable* we mean that the SBA should be able to execute even if the conditions at runtime differ from those assumed during the SBA's initial design. Flexibility can be achieved if SBAs are designed in such a way that are able to self-adapt to timely-respond to changes in their context or their constituent services or the user preferences and context. A necessary condition for achieving this property is

that the SBA itself or another application can detect these changes and deliver them (in the second case) to the SBA. Adaptability can be achieved by equipping the SBA with self-healing mechanisms that enable it to detect or even predict system failures and to react on them with adaptation actions that compensate for deviations in functionality or quality.

1.3 Contribution of paper

This paper focuses on the quality aspect of SBA monitoring and adaptation. To this end, Section 2 deals with defining a quality model for SBAs. This quality model actually defines a rich set of quality attributes that can be quantified with specific metrics used to measure the quality capabilities of the SBA and consequently to detect if these capabilities deviate from the user specified quality requirements.

Quality violations are usually detected by monitoring the SBA, which offers a reactive way of adapting to these violations. In Section 3, after explaining the main drawbacks of reactive adaptation, it is proposed that a proactive approach should be followed by predicting the future quality of the SBA and thus adapting the SBA before the actual deviation takes place.

In Section 4, adaptation is envisioned as a self-healing behavior of a SBA. So adaptation is defined as a general mechanism for allowing a SBA to react proactively or reactively through one or more adaptation actions. Moreover, two types of adaptation actions for quality are defined and analyzed.

2 Quality Modeling

Initially, our reference model consisted of three functional layers: *application/SBA*, *service*, and *infrastructure* layer. However, as an SBA is a complex service, we believe that the set of quality attributes for the SBA and service layer is the same with the only difference that the quality attributes of the SBA level are derived from the quality attributes of the service level. For this reason, we have defined only two quality models: a *Service Quality* and a *Quality of Infrastructure* (QoI) model. It must be noted that a taxonomy of quality attributes is meant by a quality model.

2.1 Service Quality Model

Previous service quality models and taxonomies (see Section 1) considered a small number of quality categories and in each category only some of the representative quality attributes were contained. In our approach, we do not consider only *domain-independent* quality categories and attributes, but also some of the most frequent and general *domain-dependent* ones, where the word *domain* means the application domain of the service. For example, the quality categories of *Data-related*, used for services operating on and/or producing data, and *Quality of Use Context*, used for context-aware adaptive services, contain *domain-dependent* quality attributes. Moreover, we consider quality categories and attributes that are relevant not only for the service and its service

provider but also for the service requestor. For example, the *dependability* quality category is more important for the service provider than the requestor while the *usability* quality category is more important for the service requestor than the provider. Thus, we take into account both the *service provider* and *service requestor views*. In addition, we distinguish between quality attributes that can be measured objectively and subjectively [8]. The former are called QoS attributes and are the typical constituents of Service Level Agreements (SLAs) [9] (e.g *response time* and *availability*), while the latter are called Quality of Experience (QoE) attributes and reflect the perception of individuals or groups of service users (e.g *reputation* and *usability*). Finally, in each category there is an extensive list of the most representative quality attributes including not only *atomic* but also *composite* quality attributes aggregated from atomic ones like *response time*, *failure semantics* and *robustness*.

In the remainder of this section, we are going to shortly analyze our service quality model by focusing on each quality category in order to justify why we have included it, explain its purpose and describe some of its representative quality attributes. The graphical representation of the quality model can be seen in Figures 1 and 2.

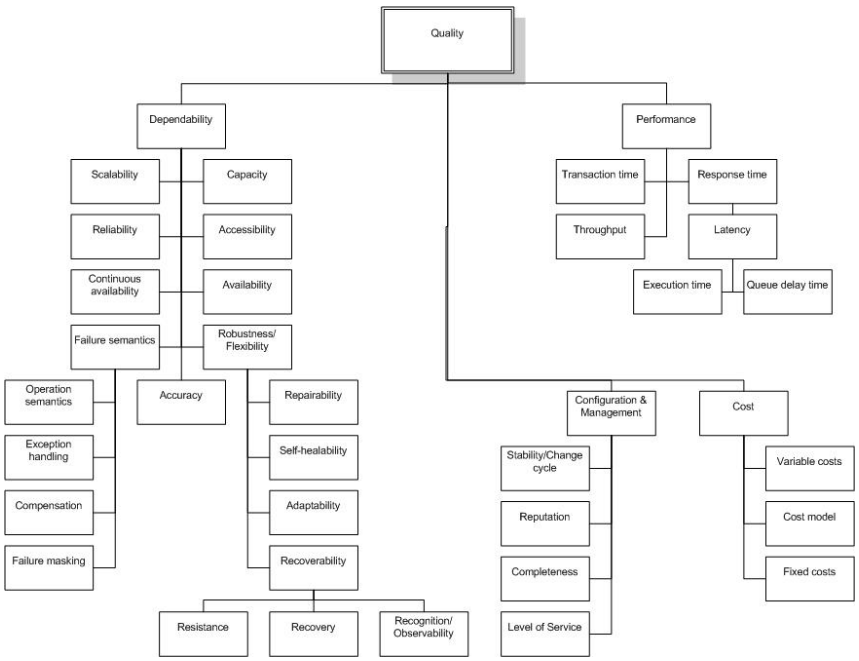

Fig. 1. Service Quality Model (first part)

Performance The *Performance* quality category contains quality attributes that characterize how well a service performs. Two quality attributes with a very well defined meaning are common among all research approaches: *response time* and *through-*

put. In our quality model, *response time* is regarded as a composite quality attribute computed from *latency* and *network delay*. Similarly, *latency* is composite and is computed from *execution time* and *queue delay time*. Finally, a quality attribute that has similar meaning with *execution time* is *transaction time* but is used in a different context (transactional services).

Dependability *Dependability* of a computing system is the ability to deliver services that can justifiably be trusted [10]. In the work of Avizienis et. al. [10], the phrase "justifiably trusted" is translated into three different quality attributes and views: *availability*, *reliability*, and *security*. In our opinion, *security* is orthogonal to *dependability* and it must be put in a separate category because it provides the mechanisms that can possibly avoid specific types of failures from happening, but it has nothing to do with the way the service has been designed and built (with respect to its proper functioning). Moreover, security mechanisms can be bypassed so even these faults cannot be prevented. Thus, we believe that dependability contains *availability*, *reliability*, *failure semantics* [11] and *robustness* [4, 12, 1] with the latter two attributes describing respectively: a) the type of faults that can be exposed and how the service reacts to them and b) the capability of the service to behave in an acceptable way when these faults happen. Another important remark is that besides *availability*, another quality attribute with similar meaning that should be added in this quality group is *accessibility* [5] as it can characterize the case where a service is available, but not accessible to some users, e.g. due to network connection problems.

Security Services should be provided with the required *security*. With the increase in the use of services which are delivered over the public Internet, there is a growing concern about security. The service provider may apply different approaches and levels of providing security policy depending on the service requestor. Security for services [13, 4, 1] means providing *authentication*, *authorization*, *confidentiality*, *traceability/auditability*, *accountability*, *data encryption*, *privacy* and *non-repudiation*. Besides these classical quality attributes, we have added two more, namely *safety* and *integrity* [10].

Data-related In specific application domains, services do not only accept input parameters, but also input data (i.e. files) and they may also produce output data. For example, a *credit card* service can accept as input a data file describing the user's credit card information and can produce as output a data file describing details of the transaction executed based on the functionality of the service. These input/output data are characterized by quality attributes that have been traditionally used in the information and data quality domains like *accuracy* and *timeliness* [12]. The set of these quality attributes is usually called Information Quality (IQ). Except from IQ attributes, we have added two more attributes that characterize the way the service behaves with respect to the data it operates on or produces when it fails (*data policy*) and the degree of validity of the data (*data integrity* [12]).

Configuration Management This quality group/category contains quality attributes that influence the way a service is configured to function (*service level* [3]) or characterize if the promised functional and quality level has been actually delivered during the service's lifetime period (*completeness*, *stability*, *reputation*).

Usability Usability collects all those quality attributes that can be measured subjectively according to user feedback. It refers to the ease with which a user can learn to operate, prepare input for, and interpret the output of the service. This quality group contains three composite (that can be further decomposed) and two atomic quality attributes. **Quality of Use Context** Services can become adaptive if they can change their

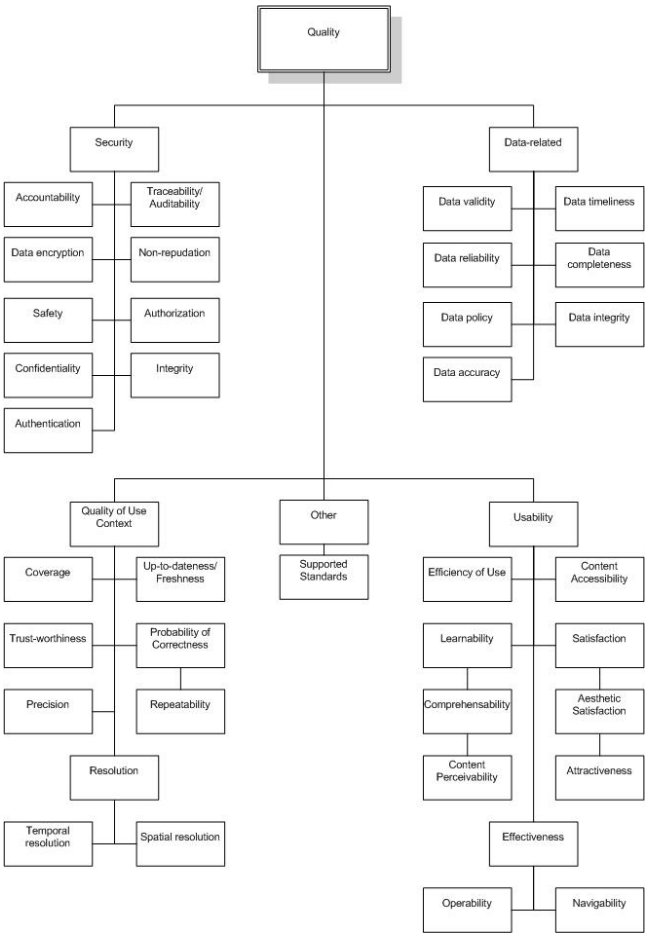

Fig. 2. Service Quality Model (second part)

configuration, behavior and appearance based on the context, where "context is any information than can characterize the situation of the entity. An entity is a person, place or object that is considered relevant to the interaction of a user and an application including the user and the application themselves" [14]. So based on this definition, which is quite general, context is any information that characterizes the service and its user, their

physical and execution environments (including the devices used) and the network that connects them. Context information has also quality [15–17] as it depends on the way it is sensed or derived, the time that it is produced and delivered, the level of detail and other factors. Thus, adaptive services should be designed and executed taking also into account the quality of the context that is delivered to them so as to be able to make rational and realistic decisions when to adapt and how. After reviewing the related literature in quality of context [15–17], we have identified seven quality attributes from which the most important ones are: *precision* (how precise is the information), *resolution* (the level of detail), *probability of correctness* and *freshness* (age of the information).

Cost Some research approaches consider cost as a service attribute that is orthogonal to the service quality because it is related to both functional and non-functional service attributes. However, the majority of research approaches [4, 18, 5, 12, 1] consider cost as a service quality attribute. In addition, all research approaches use cost at the service selection phase in order to select the best service according to its QoS and cost and user's preferences and budget. Based on the above reasons, we regard cost as a (composite) quality attribute (and group) consisting of three (atomic) service attributes: *cost model*, *fixed costs*, and *variable costs*. Actually, cost can be computed either from all atomic cost attributes or only from the *fixed costs* attribute.

Other This quality category has been created to contain various quality attributes of services that do not belong to any of the above categories. So the contained quality attributes may not be related to each other. For the time being, only one quality attribute has been considered called, *supported standards* [6, 13, 4, 5, 12, 1], used to indicate if the service complies with standards or not. This attribute can affect the portability of the service and its inter-operability with other services or applications.

2.2 Quality of Infrastructure Model

The quality of the infrastructure underpinning the service layer has a direct effect on the qualities that can be offered through the service layer. This section defines the qualities of the infrastructure layer that are and can be offered by an infrastructure provider, grouped in eight categories.

Performance The performance of an infrastructure is the rate it can complete units of work (or tasks). *Best efforts* performance is when no guarantees are placed on when a task can be completed for. Usually this is due to the following reasons: first, infrastructure often operates under a high utilisation policy rather than minimising task turnaround time. Secondly, the task may be code that has not been benchmarked on the infrastructure, therefore the execution time of the application cannot be guaranteed. Fourth, even with 'standard' applications the execution time can vary depending on the input parameters given to the application. Finally, it is common for infrastructure not to commit a queueing time for a task. Also, throughput is another measure of performance. But, because of the often heterogeneous nature of tasks executed on a particular infrastructure it is difficult to provide a meaningful figure for the amount of completed jobs per unit time as this will vary depending according to the types of jobs completed.

Newer middleware, such as that provided by the EC FP6 projects Akogrimo, AssessGrid, BREIN, BEinGRID, NextGrid and TrustCOM, can offer *guaranteed performance* through two techniques. The first allows the advance reservation of a period of

time on a resource and the user to submit the job a before the period starts. The second technique is to vary the amount of resources allocated to a computational task. For example, a task can be launched in a virtual machine (VM) which is monitored and, if the job is not meeting the agreed level of performance, more system resources are allocated to the VM so the job completes in a shorter time.

Dependability Infrastructure monitoring observes the status and availability of the infrastructure. The results of monitoring, such as capacity, the uptime, downtime and reliability show how dependable the infrastructure is. The monitoring system can also have its own QoS, such as the frequency of monitoring and the freshness of data. Middleware can build on top of this information to mine historical information and give the infrastructure a 'confidence' rating. Thus, when a user is looking to submit a job she can collate confidence ratings and use this information when deciding where to submit her job.

Security & Auditing Most (if not all) infrastructures have some form of access control, the first step of which is authenticating users to establish their identity. This is often carried out using public key cryptography offered through a Public Key Infrastructure (PKI) using keys issued by Certificate Authorities (CAs). A CA has a policy about the level of proof a person has to give in order to prove their identity and be issued a certificate containing a key. Certificates issued by different CAs using different policies will therefore have differing levels of assurance as to the entity using it.

Infrastructures can also offer levels of security quality depending on the length of the keys used (as information encrypted with a longer key length is more difficult to decrypt). In addition, another quality of PKI-issued keys is the lifetime of a key, after which it will not be accepted: the shorter the lifetime the more secure is a key as there is less time for it to be compromised before it expires.

After authentication users must be authorised. It is general policy that PKI keys are not shared between users, therefore they provide a strong assertion that the entity using the key is the entity the key was issued to. Thus, non-repudiation is a built-in function of PKIs and is a method of accounting for who did what. A common method of auditing is to analyse the log files produced the infrastructure that record the start and stop times of tasks and which user's identity they were run under.

Logging of tasks submitted to an infrastructure may not be the only data collected by a PKI-enabled infrastructure. For instance, a CA will also log events such as certificate requests from users, acceptances of requests, certificates issued and revocation requests from users. This is to detect any strange behaviour from anyone attempting to breech the infrastructure's security by obtaining multiple or counterfeit certificates.

Data-related Infrastructure can provide facilities to store data. Local data written to disk may have a QoS describing the performance of the storage subsystem (i.e., speed of data read or written). In addition local or remote data can be guaranteed a level of redundancy or robustness through the capabilities of the storage subsystem, such as mirroring or RAID5. On-line data will often be subject to a backup and retention plan. Data stored on scratch disks may not have any retention policy at all, however. Backed-up data may copied to more than one replica and stored off-site for greater protection against a data-centre failure.

Configuration management Infrastructure providers attempt to maintain a stable environment and many have a minimum specification and configuration for each component. Users are informed of changes through mailing lists and on central websites in advance of work being carried out. When changes are performed the upgrades and modifications are documented in change control logs. The presence of clear change management procedures can indicate how stable the infrastructure is. However, given the number of nodes an infrastructure can contain it is often the case that some nodes are out-of-step with the current configuration. Thus, infrastructures can have a set of conformation scripts and tests that are run against each node to determine its configuration and send an alert if a node doesn't meet the current specification or configuration.

Cost With the advent of business-oriented infrastructures such as those provided by Akogrimo, BREIN and NextGRID the real cost of providing a service has moved to the forefront of infrastructure quality characteristics as a means of differentiating between services. Many different economic models have been proposed and/or implemented to determine the price of using an infrastructure.

Network and Infrastructure-Related Infrastructure nodes and sites are connected through a network infrastructure. Network paths may also have QoS guarantees associated with them. Circuit-switched networks, like ATM (Asynchronous Transfer Mode) technology, can provide guaranteed and predictable data transfer rates because each circuit is dedicated. However, in packet-switched networks it is typical not to guarantee the amount of bandwidth or the latency of a link as these may vary during the period it is being used because of other traffic using the same infrastructure. Therefore, QoS are usually are given as a guideline from aggregated historical information. For example, providers will usually state a network link can provide a bandwidth of 'up to 2Mb/s', with no guarantee this will be met.

Usability Usability is a quality not often stated as a requirement within infrastructure. For example, when interacting with a Grid, users are expected to have some knowledge of the UNIX operating system, the middleware the Grid is using and how a PKI infrastructure operate. Some relief for non-technical users has come through infrastructure portals. These portals increase the usability of infrastructure for users as they give a web-style working environment and insulate them from the workings of the middleware. Because of their ease-of-use, portals play an important role in interoperation as well as usability as different infrastructure may be available from a portal with a consistent user interface.

3 Quality prediction

Service-based applications (SBAs) operate in highly dynamic and flexible contexts. Those applications should therefore be able to self-adapt to timely respond to changes in their context or their constituent services, as well as to compensate for deviations in functionality or quality. Currently, such a self-adaptation often happens after a change or a deviation has occurred. Yet, such reactive adaptations have the following drawbacks (cf. [19]):

- Executing faulty services can lead to unsatisfied users, can result in loss of money (e.g., wrong bank transactions) and typically requires the execution of additional activities (for instance, compensation actions must be planned / designed).
- Execution of adaptation activities takes time and thereby can reduce the system performance (e.g., response to user input).
- It can take time before problems in the system lead to monitoring events (e.g., time needed for the propagation of events from the infrastructure to the business process level), thus events might arrive so late that an adaptation of the system is not possible anymore (e.g, because the system has entered into an inconsistent or deadlock situation).

As a consequence, SBAs should be able to proactively adapt to prevent the above drawbacks. Key to proactive adaptation is to predict the future quality (and functionality) of a SBA and to proactively respond if the prediction uncovers deviations from expected quality (or functionality).

In this section, we propose different kinds of approaches in order to predict different kinds of quality attributes, specifically QoS and QoE. For what concerns QoS attributes, more traditional approaches can be employed in order to predict the future quality of a service-based application. Section 3.1 sketches two ideas on how existing quality assurance techniques can be exploited. Moreover, this section also analyzes another idea of using benchmarks to predict the QoS of a service.

For what concerns QoE attributes, the advent of the Web 2.0 [20] provides novel tools and platforms which can be exploited. Examples include reputation systems (used in platforms like YouTube, Flickr, or eBay), as well as social bookmarking tools (like Delicious, Digg and StumbleUpon). Those tools and platforms open the door for novel ways of determining and predicting QoE attributes. This is sketched in Section 3.2.

3.1 Predicting Quality of Service Attributes

Quality prediction based on testing: Testing can be considered as a means to measure / assess the quality of a system. As an example, through performance tests, the systems response time under different loads can be measured. In order to extend the traditional testing techniques towards quality prediction, one idea is to use online testing techniques, i.e. to perform testing activities in parallel to the operation of service-based applications (in contrast to offline testing which is done during the design phase). Obviously, an online test can fail; e.g., because a faulty service instance has been invoked during the test. This reveals a potential problem that the service-based application can face in the future of its operation; e.g., when the application invokes the faulty service instance. In [Hielscher et al. 2008] an initial framework for such a use of online testing has been presented.

Quality prediction based on model analysis: Another approach which allows predicting the future quality of a system is to exploit models to reason on QoS attributes during the run-time of an SBA. As an example, the approach presented in [21] uses models, which are continuously updated during the operation of the SBA, in order to predict violations in the future execution (states) of the system.

Quality prediction based on benchmarks Another possible strategy is to use benchmarks as means to calculate expected performance of services in new environments. As prerequisite server side benchmarks can serve as baseline to establish a metric for the performance of the service environment. Note that these numbers only provide a *rule of thumb* and provide an rough estimation for the expected performance. In this context, one must distinguish between (i) data-centric services that depend on databases (e.g., the provision of business reports) and (ii) services that perform data transformations (e.g., transformation of ASCII text to PDF, etc.) or calculations (e.g., scoring values of companies based on balance sheet data). In the case of data centric services, the major part of the performance depends on the performance of the database. Database benchmarks [5] can be used to establish a performance index for these type of services. Services that do not use databases but do data transformation depend on criteria that can be measured by benchmarks that focus on CPU, memory and I/O performance [6].

3.2 Predicting Quality of Experience Attributes

The Internet is currently evolving towards the ”Web 2.0”, in which social networks, folksonomies, reputation and rating systems play an ever more dominant role [20]. Reputation systems attempt to rate entities (e.g., book, images, videos, etc.) based on a collection of opinions (subjective perception). As examples, Flickr provides individual (subjective) opinions of people about certain photos, the videos provided by YouTube are rated by the viewers of these videos, the participants in eBay are rated based on the experience of previous sales.

Considering the fact that in the future Internet of Services, an abundance of services will be available and accessible over the Web, it is only natural to assume that a rating system for these services will be put in place (cf. [22]). We believe that this fact can be exploited to measure and predict QoE attributes, i.e. to determine quality attributes that are strongly influenced by how the quality of a service has been perceived by the users of that service.

In this respect, we envision exploiting techniques for aggregating individual views into a public opinion (as has been done for the quality attributes trust and reputation; e.g., see [23, 24]). As an example, this approach could be extended and adapted to aggregate the experience of individual users about the usability of a service into a public opinion about the service’s usability. Analyzing the change of this public opinion over time, as an example, could then support pro-active adaptation (see below).

4 Adaptation

The quality model together with the quality prediction techniques from above allow assessing services and thus provide a basis for enforcing actions on the system to guarantee that the users’ requirements are satisfied.

[5] http://www.tpc.org/

[6] http://www.specbench.org/

In general such actions have the goal of guaranteeing the functionalities and quality of the system even when perturbed situations arise. In general, a self-healing behavior is envisioned. Prediction allows acting before actual failures occur in the system (*proactive approach*), while a *reactive approach* based on occurred failures is adopted after failures. In the following, we define adaptation as the general mechanism which allows reacting either in a proactive or a reactive way, as a combination of one or more adaptation actions.

Several aspects have to be considered for adaptation, which are based on implicit or explicit classification of error states in the system. We illustrate in the following the principal aspects that influence decisions:

- persistence of faults: faults originating error states may be persistent, temporary, or intermittent;
- context variability: systems with a variable context require an adaptive behavior more than in the case of stable context situations;
- origin of faults: faults may occur at the application or at the infrastructure level;
- service evolution: services may present more or less well defined and stable interfaces to interacting partners;
- service compositions: service can be used in isolation, or in a service composition, which can be more or less dynamic.

Concerning quality, adaptation actions can be of two main types: negotiation and repair. *Negotiation* actions allow defining or redefining the quality characteristics of interacting services and infrastructure components with a single step or iterative process which allows the definition of new quality thresholds which are acceptable for all participating components in a service composition. Negotiation can be performed at design time, in particular when a set of predefined components participate in a composition, thus preparing the potential components in a composition, as proposed in [2]. In variable context and evolving services a more dynamic approach to negotiation can be envisioned, for instance based on on-line auctions [25].

Repair actions are proposed to repair system errors. At the service level, the main repair actions are *substitution* and *retry*, which may need to be executed in a repair plan to maintain a consistent system state, involving for instance also *compensation* actions [26]. Other actions have been proposed for the infrastructural level, mainly based on reconfiguration of services or infrastructural components. The reasons for faults at the infrastructure (e.g., Grid) level can be manifold; the geographically widespread nature of infrastructures lead them vulnerable to connectivity problems and the variations in the configuration of different systems, resources that have unpredictable behaviour, problems with the connecting infrastructure or systems just running out of consumable resources (i.e., memory or disk space) are some other possible sources of failure. When discussing the repair actions of an infrastructure, how it recovers from one of these faults is an important factor. Thus, automated procedures may be in place to recover from faults. For example, a task may be retried (possibly on an alternate resource) or the task can be restarted from a checkpoint taken before the task failed.

Decision mechanisms for applying repairs on the basis of the aspects listed above have been proposed, including rule based approaches [27] or automatic plan generation [26].

5 Concluding remarks

A quality model of a service/SBA (and its infrastructure) is the first step for defining service quality. The second and more difficult step is to associate the quality categories and attributes with each other modeling in this way their quantitative and qualitative dependencies. This step is essential if we want to be able to derive more information from measurements or to evaluate the correctness of these measurements or of quality predictions. A further step will be a definition of a semantic, rich and extensible quality meta-model that will include all possible concepts and their relationships and inter-dependencies required for defining and monitoring end-to-end quality characteristics and negotiating SLAs.

In adaptive services, the decision for repair is examined only at the service level while decisions at the infrastructural level are taken independently to manage the networked system resources. An interaction to manage quality across levels is being proposed in [28] for energy management. However, the complexity of such decisions might require completely new approaches in future research, since a complete description and control on all system's variables might result impractical. Proposed approaches include automatic classification and learning [29] and the application of systems theory to control the stability of the system.

References

1. Kritikos, K.: QoS-based web service description and discovery. PhD Thesis, Computer Science Department, University of Crete, Heraklion, Crete, Greece (2008)
2. Ardagna, D., Comuzzi, M., Mussi, E., Pernici, B., Plebani, P.: PAWS: A framework for executing adaptive web-service processes. IEEE Software **24**(6) (2007) 39–46
3. Sabata, B., Chatterjee, S., Davis, M., Sydir, J.J., Lawrence, T.F.: Taxomomy of QoS specifications. In: WORDS '97: Proceedings of the 3rd Workshop on Object-Oriented Real-Time Dependable Systems, California, LA, USA, IEEE Computer Society (1997) 100–107
4. Ran, S.: A model for web services discovery with QoS. SIGecom Exch. **4**(1) (2003) 1–10
5. Truong, H.L., Samborski, R., Fahringer, T.: Towards a framework for monitoring and analyzing QoS metrics of Grid Services. In: International Conference on e-Science and Grid Computing, Amsterdam, The Netherlands, IEEE Computer Society (December 2006)
6. Anbazhagan, M., Nagarajan, A.: Understanding quality of service for Web services. IBM Developerworks website (January 2002)
7. Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., Sheng, Q.Z.: Quality driven web services composition. In: WWW '03: Proceedings of the 12th international conference on World Wide Web, Budapest, Hungary, ACM (2003) 411–421
8. van Moorsel, A.: Metrics for the Internet Age: Quality of Experience and Quality of Business. Technical Report HPL-2001-179, HP Labs (2001)
9. Parkin, M., Badia, R., Martrat, J.: A comparison of SLA use in six of the European Commissions FP6 projects. Technical report, TR-0129, Institute on Resource Management and Scheduling, CoreGRID - Network of Excellence (April 2008)
10. Avizienis, A., Laprie, J.C., Randell, B.: Fundamental concepts of dependability. Technical Report 0100, Computer Science Department, University of California, Los Angeles, LA, USA (2001)
11. Kopetz, H.: Real-Time Systems: Design Principles for Distributed Embedded Applications. Kluwer Academic Publishers (1997)

12. Cappiello, C.: The Quality Registry. In: Mobile Information Systems – Infrastructure and Design for Adaptivity and Flexibility. Springer-Verlag (2006) 307–317
13. Lee, K., Jeon, J., Lee, W., Jeong, S.H., Park, S.W.: QoS for Web Services: Requirements and possible approaches. W3C note (November 2003)
14. Dey, A.: Architectural support for building context-aware applications. PhD Thesis, College of Computing, Georgia Institute of Technology (December 2000)
15. Gray, P.D., Salber, D.: Modelling and using sensed context information in the design of interactive applications. In: EHCI '01: Proceedings of the 8th IFIP International Conference on Engineering for Human-Computer Interaction, Toronto, Canada, Springer-Verlag (2001) 317–336
16. Buchholz, T., Küpper, A., Schiffers, M.: Quality of Context: What it is and why we need it. In: 10th International Workshop of the HP OpenView University Association (HPOVUA 2003), Geneva, Switzerland. (2003)
17. Sheikh, K., Wegdam, M., van Sinderen, M.J.: Quality-of-Context and its use for protecting privacy in context aware systems. Journal of Software **3**(3) (March 2008) 83–93
18. Cardoso, J., Sheth, A.P., Miller, J.A., Arnold, J., Kochut, K.: Quality of service for workflows and web service processes. Journal of Web Semantics **1**(3) (2004) 281–308
19. Hielscher, J., Kazhamiakin, R., Metzger, A., Pistore, M.: A framework for proactive self-adaptation of service-based applications based on online testing. In: ServiceWave 2008, to be published (10-13 December 2008)
20. Andersen, P.: What is Web 2.0?: ideas, technologies and implications for education. JISC Report (2007)
21. Gallotti, S., Ghezzi, C., Mirandola, R., Tamburrelli, G.: Quality prediction of service compositions through probabilistic model checking. In: 4th International Conference on the Quality of Software-Architectures (QoSA 2008). Volume 5281 of LNCS., Karlsruhe, Germany, Springer (2008) 119–134
22. Laforenza, D., Nardini, F.M., Silvestri, F.: Collaborative ranking of grid-enabled workflow service providers. In: HPDC '08: Proceedings of the 17th international symposium on High performance distributed computing, New York, NY, USA, ACM (2008) 227–228
23. Drago, M.L.: A reputation management framework for web services. Master's thesis, Politecnico di milano (2008)
24. Bianculli, D., Binder, W., Drago, M.L., Ghezzi, C.: Transparent reputation management for composite Web Services. In: International Conference on Web Service (ICWS). (2008)
25. Mohabey, M., Narahari, Y., Mallick, S., Suresh, P., Subrahmanya, S.: A combinatorial procurement auction for qos-aware web services composition. CASE 2007: IEEE International Conferenc on Automation Science and Engineering (Sept. 2007) 716–721
26. WS-Diamond team: DIAMOND Web Services - DIAgnosability, MONitoring and Diagnosis. In: At your service: An overview of results of projects in the field of service engineering of the IST program MIT Press Series on Information Systems, MIT Book). (in press)
27. Baresi, L., Guinea, S.: A dynamic and reactive approach to the supervision of BPEL processes. In: ISEC '08: Proceedings of the 1st conference on India software engineering conference, New York, NY, USA, ACM (2008) 39–48
28. Ardagna, D., Cappiello, C., Lovera, M., Pernici, B., Tanelli, M.: Active energy-aware management of business-process based applications. In: ServiceWave. (2008)
29. Pernici, B., Rosati, A.M.: Automatic learning of repair strategies for web services. In: ECOWS 2007: Fifth IEEE European Conference on Web Services, Halle (Salle), Germany (2007) 119–128

Online Testing, Requirements Engineering and Service Faults as Drivers for Adapting Service Compositions

Andreas Gehlert[1], Julia Hielscher[1], Olha Danylevych[2], Dimka Karastoyanova[2]

[1]University of Duisburg-Essen, Schuetzenbahn 70, 45117 Essen, Germany
{andreas.gehlert | julia.hielscher}@sse-uni-due.de
[2] IAAS, University of Stuttgart, Universitaetsstr. 38, 70569 Stuttgart, Germany
{olha.danylevych | dimka.karastoyanova}@iaas.uni-stuttgart.de

Abstract. Adaptability is a key feature of service-based applications (SBAs). Multiple approaches for adaptability, including those borrowed from the traditional workflow technology, can be used to react to various types of changes in the SBA's environment. Unlike previous fragmented research, we aim at presenting a unified view reflecting the convergence of approaches from requirements engineering, online testing and adaptation mechanisms for service compositions. The main result of our approach is that a dynamic binding strategy known from service composition research leads to an interaction of the requirements engineering and online testing activities with an enterprise service registry only and, therefore, to a loose coupling between the three activities.

Keywords: Service Composition, Adaptability, Requirements Engineering, Online Testing, Self-optimization, Web Services

1 Introduction

The software development life-cycle for service-based applications (SBAs) usually has a very short design and testing phase [1, p. 46]. This modification of traditional software life cycles is due to two facts: First, the increased importance of software systems for the success of a company implies that software systems are developed more rapidly to shorten the time to market and, second, companies rely on the inherent flexibility of SBAs that facilitates their runtime adaptation according to the current service provision, changes in (legal) regulations, systems support, and according to existing and/or new requirements. This flexibility allows reducing the time needed for analysing, designing, deploying and testing SBAs since testing as well as eliciting new requirements can be postponed to the runtime phase of the SBA.

In this paper we show how the adaptability feature of SBAs changes the way in which SBAs are developed and tested. Unlike previous approaches, which focus on technical mechanisms needed to enable adaptation, we focus on the integration of three different research streams, namely the adaptability of service compositions,

online testing and requirements engineering. We aim at showing how techniques from these three areas may be intertwined.

Service-based applications are typically implementing a business process. The accepted approach for implementing business processes in a service-based environment is the workflow-based approach. Workflows using services are also called service compositions and are described in terms of control flow (tasks and their ordering), data flow (data exchanges between tasks and with services), exception handling and the service implementations that realise the individual tasks. Therefore, service composition descriptions are organized in two dimensions – control logic and functionality. For simplicity we assume that an SBA is a service composition and, hence, use both terms synonymously in this paper.

The mechanisms enabling adaptability of service compositions with respect to their functions dimension are currently restricted to anticipating service failures. Consequently, these mechanisms are not designed to react to online testing results and to newly available services, which are discovered by the requirements engineer in a continuous requirements engineering process.

This paper aims to bridge this gap and to extend current adaptation mechanism of service compositions to three adaptation drivers: service failure (traditional adaptation technique), online testing results achieved during the runtime of the SBAs and new available services discovered in a continuous requirements engineering process.

The paper is structured as follows: In section 2 we present an overview of the approach to SBA adaptability that unifies the view of three fields of research, namely requirements engineering, online testing and adaptation of service compositions. In section 3 we identify the drivers for adaptability of SBAs, which are closely linked to our adaptation approach. This integration is then demonstrated with the help of an example in section 4 and the conclusions are summarised in section 5.

2 Integrating RE, Online Testing and Service Composition Adaptation Techniques

The aim of this section is twofold: First, we describe the rationale of our approach by introducing motivating scenarios and, second, we describe the adaptation of service compositions, requirements engineering and online testing in more detail. The approach is outlined in Figure 1.

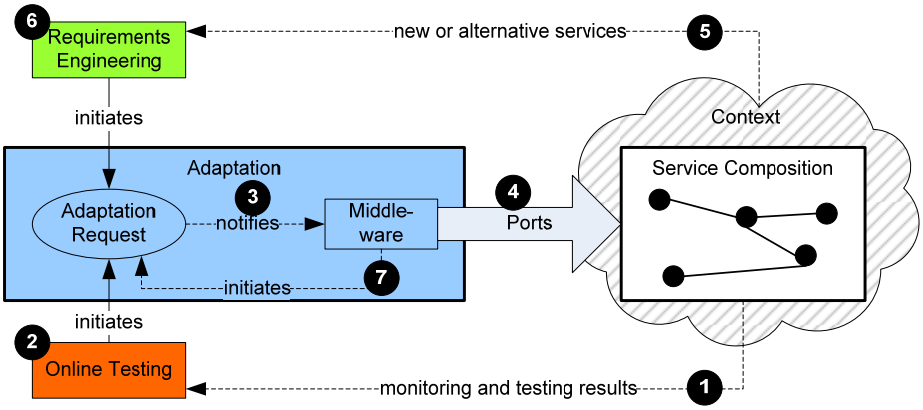

Figure 1: Integration of Online Testing, Requirements Engineering and Service Composition Adaptation Techniques

To illustrate our approach, we use the wine production example introduced in [2, p. 330] (cf. Figure 2 for the wine production process). The wine delivery process is described by six activities. The process starts with buying wine grapes (task "Buy Grapes"). After this task, the production (task "Produce wine") is carried out. Afterwards the process might proceed in two different ways. The first way is storing the wine in oak barrels to mature (task "Store"), which may be followed by the shipment task ("ship wine") or by finishing the production process (task "Wine lot production finished"). The alternative is to ship the wine immediately (task "Ship wine") followed by receiving a delivery note (task "Receive delivery note") and finished by the task "Wine lot production finished".

Assume now the following three scenarios which need three different adaptability drivers:

1. *Scenario 1 - Requirements Engineering*: Assume that the requirements engineer found a new shipping service, which is cheaper than the previously used service in the service composition (❺). If the requirements engineer decides to use this service in the service composition s/he needs to send an adaptation request (❻) to the process engine, which will eventually notify the service middleware (❸). This in turn leads to the usage of the new service in the service composition (❹).
2. *Scenario 2 - Online Testing Scenario*: Assume that online testing finds a failure in the "Buy grapes" service, which would lead to buying red instead of white grapes under some circumstances (❶) before invoking the service for a particular SBA instance. In this case the online testing activity (❷) initiates an adaptation request. This adaptation request leads to a notification of the middleware (❸), which needs to find a new service and to use this service in the service composition (❹).
3. *Scenario 3 - Service Failure Scenario*: Assume that the middleware wants to invoke the "shipment" service, which is currently not available. As a consequence, the middleware sends and adaptation request, which is either handled

by the middleware itself (❹) or by the service compositions (❼). In any case the service composition is adapted to use a new service (❹).

To understand the interplay between requirements engineering, online testing and adaptation, these techniques are described in the following subsections in more detail.

2.1 Requirements Engineering

Requirements engineering (RE) in traditional software engineering process models is carried out as separate activity before the design of the software system. It aims to elicit, document and agree upon the goals, assumptions and requirements of the software system to be [3]. Although RE activities are also an essential part of the engineering process for SBAs, which determine the goals and purposes of the SBA, we argue that requirements engineering should be a continuous process, which covers the entire life cycle of the SBA. The rationale of this argument is the adaptability of the SBA, which allows the requirements engineer to trigger an adaptation of the SBA, e. g. in case of newly available services (scenario 1).

If a new service becomes available the requirements engineer needs to check for two conditions: First, the new service must provide the functionality, which is actually needed by the SBA, i. e. the service must "fit" with the purpose of the SBA. Second, the service should fulfil the requirements better than the previously used service. The second requirement ensures that the new service is superior to existing services. This continuous requirements engineering approach rests on the assumption that new services become available over time.

The algorithms needed for this continuous requirements engineering process are described in [4, 5]. The main idea of these approaches can be described as follows: The requirements of a SBA are described as Tropos goal models [6, 7]. The rationale for using Tropos is threefold: First, Tropos is a comprehensive approach to develop software systems from early requirements to their implementation. Second, Tropos was already applied to the service discipline, e. g., it was already shown that it is applicable to SBAs [e. g. 8, 9-12]. Third, Tropos comes with a formalisation which allows analysing the influence of the satisfaction of one goal on the entire goal model [13].

Based on the assumption that services are described with goal models, the requirements engineer has to carry out the following two steps once a new service becomes available: First, the goal model of the new service should be contained in the goal model of the SBA. This step ensures that the new service provides a functionality needed by the SBA. Second, once this matchmaking is done, the goal satisfaction for all hard- and soft-goals can be calculated based on TROPOS's goal propagation algorithm. The result of this calculation is threefold: First, the goal satisfaction rates of some goals are higher while the goal satisfaction rates for the remaining goals remain unchanged. In this case the new service is superior to the existing one and it should be used (pareto principle). Second the goal satisfaction rates of some goals are lower then before and remain unchanged for all other goals. In this case the service is inferior to the previous one and it should not be used. Third, the goal satisfaction rates are higher for some goals and lower for some other goals. In this case the requirements

engineer needs to decide whether to use the new service based on the priority of the different goals.

This goal driven approach to SBA leads to perfective maintenance known from software engineering [14, p. 493] or to so called self optimization in the service world [1, p. 43]. The interaction of this approach with the adaptation of the service composition is described in section 3.

2.2 Online Testing

In traditional software development processes quality assurance including testing is almost finished when the runtime of a system starts. The loose coupling feature of SBAs, however, allows to easily adapting the SBA during runtime so that the testing phase and the runtime phase of an SBA may overlap. This means that some testing activities are purposely postponed until the SBA is up and running [15, p. 40]. The consequence of this overlap is that the SBA may be modified once an error is detected during the runtime quality assurance – in particular online testing – activities.

Similar to traditional software systems, two different inputs can be considered to assure the quality of SBAs: monitoring information and testing outputs. In contrast to traditional software development the comparison of expected and actual behaviour and possible following adaptations has to be done at runtime. Consequently, not only monitoring techniques but also online testing techniques are relevant to assure the quality of the entire SBA. However, due to space limitation, we only concentrate on online testing in this paper.

Hielscher et al. present the *PROSA* (*PRO*-active *S*elf-*A*daptation) framework in [16], which enables online testing in addition to traditional runtime quality assurance methods such as monitoring. The results of the online testing activities are compared with expected results to detect problems, which may be resolved by adaptations. Online testing aims at finding possible problems before they occur in a running instance of the SBA. The underlying assumption of PROSA is that online testing activities do not interfere with the running SBA. This requires that each service, which should be tested, offers a test mode. This test mode prevents the “normal” execution of all activities of an instance of the SBA, which trigger more than just returning a computational result – e. g. it prevents the delivery of physical goods.

Testing service compositions by testing their constituent services requires that the services are known at test time. This means that either the services are statically connected to the tasks of the service composition (see below) or that the registry from which the services are discovered contains a constant set of services during test time. If none of the two previous conditions holds, it follows that online testing must be executed in parallel to each service composition instance since the services used in different service composition instances may differ. This approach, however, requires considerable computational resources and is, therefore, discarded here.

2.3 Adaptation

Adaptations in service compositions can be carried out during design time or during runtime. During design time the process model, which describes control and data flow can be modified in any possible way, including the assignment of service implementa-

tions to tasks or the modification of the control and data flow of the workflow definition ([17]).

During runtime the process model may be altered so as to assign new service implementations to process tasks [e. g. 18] or to modify the control and data flow completely. After these modifications it must be decided whether all running instances should benefit from these modifications (instance migration), whether some of the running instances should benefit from these modifications or whether future instances should use the modifications only. If the modifications target only some instances of a process model, the changes are called ad-hoc changes and are not reflected in the process model.

According to the previous classification of adaptation, we need to distinguish two modification dimensions – e. g. modification of the process model and modification of the assignment of concrete services to process model tasks – and two phases in the SBA lifecycle – e. g. design time and runtime. In this paper, however, we only focus on modifying the assignment of service implementations to process tasks during runtime.

This assignment is based on so called *binding strategies*. According to [20] and [19, p. 536] we can distinguish between two major binding strategies: First, the *static binding strategy* requires that the service implementations are assigned to the activities during design or deployment time. Second, *dynamic binding* requires a declarative description of the requirements of the service at design or deployment time. The actual service implementations are then discovered by the underlying middleware during runtime based on the declaratively specified criteria.

The dynamic binding strategy is the most flexible approach for service binding because it enables the discovery of service implementations during runtime based on requirements for the service selection specified declaratively prior to the process execution. In case of a service implementation failure, the middleware is then able to discover a new service implementation according to the specified requirements. The functional part of the requirements is typically provided by the service composition (e. g. operation and port type) and the non-functional requirements are specified in separate artefacts associated with the composition (e. g. WS-Policy definitions). It is important to note that these requirements for service selection are specified at the latest during the deployment phase of SBA development. This means, that the selection *criteria* even in the most flexible strategy for service binding are fixed during runtime.

3 Adaptability Drivers

Service-based applications that are implemented using the process-based approach can be adapted as a reaction to changes in the environment or according to the occurrence of exceptional situations using the approaches for process adaptability described in the previous section. Adaptation of the SBAs can be thus triggered by the following drivers:

1. *Recommendations from the requirements engineer*: Typically, enterprises have contract relationships with other business partners. This fact is reflected in the set of service implementations that may be used in service compositions. These partner service implementations usually meet the requirements specified by the requirements engineers in the enterprise. In some cases however, due to the dynamic nature of the service market, new relationships are established with other (previously unknown) partners. If the newly introduced service is better and/or more appropriate (e. g. cheaper, faster etc.), the requirements engineer is entitled to recommend the use of this new service (adaptation trigger).
2. *Results from online tests*: Online tests of service implementations are performed during the runtime of the SBA. If a test of a service fails the online testing expert may recommend to discontinue using the service (adaptation trigger).
3. *Service failures*: During the execution of process instances the discovery, selection and invocation of a service is delegated to the service middleware (Note that in the case of static binding strategy the discovery and selection steps are not needed). The middleware is responsible for tackling the situation in which selected services exhibit failures when invoked. In case the middleware cannot discover any service compliant with the requirements provided by the process model and the process execution environment, and as specified by the requirements engineer, the service selection criteria may be adapted. The new alternative criteria are then used to discover and select another service that can perform on behalf of the process (adaptation trigger).

All three mechanisms are explained in the next section along with our wine example.

4 Example

The presentation of the mechanisms enabling the reaction to the adaptability drivers presented in the previous section will be demonstrated with the help of the example introduced in section 1. Figure 2 depicts the process described by the example. The figure also shows the available service implementations assigned to the tasks either by static or dynamic binding – e. g. the “Grapes retailer” service is bound to the “Buy grapes” task while three different shipment services are available to support the “Ship wine” and “Receive delivery note” tasks.

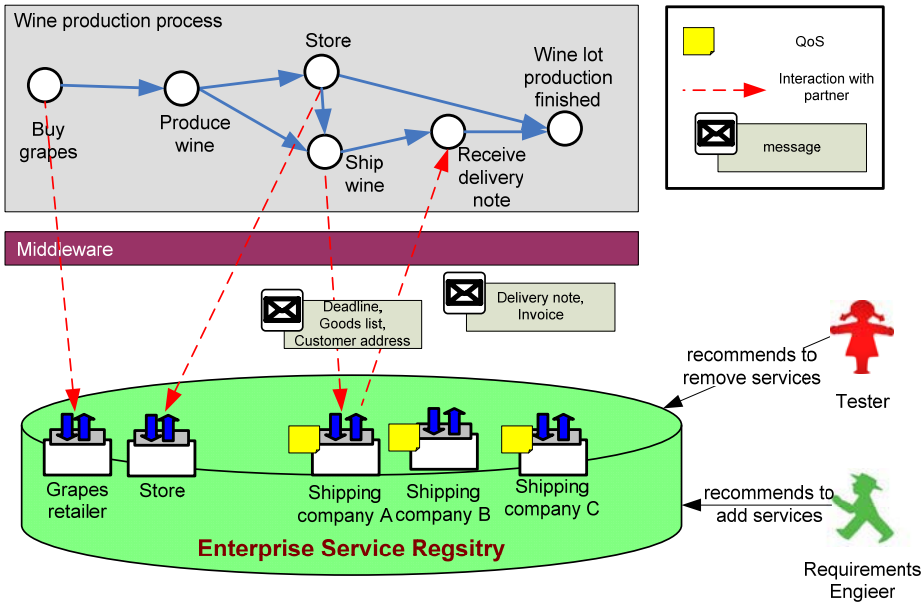

Figure 2. Example of a Wine Production Process.

The process is executed by a process engine which relies on a service middleware [20]. The middleware supports among others the discovery and invocation of services that the process composes (infrastructure services). If the binding strategy for a task is static, the middleware can only invoke the concrete service implementation as defined in the process definition; the middleware receives the input data for the service and its endpoint reference (EPR [20]). After the invocation of the service implementation the middleware returns the result from the service invocation to the process engine and hence to the process instance. The middleware performs an additional step if the binding strategy is dynamic. Before a service invocation can be carried out a discovery of a set of service implementations compliant with the requirements (provided in the process definition) is triggered and the most appropriate service implementation is selected.

All the adaptation mechanisms listed in the previous section require service invocation. Depending on the binding strategy (static vs. dynamic) service discovery and selection may (dynamic binding) or may not (static binding) be needed. In the rest of this section we present the adaptation approaches that can be triggered as a possible reaction to the adaptation drivers we cover here. All these approaches make use of the infrastructure services supported by the service middleware.

4.1 Adaptation Triggered by the Requirements Engineer

As new services become available outside the enterprise service registry, the requirements engineer needs to decide whether the newly available service should be used. This decision is made according to the mechanism described in subsection 2.3. Once a

decision to use the new service is made, the requirements engineer has two choices to make use of the service in the service composition: Using the static binding strategy, the requirements engineer may exchange the static service reference of one or more tasks directly in the process model and to either migrate all instances, selected instances or no instances to this new process model. If the dynamic binding strategy is used, it is sufficient to register the new service in the service registry. Due to the enhanced service characteristics, the new service implementation will be given preference by the middleware's service discovery component.

Assume that the shipment service of company D becomes available and the requirements engineer decides to use it for the above service composition. In a static binding scenario the requirements engineer initiates the exchange of the static reference for the task "Ship wine" from the shipment service of company A to the shipment service of company D. In a dynamic binding scenario, the shipment service of company D is registered in the enterprise service registry and the middleware will automatically prefer it over the shipment service of companies A–C because of its better suitability to the requirements.

Requirements engineering for service based applications is a relatively new field. Existing approaches cover the description of requirements for SBAs [e. g. 8], the discovery of services using goal models [e. g. 4], the consistency check between goal models and workflows [e. g. 12] and the continuous goal-driven optimization of SBAs [e. g. 5].

4.2 Adaptation Triggered by Online Testing

As described above, online testing may find failures, which require an adaptation. The adaptation action required depends on the binding strategy. If the binding strategy prescribes static binding, the faulty service must be replaced in the process model (usually in the deployment information). After this replacement the process model must be re-deployed and it must be decided whether all running instances (instance migration), selected running instances or all new instances should be migrated to the new process model.

Assume that the shipment service of company A should be tested in our example above. The tester generates test cases with deadlines, lists of goods to be shipped and customer address data as input. Then the output is checked against the expected output. In our scenario, the expected output is a delivery note and an invoice. If the service of shipping company A cannot deliver a delivery note, this failure is reported and an adaptation is triggered. The service of shipping company A is then removed from the process model (static binding) or from the registry (dynamic binding) and replaced by another service, e. g. the shipping service of company B.

Numerous testing techniques exist in the literature. An overview of testing techniques for services is given in [21, pp. 48]. Especially test case generation from existing Business Process Execution Language (BPEL) specifications is described in [22, 23]. These existing approaches have to be adapted for using them in parallel to process instance execution.

4.3 Adaptation Triggered by Service Faults

The third driver for adaptation of service compositions are failures of the composed services. A mechanism for tackling such failures for both statically and dynamically bound services is needed to ensure the completion of the process instance. This mechanism involves discovery and selection of a new service capable of performing the task in the process with similar requirements.

In the dynamic binding strategy it is up to the middleware to discover and invoke a service compliant to both the functional and quality requirements specified in the process definition. If the middleware in not able to discover such a service implementation, an alternative set of (quality) requirements may be specified in the process model and hence used by the middleware for service discovery and selection. In case of a static binding strategy, the middleware does not have to discover a new service and, hence, the service composition must be repaired manually or automatically using an alternative set of requirements to the service.

In the context of the wine production example, consider for instance that shipment company A ships goods for less than 100,000.00 € and within one week. Assume now that the shipment service of company A is temporarily not available. As an alternative service implementation the wine producer might be willing to pay more than originally specified to a shipping company that can comply with the time constraint (e. g. shipment company B). The discovery and invocation of such a service is performed by the middleware provided that the alternative quality requirements are made available to it by the service composition definition.

Adaptation mechanisms for BPEL processes are presented in [18, 24], where the quality requirements are specified as Web Ontology Language for Web Services (OWL-S) service descriptions and encapsulated in WS-Policies. Prototypical implementations of an infrastructure including a BPEL engine and a service bus are also available for both the OWL-S and Web Service Modelling Ontology (WSMO) based approaches.

5 Conclusion

In this paper we integrated requirements engineering, online testing and state-of-the-art adaptation mechanisms for service compositions. The paper shows clearly that the dynamic binding strategy driven by pre-described service requirements is beneficial over the static binding strategy. The dynamic binding strategy resolves automatically all service faults, e. g. due to unavailable service implementations. In addition, requirements engineering and online testing only need to interact with the enterprise service registry and have no other influence on the existing service middleware. While the online testing aims to remove faulty services from the registry, the requirements engineering activity aims to add new and innovative services to it, which lead to a better fulfilment of the SBA's requirements. The static binding strategy, however, requires a modification of the process model and its redeployment.

We are also able to identify future research for the integration of online testing and workflow modelling as well as the integration of requirements engineering with work-

flow modelling. Online testing techniques require a certain sequence of tests, e. g. the tester needs to know, which service implementation to test first. This sequence depends heavily on the workflow(s) in which the service implementation is used. In addition, this paper and the cited online testing techniques cover only service testing. However, it is also important to test the whole service composition, e. g. the workflow itself (integration test). Techniques need to be developed and integrated with adaptation techniques.

For the requirements engineering field, the most predominant problem is the mapping of the requirements to service descriptions, e. g. to OWL-S. This mapping ensures that services identified as superior in the requirements engineering activity, are actually given preference during service discovery carried out by the middleware. In addition, workflow instances may be tailored to the so-called context-factors, e. g. workflows may be adapted to certain users or a certain device. These context-aware workflow adaptations require a tight integration of workflow and requirements engineering research.

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube).

References

1. Kephart, J.O., Chess, D.M.: The Vision of Autonomic Computing. IEEE Computer **36** (2003) 41-50
2. Nitto, E.D., Ghezzi, C., Metzger, A., Papazoglou, M., Pohl, K.: A Journey to Highly Dynamic, Self-Adaptive Service-Based Applications. Automated Software Engineering (2008) 257-402
3. Pohl, K.: Requirements Engineering - Grundlagen, Prinzipien, Techniken. dpunkt (2008)
4. Gehlert, A., Bramsiepe, N., Pohl, K.: Goal-Driven Alignment of Services and Business Requirements. 4th International Workshop on Service-Oriented Computing Consequences for Engineering Requirements (SOCCER 2008), Barcelona, Spain (2008)
5. Gehlert, A., Heuer, A.: Towards Goal-Driven Self Optimisation of Service Based Applications. 1st International Conference of the Future of the Internet of Services (ServiceWave 2008). Springer, Madrid, Spain (2008), 13–24
6. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An Agent-Oriented Software Development Methodology. Autonomous Agents and Multi-Agent Systems **8** (2004) 203–236
7. Castro, J., Kolp, M., Mylopoulos, J.: Towards Requirements-Driven Information Systems Engineering: The Tropos Project. Information Systems **27** (2002) 365–389
8. Aiello, M., Giorgini, P.: Applying the Tropos Methodology for Analysing Web Services Requirements and Reasoning about Qualities. UPGRADE: The European Journal for the Informatics Professional **5** (2004) 20–26

9. Lau, D., Mylopoulos, J.: Designing Web Services with Tropos. International Conference on Web Services (ICWS 2004), San Diego, CA, USA (2004) 306–313
10. Misra, S.C., Misra, S., Woungang, I., Mahanti, P.: Using Tropos to Model Quality of Service for Designing Distributed Systems. International Conference on Advanced Communication Technology (ICACT 2006), Phoenix Park, Gangwon-Do, Republic of Korea (2006) 541–546
11. Penserini, L., Perini, A., Susi, A., Mylopoulos, J.: From Stakeholder Needs to Service Requirements. 2nd International Workshop on Service-Oriented Computing: Consequences for Engineering Requirements (SOCCER 2006), Minneapolis, Minnesota, USA (2006) 8–17
12. Pistore, M., Roveri, M., Busetta, P.: Requirements-Driven Verification of Web Services. Electronic Notes in Theoretical Computer Science **105** (2004) 95–108
13. Giorgini, P., Mylopoulos, J., Nicchiarelli, E., Sebastiani, R.: Formal Reasoning Techniques for Goal Models. Journal on Data Semantics. Springer, Berlin, Heidelberg (2003) 1–20
14. Swanson, E.B.: The Dimensions of Maintenance. International Conference on Software Engineering (ICSE 1976), San Francisco, CA, USA (1976) 492–497
15. Baresi, L., Nitto, E.D., Ghezzi, C.: Toward Open-World Software: Issue and Challenges. IEEE Computer **39** (2006) 36–43
16. Hielscher, J., Kazhamiakin, R., Metzger, A., Pistore, M.: A Framework for Proactive Self-Adaptation of Service-based Applications Based on Online Testing. 1st International Conference of the Future of the Internet of Services (ServiceWave 2008), Madrid, Spain (2008)
17. van der Aalst, W.M.P., Jablonski, S.: Dealing with Workflow Change: Identification of Issues and Solutions. International Journal of Computer Systems Science and Engineering **15** (2000)
18. Karastoyanova, D., Houspanossian, A., Cilia, M., Leymann, F., Buchmann, A.: Extending BPEL for Run Time Adaptability. 9th International Conference on Enterprise Distributed Object Computing (EDOC 2005), Enschede, The Netherlands (2005) 15–26
19. Karastoyanova, D., Leymann, F., Buchmann, A.P.: An Approach to Parameterizing Web Service Flows. In: Benatallah, B., Casati, F., Traverso, P. (eds.): 3rd International Conference on Service Oriented Computing (ICSOC 2005). Springer, Amsterdam, The Netherlands (2005) 533–538
20. Weerawarana, S., Curbera, F., Leymann, F., Ferguson, D.F., Storey, T.: Web Services Platform Architecture: Soap, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More. Prentice Hall (2005)
21. Pernici, B., Metzger, A. (eds.): S-Cube Project Deliverable PO-JRA-1.3.1 - Survey of quality related aspects relevant for SBAs (2008)
22. Lübke, D.: Unit Testing BPEL Compositions. In: Baresi, L., Nitto, E.D. (eds.): Test and Analysis of Web Services. Springer (2007) 149–171
23. Dong, W.-L., Yu, H., Zhang, Y.-B.: Testing BPEL-based Web Service Composition Using High-level Petri Nets. 10th IEEE International Enterprise Distributed Object Computing Conference (2006) 441–444
24. Karastoyanova, D., van Lessen, T., Nitzsche, J., Wetzstein, B., Wutke, D., Leymann, F.: Semantic Service Bus: Architecture and Implementation of a Next Generation Middleware. Proceedings of the 23rd International Conference on Data Engineering Workshops (ICDE 2007), Istanbul, Turkey (2007)

Strategies for Automated Performance Simulation Model Adjustment

Preliminary Results

Martin Pinzger

Telecooperation Department Johannes Kepler University,
Altenberger Str. 69, 4040 Linz, Austria

Abstract. System performance is a key feature nowadays, for this reason a lot of resources are diverted to test systems for adequate performance. Sometimes the performance analysis is done with the help of a simulation model. In this case it is necessary to adjust the parameters of the simulation model to get accurate results.
The task of determining the parameters for a simulation model is a quite challenging and time consuming one. In this work some strategies like a variation of binary search, moving average and derivations of ARMA (autoregressive moving average) are used for automatically adjusting the simulation model parameters until a given accuracy is reached. Furthermore a case study with these strategies is done to get first results on the usability and the strengths of the different strategies.
From these case studies the conclusion can be drawn that in the case of reduced calculation power or a reduced set of information about the system under test (SUT) the moving average strategy is performing best. If the calculation power and the set of information about the SUT aren't the limiting factors the derivations of the ARMA strategies should be chosen. The special strength of the ARMA strategies is in the field of long term analysis and predictions for the SUT, or to express in an other way in the parameter estimation for the simulation model of the SUT.

1 Introduction

The task of adjusting a performance simulation model e.g. queuing-network-model [6], to a system under test (SUT) is quite a demanding task, even for a system analyst. So the idea is to automate the step of adjusting a simulation model to a SUT. For this some demands concerning the environment have to be handled, so that the system gets the required information about the SUT, to automatically determine the parameters for the simulation model. Besides the information about the SUT the system needs a strategy for the simulation model parameter estimation. These strategies, their evaluation and a practical study on a web system are the key points of this paper. The strategies considered in this paper have in common that their goal is to get the best prediction for the next simulation model parameter so that the simulation model will perform as well as possible on new/future requests.

This paper consists of two major parts, on the one hand the part where the system, the strategies and the possibilities are described, and on the other hand the actual concrete study is presented.

The first part is structured in the following way: Section 2 describes the surrounding environment for the evaluation of the different strategies for simulation model parameter estimation also called AWPS - System. Section 3 explains in detail the four strategies for simulation model parameter estimation. An exact definition of the possibilities when a test run is executed, concerning the monitored/recorded data about the SUT and usage of this data is reached.

The second part deals with the actual study. First, the objectives of the study are explained. Then the executed test cases are presented and the reasons for using these test cases are given and the chosen settings for the test run are provided. In the next section the results of the study and the conclusions that can be drawn from it are presented.

2 AWPS - System

The automatic web performance simulation and prediction system (AWPS) [8][9] represents a system capable of automatically creating a web performance simulation and conducting a trend analysis of the SUT. The system has to provide three main functions: data collection, simulation and prediction.

- Data Collection Component: The component monitors the SUT, records relevant information and provides static information about the SUT. These pieces of information are then preprocessed and fed into the simulation and the prediction component. A special functionality which is useful for testing the AWPS System is that the system could be executed on pre-recorded data. This provides the opportunity to evaluate the different strategies of the subcomponents without having to care about the influence of variations in the base data.
- Simulation Component: The aim of the this component is to automatically generate and adjust a simulation model of the SUT. For this task the component has to combine the information provided by the data collection component and apply the extracted knowledge of the SUT to create the simulation model or adjust it [5][10]. As simulation framework several products have been evaluated [2][5][7], currently JSIM is used [1].
- Prediction Component: This component uses the information, about the SUT, provided by the data collection component to accomplish its task. Based on the information about the SUT and by using the simulation component the prediction component generates possible scenarios and creates a longterm trend analysis.

This paper focuses on the simulation component, and sets itself the aim to complete the automation of the simulation model adjustment.

3 Strategies for Parameter Estimation

The base for automating the performance simulation model adjustment is to have a suitable strategy for simulation model parameter determination. The goal of the simulation

model parameter determination is to find the parameter which best fits the simulation model. The defined parameters are supposed to ensure that the simulation model will provide excellent results when confronted with new/future requests. To accomplish this, the algorithms predict the parameters for the simulation model based on recorded data. Hereby the focus is set to determine parameters which suit the latest and the expected new requests best, it is acceptable that this simulation model may not bring the best results for the old recorded requests.

In this section the four different strategies which are evaluated within this study are described. Considering the four strategies, the first three, Binary, AVG and ARMA, use no prior grouping whereas the last strategy ARMA with 60 sec. grouping, pre-groups the data before the process is started.

3.1 Binary Strategy Parameter Estimation

The Binary strategy for parameter estimation works with similar principles as a standard binary search algorithm. In a self repeating process the results of the simulation model and the results from the SUT are compared. If the result of the simulation model is too low the simulation parameter is doubled. In the opposite case, if the result of the simulation model is to high, the value of the simulation parameter is divided by two. The two parameters which have to be determined for this strategy are the initial starting value and the minimal error delta. The initial starting value which is doubled or divided should be set to an approximate value to ensure a quick adjustment, e.g. the first recorded value of the SUT. With the minimal error delta the minimal difference between the results of the SUT and the simulation model, when the simulation parameter should be adapted, is defined.

3.2 AVG Strategy Parameter Estimation

The AVG strategy is an implementation of a moving average method. The key advantages of this strategy are that it is easily implemented, the calculation is fast and can be done in increments. This strategy is defined by only one parameter, the window size. The disadvantage with this parameter is, that depending on the underlying function of the SUT, it should be small or large. For example if the underlying function of the SUT is a constant value with some added noise, a large sliding window will likely provide the best results. In the case of a function of the SUT with a strong drift a small sliding windows will be the better option. Clearly, to define the right size of the sliding window, some preliminary knowledge about the SUT is required, or it must be approximated by the use of statistics.

3.3 ARMA Strategy Parameter Estimation

The ARMA (Auto Regressive Moving Average) [3] strategy is a mathematically costly method used in this study to determine the parameters for the simulation model. The basic methods used to parameterize the ARMA strategy are listed below. These methods are also used in the case of the ARMA with 60 sec. grouping strategy, however the data on which the operations are based are different.

To illustrate the process of parameterization and execution of the ARMA strategy, some simplifications and assumptions concerning the basic data are made. Basically the data upon which the operations are based is assumed to show all the characteristics required. The initial data is stored in the timeSeriesDataIn variable [4][12][11].

1. Calculate the correlation function of the input data

$$CorrelationFunction[timeSeriesDataIn];$$

2. Calculate the partial correlation function of the input data

$$PartialCorrelationFunction[timeSeriesDataIn];$$

3. Calculate the mean and the zero mean time series data

$$timeSeriesDataMean = Mean[timeSeriesDataIn];$$

$$timeSeriesData = timeSeriesDataIn - timeSeriesDataMean;$$

4. Set pMax and qMax based on the outcome of the correlation and partial correlation function

$$pMax = CorrelationFunction[timeSeriesDataIn] < 1.96/sqrt[n];$$

$$qMax = PartialCorrelationFunction[timeSeriesDataIn] < 1.96/sqrt[n];$$

5. Create the ARMA models using the Hannan Rissanen estimate method

$$models = HannanRissanenEstimate[timeSeriesData, 20, pMax, qMax, 1];$$

6. Conditional maximum likelihood to re-estimate the model selection

$$arma11 = ConditionalMLEstimate[timeSeriesData, \#]\&/@Take[models, 1];$$

7. Predict the next values for the simulation model

$$result = BestLinearPredictor[timeSeriesData, arma11, 3][[1]]$$

8. Use the predicted values to calculate an AVG and set this as next parameter for the simulation model.

$$nextVal = Sum[result, 1, 3]/3;$$

3.4 ARMA with 60 sec. Grouping Strategy Parameter Estimation

This variation of the ARMA strategy is based on the same method, only the basic data is altered before it is used. Basically the limitation of predictions done with the ARMA strategy is that it is only possible to predict the next few values. These few values in the ARMA strategy case are representative for the next few seconds. So the idea is to group the data before it is used in the prediction process. In this special case the data of 60 seconds is grouped (AVG calculation) and used, so the time frame for which predictions are made extends itself by the factor of 60. The disadvantage with this variation is that our prediction is based on fewer data points, but this is only a problem during the initial phase of prediction. In short the ARMA with 60 sec. Grouping Strategy is named ARMA G in this paper.

4 Possible Test Case Settings

This section deals with the possible settings for a test run, specifically the settings which deal with the behavior of the strategies for simulation model parameter estimation and their evaluations.

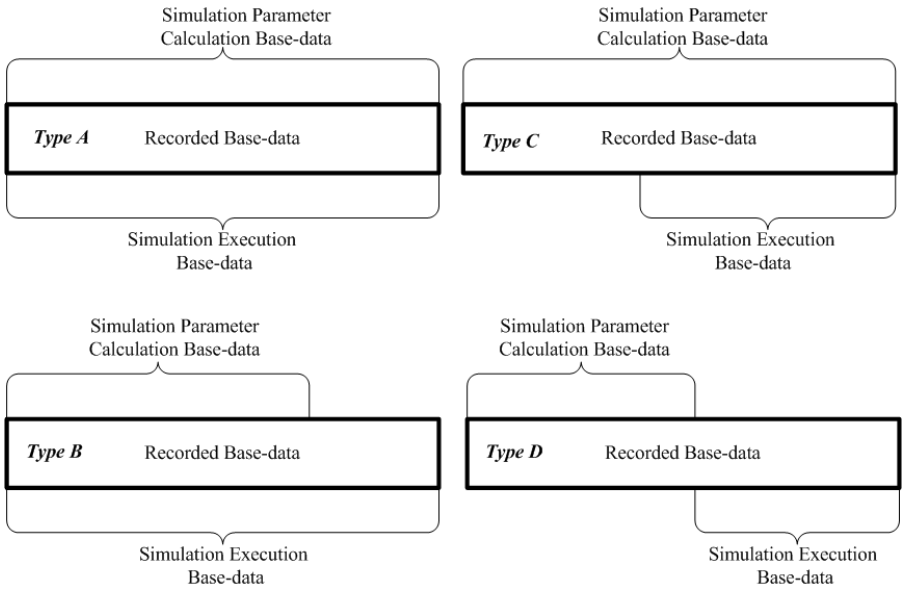

Fig. 1. Types of Base Data Usage

One of the very basic decisions to be made for a test run is the decision whether to execute the test run on current data or to use previous recorded data (for more details see section 2). The next also very basic decision is how this data should be used; in general there are four possible combinations. These four combinations are shown in figure 1, the assigned names are Type A to Type D. In the normal test case, when the strategies for simulation model parameter estimation and the other components of the surrounding system have been evaluated and the focus is back on the analysis and prediction of the SUT, Type A or Type C are best suitable. To get optimal results the simulation model should be as good as possible. But in the current phase the Type D should be used, because it allows the system to learn from ”old” data and have ”new” data for the configured simulation model available to test it.

Other options which could be set for the test run are, after how many new requests the simulation should be executed, how many requests should be saved for evaluating the configured simulation model and what size the sliding window, which is used to keep the runtime of the system in an acceptable range, should be. This sliding window

has only minor influence on the Binary strategy, ARMA and ARMA G; if it is not chosen to small. But the sliding window has a wide influence on the AVG strategy (moving average) as described in subsection 3.2.

5 Objectives of this Study

At the beginning of the second part of this paper, the primary goals of this study will be reviewed. In the introduction and in section 2 it was already stated that the purpose of this work is to build an automatic web performance simulation and prediction system. It is necessary that such a system is able to automatically adjust the parameters of the simulation model. This paper presents strategies that can accomplish this task and tries to evaluate the usefulness of the different strategies for different requirements. These are the questions which should be answered by this study:

- Are these strategies useful to adjust the simulation model parameters?
- What are the advantages and disadvantages of the different strategies?
- Is the additional effort for using more complex techniques like ARMA justifiable?

6 Composition of Test Cases

The composition of test cases is always a very important step when a study is designed. In this case, the study is based on the recorded requests executed on a web system with a single apache web server and a MySQL database. The recorded requests are based on the load produced by a performance test tool always requesting the same PHP web site, which lists some entries of the database. More details concerning the web site and the recorded information are listed in figure 2 and in table 1. The time intervals used for the study are request phases of 1 minute, 15 minutes and 60 minutes. These time intervals should provide the possibility to evaluate on the one hand how fast a strategy adjusts the simulation model parameters and on the other hand how the accuracy of the strategies differ with the long runtime perspective. A special focus is placed on the long runtime perspective, as the strategies should be integrated into a system which runs in parallel to the SUT. Furthermore to provide all chosen settings used for the analysis and test cases, the parameters as described in section 4 are listed in table 2. This table shows the settings which were used for all test cases independent from the chosen strategy for simulation model parameter adjustment.

Type	Timestamp	Resource	Id
Start	1223278973740	http://localhost/mysql.php	2544748e9c17da037e8.40733164
Event	1223278973847	http://localhost/mysql.php	2544748e9c17da037e8.40733164
Ende	1223278973951.3	http://localhost/mysql.php	2544748e9c17da037e8.40733164

Table 1. A Schematic Example data set of a recorded request

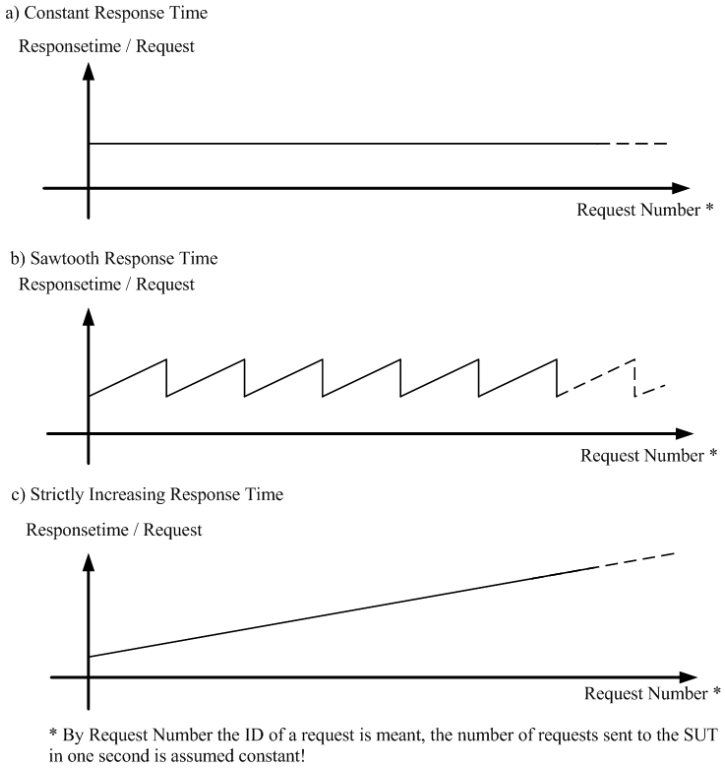

Fig. 2. Respons Time of Web Sites ”Simplified”

Request Phase Length	Response Time of Web Sites	Base Data Usage Type	Simulate Every Nr. of Requests	Save Nr. of Request for Evaluation
1 Minute	Constant Response Time	Type D	1	1
15 Minute	Constant Response Time	Type D	5	5
60 Minute	Constant Response Time	Type D	25	25
1 Minute	Sawtooth Response Time	Type D	1	1
15 Minute	Sawtooth Response Time	Type D	5	5
60 Minute	Sawtooth Response Time	Type D	25	25
1 Minute	Strictly Increasing Response Time	Type D	1	1
15 Minute	Strictly Increasing Response Time	Type D	5	5
60 Minute	Strictly Increasing Response Time	Type D	25	25

Table 2. Test Case Settings

7 Results of the Test Cases

In this section the results of the test cases described above will be summarized. The main results of the test cases are a ranking of the different strategies for the three time intervals, and a general ranking. In this ranking the strategies which have provided results that were not statistically significant different are assigned the same rank. For the statistical significant the double-sided case has been used, as no strategy should be favored in advance.

7.1 Test Cases One Minute

In the one minute test cases only the three strategies, AVG, Binary and ARMA have been tested, as the ARMA G strategy is not applicable to this timeframe. The ranking can be seen in table 3, based on this ranking ARMA with an average place of 1.0 is the best suitable strategy. Below that, with an average place of 2.0, comes the AVG strategy, where the advantage of reduced computation complexity has to be considered. In the last place is the Binary strategy. A graphical representation of the one minute sawtooth test case can be found in figure 3. In the one minute time frame only the increasing periode of the first sawtooth is considered.

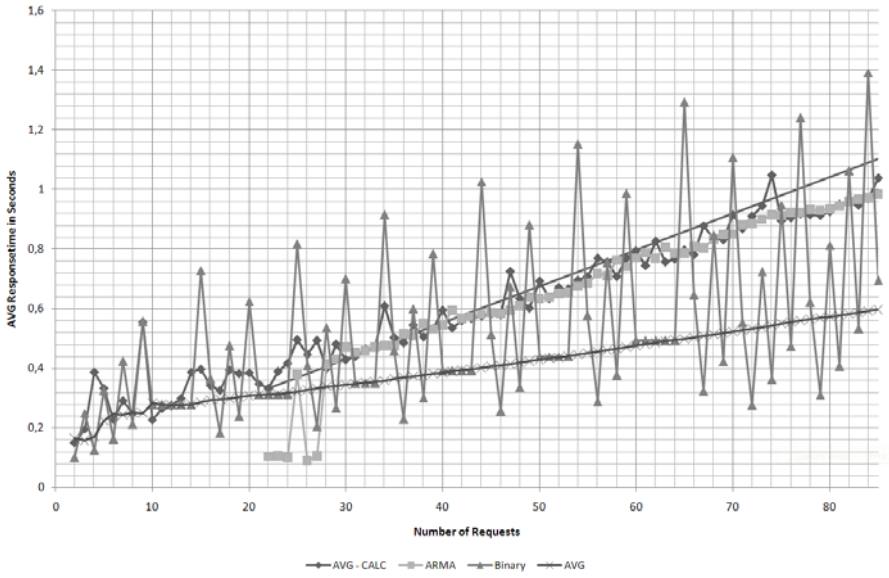

Fig. 3. Response time comparison in the ”Sawtooth” one minute test case

Web Side Response Time Type	Rank	Rank	Rank
Constant Response Time	(1) ARMA	(2) AVG	(2) Binary
Sawtooth Response Time	(1) ARMA	(2) AVG	(2) Binary
Strictly Increasing Response Time	(1) ARMA	(2) AVG	(3) Binary

Table 3. Test Case One Minute Ranking

7.2 Test Cases 15 Minutes

The ranking of the 15 minutes test cases can be found in table 4. In this time frame all presented strategies can be included in the test. The average ranking shows clearly the advantage of both, the ARMA strategy and the ARMA G. strategy with an average place of 1.333. In the third place follows the AVG strategy with an average place of 2.333, and the last place again falls to the Binary strategy. In figure 4 the comparison of the average response time results from the execution of the simulation with the real average response time of the SUT for ARMA, ARMA G. and AVG in the Strictly Increasing 15 Minutes test case is provided. Each dot represents the average response time of five requests, the AVG - Calc line visualizes the real average response time.

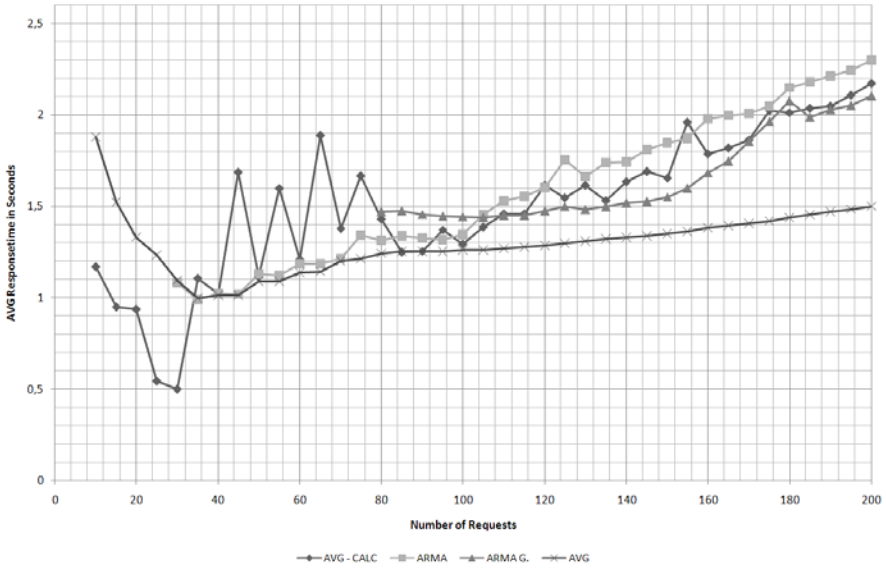

Fig. 4. Response time comparison in the ”Strictly Increasing” 15 minutes test case

Web Side Response Time Type	Rank	Rank	Rank	Rank
Constant Response Time	(1) AVG	(1) ARMA	(1) ARMA G.	(4) Binary
Sawtooth Response Time	(1) ARMA	(2) ARMA G.	(3) AVG	(4) Binary
Strictly Increasing Response Time	(1) ARMA G.	(2) ARMA	(3) AVG	(3) Binary

Table 4. Test Case 15 Minute Ranking

7.3 Test Cases 60 Minutes

The ranking of the 60 minutes test cases is provided in table 5, these results demonstrate that the difference between the ARMA/ARMA G. strategies and the AVG strategy depend on the underlying base response time. The first average place goes to the ARMA strategy with a value of 1.333, in the second place follows the AVG strategy with a value of 1.666 and in the third place the ARMA G. strategy with a value of 2. The last placed strategy is again the Binary strategy. The figure 5 compares the average response time results of the execution of the simulation with the real average response time of the SUT for ARMA, ARMA G. and AVG in the Sawtooth 60 Minutes test case. Each dot represents the average response time of 25 requests, the AVG - Calc line visualizes the real average response time. Figure 5 shows the results of the 60 minutes Sawtooth

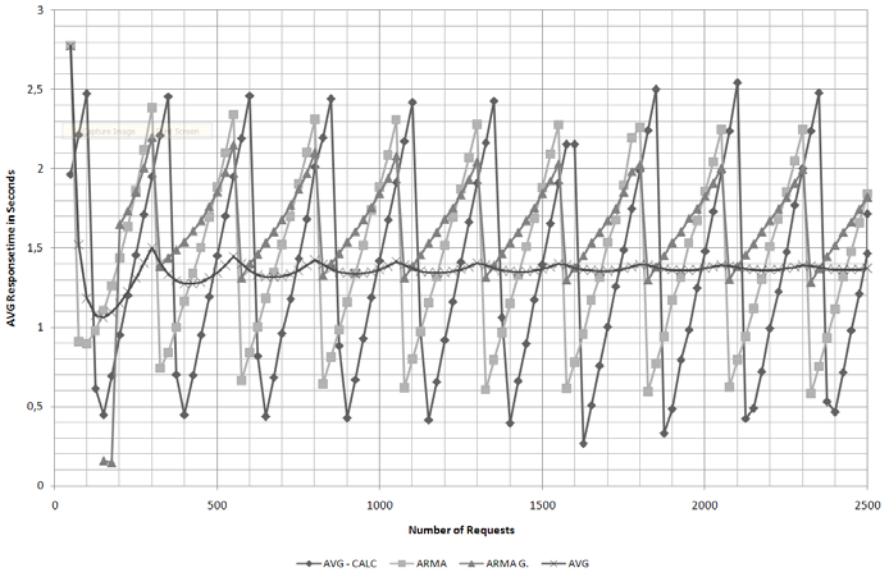

Fig. 5. Response time comparison in the ”Sawtooth” 60 minutes test case

Web Side Response Time Type	Rank	Rank	Rank	Rank
Constant Response Time	(1) ARMA	(1) AVG	(3) ARMA G.	(3) Binary
Sawtooth Response Time	(1) ARMA	(2) ARMA G.	(2) AVG	(4) Binary
Strictly Increasing Response Time	(1) ARMA G.	(2) Binary	(2) ARMA	(2) AVG

Table 5. Test Case 60 Minute Ranking

test case. They lead to the conclusion that the ARMA G. strategy does not follow the extreme values of the SUT response time can be drawn. Obviously the AVG strategy will result in a constant value for a sawtooth function over time.

7.4 Overall Ranking

The overall ranking clearly shows that the ARMA strategy is generally the most applicable but, taking into account computational complexity, the AVG strategy proves also a well suitable alternative. The ranking with the according average place values is provided in table 6.

Strategy	Rank	Value
Binary	4	2.666
AVG	3	2.000
ARMA	1	1.222
ARMA G.	2	1.666

Table 6. Strategy Ranking

8 Conclusions Drawn Based on the Results of the Study

The results of the test cases presented here lead to the conclusion that sometimes a simple strategy for simulation model parameter estimation like AVG can provide quite satisfactory results. Especially in cases where the computational power is limited, simple methods like AVG should be preferred. In cases where the computational power requirement is of no issue, the results of this study would recommend using ARMA and AVG in combination and to decide, based on the underlying function, which strategy to use for the prediction of the simulation model parameters.

If these two strategies could be combined in an advantageous way, the resulting strategy might provide the best results for all test cases. To this end, further studies of the behavior of the ARMA and ARMA G strategy should be conducted. The results obtained in the test cases presented here should furthermore be reevaluated on other web site structures and corresponding response time structures. A long term analysis

(for more than 24 hours) on a productive web site environment with a high fluctuation of requests are done with the AVG, ARMA and ARMA G strategy.

Additionally the impact of the predefined time period (cf. section 4 ”after how many new requests the simulation should be executed”) should be analyzed in more detail, as the latest studies suggest that this parameter has an influence on the results provided by the ARMA strategy.

Furthermore, the popular field of evolutionary programming will be evaluated in more detail. Nevertheless the expected problem with the method is that most likely the time constraints which come with the idea of the system will render this method not practicable, for this application.

References

1. Jsim http://www.j-sim.org.
2. Network simulator http://www.isi.edu/nsnam/ns.
3. George Box, Gwilym M. Jenkins, and Gregory Reinsel. *Time Series Analysis: Forecasting & Control (3rd Edition)*. Prentice Hall, February 1994.
4. Peter J. Brockwell and Richard A. Davis. *Time Series: Theory and Methods*. Springer Series in Statistics.
5. L. G. Cardenas, J. A. Gil, J. Domenech, J. Sahuquillo, and A. Pont. Performance comparison of a web cache simulation framework. In *AINA '05: Proceedings of the 19th International Conference on Advanced Information Networking and Applications*, pages 281–284, Washington, DC, USA, 2005. IEEE Computer Society.
6. Raj Jain. *The Art of Computer Systems Performance Analysis*, pages 93–110. John Wiley and Sons, Inc., 1991.
7. John A. Miller, Youngfu Ge, and Junxin Tao. Component-based simulation environments: Jsim as a case study using java beans. In *WSC '98: Proceedings of the 30th conference on Winter simulation*, pages 373–382, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press.
8. Adrian Mos and John Murphy. A framework for performance monitoring, modelling and prediction of component oriented distributed systems. In *WOSP '02: Proceedings of the 3rd international workshop on Software and performance*, pages 235–236, New York, NY, USA, 2002. ACM.
9. Martin Pinzger. Automated web performance analysis. In *ASE*, pages 513–516. IEEE, 2008.
10. P. Schwarz and U. Donath. Simulation-based performance analysis of distributed systems, 1997.
11. Robert A. Stine. Time series models and Mathematica. pages 368–406. 1993.
12. Stephen Wolfram. *The Mathematica Book*. Cambridge University Press, third edition edition, 1996.

Adaptation and Monitoring in S-Cube: Global Vision and Roadmap*

Raman Kazhamiakin**

FBK-Irst, via Sommarive 18, 38050, Trento, Italy
raman@fbk.eu

Abstract. The need to adapt to the changes implied by dynamic business environments and constantly evolving requirements impose new challenges for the development of modern Service-Based Applications. These applications become drastically more flexible; they should be able to adequately identify and react to various changes in the business requirements and application context. These challenges make monitoring and adaptation the key elements of modern SBA functionality.
The vision on the adaptation and monitoring we adopt in the S-Cube project identifies a generic process that integrates adaptation and monitoring activities in order to identify critical changes and problems. This vision aims to generalize and broaden the state of the art in the SBA adaptation. First, it broadens the perspective on what, how, and when may be monitored and adapted in order to accommodate to changes and deviations. Second, it extends the definition of monitoring and adaptation themselves in order to bring into the game approaches and mechanisms traditionally not exploited for these purposes. Last, but not least, the S-Cube adaptation and monitoring vision aims to to cover and integrate various research disciplines (such as Service-Oriented and Grid Computing, Business Process Management, and Software Engineering), application domains (B2B, user-centric systems), as well as different functional SBA layers. On the one hand, this allows us to bring and re-use the ideas and approaches from completely independent areas. On the other hand, such an integration makes the cross-layer adaptation and monitoring first-class elements of this vision.
Seen from this broader perspective, such adaptation and monitoring picture aims to integrate otherwise isolated and fragmented adaptation and monitoring approaches, methodologies and application domains, thus opening up new research challenges and opportunities not only within the project, but also for the whole SOA science and technology.

1 Introduction

Service-Based Applications (SBA) run in dynamic business environments and address constantly evolving requirements. These applications should hence become drastically

* The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube).

** This paper reports the vision jointly provided by the participants of the S-Cube workpackage WP-JRA-1.2 "Adaptation and Monitoring Principles, Techniques, and Mechanisms for Service-based Applications".

more flexible, as they should be able to adequately identify and react to various changes in the business requirements and application context. These challenges make monitoring and adaptation the key elements of modern SBA functionality.

The problem of monitoring and adaptation of various types of software system has gained a lot of interest both in the research community and in industry. In the recent years, these aspects have attracted more and more interest in the area of SBA and in Service-Oriented Computing (SOC). However, the results and directions are still insufficient. First, the proposed approaches are very *fragmented*; they address only specific problems, particular application domains, and particular types of applications and systems; the monitoring solutions are often isolated from the adaptation needs and approaches. Second, most of the approaches dealing with adaptation address the problem *reactively*: the solutions aim to define a way to recovery from the problem when it is already happened rather than to prevent it to happen. This is, indeed, insufficient in certain applications and domains. Third, as the applications, their users, and the settings where they operate become more and more dynamic, open, and unpredictable, the role of the *application context* (being a physical, business, or user-specific) becomes much more critical. These issues are often omitted by the state-of-the-art solutions both for monitoring and adaptation. Very relevant to this is also the role and participation of *various types of users* in the monitoring and adaptation process. The service-based applications are often designed to target final users, and, therefore, should be able to collect and properly exploit the information about the user in order to customize and personalize those applications as well as to let the users participate to the corresponding activities.

In the S-Cube project the work on SBA monitoring and adaptation is devoted to the development of the novel principles, techniques, and mechanisms focused on the following key research aspects and questions [1]:

- **Comprehensive adaptation and monitoring framework**. The work will concentrate on providing holistic framework for adaptation and monitoring principles, techniques, and methods, which will enable the integration of different, isolated, and fragmented solutions. In particular, the framework will allow for:
  - *Cross layer integration of monitoring approaches*. This form of integration is crucial for modern SBA provisioning, as it provides a way to properly locate and evaluate the source of the problem and its impact.
  - *Cross layer integration of adaptation approaches*. This form of integration is complementary to the previous one and will allow us to properly identify and propagate the necessary adaptation activities in different elements of the SBA architecture. As well as in case of monitoring, new solutions will integrate isolated adaptation mechanisms available at different functional layers into the holistic cross layer approaches.
  - *Cross boundary integration* of monitoring and adaptation techniques. Here the focus on identifying the role and the impact of various monitored events and adaptation actions on the different participants of the system and its environment, as well as on distributing the information and the actions across those participants accordingly.
  - *Cross life-cycle integration* of monitoring and adaptation techniques to exploit the knowledge and mechanisms available at different phases of the life-cycle

(e.g., design-time or post-operational information) in order to devise, e.g., new monitoring approaches (e.g., exploiting post-mortem process analysis for prediction) or adaptation decision mechanisms (e.g., explore previous decisions and adaptation effects to select proper adaptation strategy).

- **Predictive SBA monitoring and proactive SBA adaptation**. This work will concentrate on the problem of predicting the critical changes in SBA functioning in order to proactively prevent undesired situations. In particular, the work will focus on new techniques and solutions for adapting the system based on the predicted quality values.
- **Exploiting contextual information and user aspects for SBA monitoring and adaptation**. The information about different types of the SBA context, as well as about the user and its settings, is crucial for the application logic. Novel approaches are necessary for being able to specify and observe this information and for driving the selection, realization, and enactment of the corresponding adaptation actions.

Here we illustrate the novel vision on the SBA adaptation and monitoring that we have defined in S-Cube; this vision will provide a comprehensive, coherent framework for the existing challenges and for the different research lines undertaken by S-Cube and by the broader SOC research community. The vision will also place the adaptation and monitoring research within the global picture and objectives of the S-Cube project; will drive the identification of the competences – and gaps – of the consortium, and to define the research roadmap, which will also be addressed by the project.

2 Conceptual Adaptation and Monitoring Framework

At the high level of abstraction, the adaptation and monitoring framework can be described by the concepts represented in Figure 1. This figure identifies Monitoring Mechanisms, Monitored Events, Adaptation Requirements, Adaptation Strategies, Adaptation Mechanisms, and the relations between these concepts, as the key elements of the S-Cube A&M framework.

It is important to remark that the significance and the novelty of this conceptual framework is not in the figure itself – it describes a standard sensing/planning/actuating control chain. The significance and novelty is in the interpretation and a very broad meaning that we give to the different concepts, and to the capability of the chain (i) to allow for a very general integration of a wide range of mechanisms, techniques and methodologies for monitoring and adaptation; and (ii) to allow for an ability to capture and address new challenges and problems like cross-layer adaptation and monitoring, pro-active SBA adaptation, and HCI-driven monitoring.

2.1 Elements of the Framework

A generic adaptation and monitoring framework consists of the following elements:

- With *Monitoring Mechanism* we mean **any** mechanism that can be used to check whether the actual situation corresponds to the expected one. The meaning we give

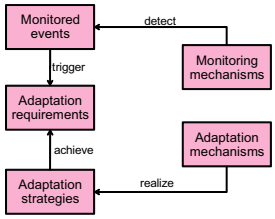

Fig. 1. Conceptual A&M framework

to the monitoring mechanisms is very broad; in this way, we refer not only to "classical" run-time monitoring facilities, but also to techniques such as post-mortem log analysis techniques, data mining, online and offline testing and even verification/validation, etc. Realization of monitoring mechanisms is provided by the corresponding monitoring engines built on top of the monitoring infrastructures.

- Monitoring mechanisms are used to detect *Monitored Events*, i.e., the events that deliver the relevant information about the application execution, evolution, and context. These events represent the fact that there is critical difference with respect to the expected SBA state, functionality, and environment. The monitored events result from observing monitoring properties, derived from the adaptation requirements as a specification of the expected state and functionality of the SBA and its environment. The notion of monitored events may be very broad ranging from basic failures, deviation of QoS parameters, to complex properties over many executions of SBA, certain trends in the SBA environment, changes in business rules, etc.
- Monitored events in turn trigger *Adaptation Requirements*, which represent the necessity to change the underlying SBA in order to remove the difference between the actual (or predicted) situation and the expected one. They may include dependability and functional correctness requirements, optimality, interoperability, usability, etc.
- In order to satisfy adaptation requirements, it is necessary to define *Adaptation Strategies*, which define the possible ways to achieve those requirements given the current situation. Note that it is possible to have a set of different adaptation strategies applicable in the same situation. In this case the process requires certain decision mechanisms that operate autonomously or involve humans.
- Finally, the adaptation strategies are realized by the *Adaptation Mechanisms* – the techniques and facilities provided by the underlying SBA or by the operation and management platform in different functional SBA layers that enable corresponding strategies. The adaptation may be also done "manually", i.e., by re-designing/re-engineering the application. In this case we should speak about application evolution as the permanent SBA changes are required that should be done via SBA re-design.

An important aspect of these conceptual elements is the necessity to define and implement the corresponding *decision mechanisms*, which correspond to the four arrows in the picture in Figure 1 and coordinate the work of the framework and realize the relations among them. In particular,

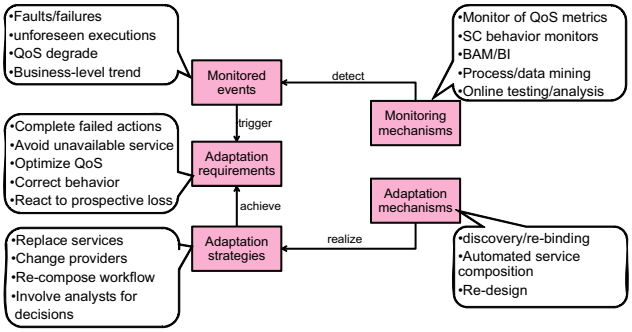

Fig. 2. Realization of conceptual elements

- *Monitoring properties* allow us to analyze the variety of SBA information observed during its execution and evolution, and to extract and report those events and situations that are critical from the point of view of the monitoring.
- *Adaptation decision mechanisms* relate the monitoring activities with the adaptation activities: they regulate when a particular monitored event corresponds to a situation in which the system should be changed.
- *Strategy decision mechanisms* define the way a particular adaptation strategy is chosen based on the adaptation needs, SBA state, history of previous adaptations, etc. In particular, these mechanisms will provide a way to resolve conflicts among different adaptation requirements.
- *Realization mechanisms* define how a particular strategy is realized, when there is a wide range of available options (e.g., many services to bind in place of failed one).

Note that the realization of these decision mechanisms may be done automatically or may require user involvement. In the latter case we speak about the human-in-the-loop adaptation: the users (with different roles) may decide whether the adaptation is needed, which strategy to choose, and even participate to its realization (e.g., manual ad-hoc SBA adaptation through re-design).

2.2 Usages of the Framework

The role of the picture in Figure 2 is threefold: to provide an integrated model of the A&M framework, to define a conceptual architecture of such a framework, and to identify an overall adaptation process.

As an **integrated model** for the A&M framework it defines key concepts for monitoring and adaptation which are by design very general. This allows us to capture any adaptation approach in a uniform way, independently from the problem or application domain, discipline, functional layer, or type of the problem addressed. This also provides a basis for the integration of different solutions within a single approach, and re-use of existing solutions for various purposes.

Each conceptual element may be instantiated in a variety of ways (see Figure 2):

- different mechanisms and techniques may be exploited for the SBA monitoring such as run-time monitoring tools, online testing, process log analysis.
- a variety of different events may be observed for the same application such as various faults, QoS degrade and violation of SLAs, deviation from the expected behavior. These event may refer to a particular instance (or execution) of an application or to all the instances; they may also correspond to different functional SBA layers.
- the list of adaptation requirements, strategies, and their implementations may be also very broad (e.g., re-execution of a particular service or changing a provider, modifying the composition or even re-design of the application) and may also refer to a particular instance or to the whole application model.

Furthermore, these instances may be further combined in a variety of ways within different approaches, and then applied to the same SBA.

Almost all the existing approaches covered in the state of the art [2] can be mapped into this model by suitable instantiations of the conceptual elements. For instance,

- Approaches with dynamic re-binding: the service composition is monitored, service faults are detected, a re-execution strategy is realized through discovering, re-binding, and invoking alternative service.
- Provider reputation-based adaptation: QoS metrics statistics is collected, SLA violations are detected, a reputation management strategy is applied, the provider is added to the black list and replaced.

More important, novel approaches are being defined as part of the S-Cube research activities by new instantiations of this model:

- Cross-layer adaptation: the service events are monitored, the dependency analysis is applied to correlate service events to the business activity events (like unforeseen process execution), the process instance is modified by the business analysts in ad-hoc manner, the modification is propagated to the composition and infrastructural layers.
- Proactive adaptation based on online testing [3]: composition is tested, the unavailability of a service component is detected, alternative service is discovered and rebind in the composition.

Second, this picture defines a **conceptual architecture** of a comprehensive adaptation framework. The modularization of these concepts allows us to define key components of the A&M tools, to identify the interfaces between these components and to abstract from any specific realization of these components. On the contrary, within the same component different mechanisms and techniques may be applied in combination. In this way, one can obtain more flexible, customizable, and powerful adaptation and monitoring solutions.

Third, the picture identifies an **overall adaptation process**, where: (i) the relevant information is collected through the monitoring mechanisms; (ii) critical events are recognized; (iii) the need for adaptation is identified; (iv) an appropriate way to perform the adaptation is identified, i.e., an adaptation strategy is selected; and (v) the adaptation is realized by exploiting the available adaptation mechanisms. This adaptation approach is aligned with the SBA life-cycle and is represented in Figure 4.

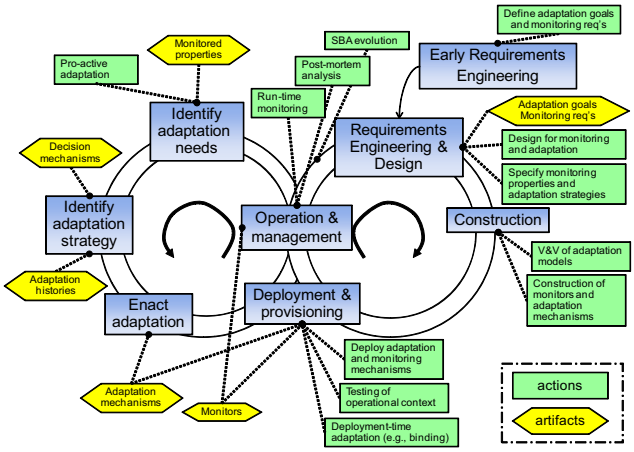

Fig. 3. Life-cycle: contributions related to adaptation

2.3 Monitoring and Adaptation in SBA Life-cycle

An orthogonal view on the monitoring and adaptation problem may be thought of if we try to place this problem within the SBA life-cycle described in [4, 5] (Figure 3). This view shows how the relevant activities and artifacts related to the conceptual elements presented previously are spread across the whole SBA life, and provides a basis for the cross-life-cycle integration of various A&M-related mechanisms, approaches, and techniques.

In particular, the figure shows how various adaptation- and monitoring-specific actions (depicted in green) are carried out throughout the life-cycle of the SBA and how the corresponding artifacts (depicted in yellow) are exploited within those activities. We remark that different adaptation and monitoring activities and artifacts are highly interleaved through the SBA life-cycle and affect each other. Consequently, their definition, development, and exploitation should be performed in holistic manner motivating the need for cross life-cycle monitoring and adaptation methods and approaches.

During *early requirements engineering*, the adaptation goals and monitoring requirements are defined. The goals and requirements are based on the quality model of the developed SBA and may involve various aspects, quality characteristics and attributes. In this phase, it is important not only to devise the appropriate monitoring and adaptation facilities, but also to see how the adaptation goals should be achieved, that is, whether the application has to be modified pro-actively or re-actively, what is the role of the contextual and user-specific information, etc.

At the *requirements engineering and design* phase the goals and requirements are brought in to perform the design for adaptation and monitoring. In particular, the appropriate monitoring and adaptation architecture, the decision mechanisms, and the corresponding techniques are being selected, adopted, and instantiated. Using this model, the monitoring requirements and adaptation goals are being transformed into the monitoring properties and the adaptation strategies respectively.

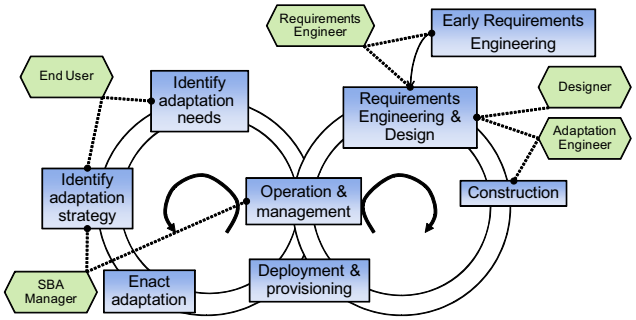

Fig. 4. User perspective on the Adaptation and Monitoring across SBA life-cycle

During SBA *construction*, together with the construction of the SBA, the corresponding monitors and the adaptation mechanisms are being realized. It is also important in this phase to ensure that the developed adaptation mechanisms are neither contradictory nor in conflict with the application logic, i.e., to perform adaptation-specific quality assurance activity, such as validation and verification of the adaptation strategies, specifications, and realizations.

The *deployment* phase involves also the activities related to adaptation and monitoring: deployment of the monitors and monitoring mechanisms; deployment time adaptation actions (e.g., binding), testing and validation of operational context (e.g., evaluation of QoS metrics). After the *operation and management* phase, where the monitoring and adaptation activities interleave the application execution, specific post-execution activities may take place. This includes, for instance, the analysis of the previous SBA executions and adaptations, as well as the evolution of SBA and related mechanisms, bringing the latter back to the design phase.

The adaptation and monitoring mechanisms, tools, and facilities are actively exploited during the phases related to the modification of SBA, i.e., during the identification and realization of the adaptation needs. In particular, the most critical monitored properties characterize a serious deviation of the SBA functioning from the expected one, and therefore *identify adaptation needs*. Depending on the adaptation requirements this identification may be done re-actively or pro-actively. Various *adaptation strategies* developed during the design-time phase are instantiated and selected using the corresponding decision mechanisms, based on the current situation and/or on the knowledge obtained from the previous adaptations and executions. Finally, the *enactment of the adaptation strategy* is performed by the developed adaptation mechanisms. We remark that the implementation of these activities and phases may be performed by the SBA autonomously or may involve active participation of the various human actors (human-in-the-loop adaptation).

2.4 User Perspective

An important perspective of the introduced vision concerns the involvement of different user roles in the adaptation- and monitoring-related activities across the overall

A&M process and across the activities of the SBA life-cycle (Figure 4). This perspective introduces an additional dimension of the problem, which makes the corresponding approaches range from completely autonomous (self-* approaches) to interactive and manual (human-in-the-loop approaches). We can distinguish the involvement of the users according to the participation to the life-cycle of the adaptable SBA and to the adaptation and monitoring process.

In case of participation to the life-cycle activities one can identify the roles of requirements engineers, designers, and adaptation engineers. A *Requirements Engineer* defines the application requirements and, therefore, identifies and derives the adaptation and monitoring requirements. A *Designer* (besides designing the SBA itself) may perform manual or semi-automatic design-time adaptation of the application based on the information and requests for changes triggered at the operation and management phase of the SBA life-cycle. An *Adaptation Engineer* performs specific activities that target design for monitoring and adaptation, i.e., definition and specification of monitored properties and adaptation strategies, and possibly engineering of novel A&M techniques and mechanisms.

In case of participation to the adaptation process, specific roles may also be identified. Given the conceptual model, these roles correspond to the participation of the user in the realization of various decision mechanisms (to define whether adaptation is needed, which strategy to use and how to realize it). This includes *SBA Manager* (or Integrator), who observes how the application is executed and evolves in order to make critical decisions (e.g., triggering requests for SBA re-design/re-engineering), and *End Users*. The latter may be involved into the A&M process as follows: in case of user-centric SBAs, the adaptation aims to address the needs, preferences, and expectations of a particular user; the system adapts to the context of the user and to the way the user interacts with the application. Therefore, end-users directly or indirectly influence the way the adaptation and monitoring is performed, i.e., affect adaptation and monitoring mechanisms.

3 Roadmap

We have presented a global vision on the problem of SBA monitoring and adaptation adopted within the S-Cube research project. Having started from a very abstract and high-level conceptual model, we have considered the monitoring and adaptation from different perspectives, including the overall adaptation and monitoring process, role of these activities in the SBA life-cycle, user involvement etc. The aim behind this perspectives was to to bring together different research disciplines, application domains, and functional elements of the SBA architecture.

The presented perspectives on the SBA monitoring and adaptation identify not only the key elements and concepts of the corresponding SBA functionalities and architecture, but also highlight the potential challenges and research problems across relevant SBA areas.

First, it makes evident the need of dramatically higher level of integration of currently fragmented research results in adaptation and monitoring. The scope of these functionalities should be broadened both horizontally and vertically. In the former case

the integration requires frameworks and methodologies that consider whole life-cycle of SBAs: novel techniques and approaches focusing on the design for monitoring and adaptation, Quality Assurance methods specifically focusing on the adaptation activities; integration of proactive monitoring and preventive adaptation with post-mortem analysis techniques and SBA re-engineering activities. Furthermore, such integration enables not only extending the adaptation and monitoring from being narrowly run-time activities to end-to-end activities, but also brings the other activities (such as testing, verification, model analysis) into the adaptation process (e.g., to enable pro-active adaptation). In the latter case, the integration requires bringing together and considering various SBA elements that are subject to monitoring and adaptation. As a result, integrated frameworks are necessary that enable observing information and enacting adaptation activities at different functional SBA layers, across single SBA instance or across variety of instance and classes, and even across the adaptation and monitoring mechanisms themselves. This also concerns the SBA context, which, considered from broader perspective, may include not only execution infrastructure of the SBA, but also SBA users, user preferences and environments, business settings, grid architectures, etc.

Second, the vision spots the importance of the human involvement in the adaptation and monitoring process. While there exists a wide range of approaches that target autonomous mode of functionality and self-adaptation of SBA, the role of the user in many adaptation and monitoring activities is crucial. This ranges from the design phase, where the traditional steps are extended in order to take into account specific adaptation activities, to run-time phase, where the SBA users, SBA managers and operators are actively involved and influence the adaptation process, e.g., by making decisions or by changing preferences and SBA configurations. Consequently, novel methodologies are needed that take into account the specific user activities and interfaces at different phases of the SBA life-cycle.

References

1. Hielscher, J., Metzger, A., Kazhamiakin, R., eds.: Taxonomy of Adaptation Principles and Mechanisms. S-Cube project deliverable (2009) S-Cube project deliverable: CD-JRA-1.2.2. http://www.s-cube-network.eu/achievements-results/s-cube-deliverables.
2. Benbernou, S., Cavallaro, L., Hacid, M.S., Kazhamiakin, R., Kecskemeti, G., Pazat, J.L., Silvestri, F., Uhlig, M., Wetzstein, B.: State of the Art Report, Gap Analysis of Knowledge on Principles, Techniques and Methodologies for Monitoring and Adaptation of SBAs. S-Cube project deliverable (2008) S-Cube project deliverable: PO-JRA-1.2.1. http://www.s-cube-network.eu/achievements-results/s-cube-deliverables.
3. Hielscher, J., Kazhamiakin, R., Metzger, A., Pistore, M.: A Framework for Proactive Self-Adaptation of Service-based Applications Based on Online Testing. In: ServiceWave 2008, to be published (2008)
4. Nitto, E.D., ed.: State of the art report on software engineering design knowledge and Survey of HCI and contextual Knowledge. S-Cube project deliverable (2008) S-Cube project deliverable: PO-JRA-1.1.1. http://www.s-cube-network.eu/achievements-results/s-cube-deliverables.
5. Pistore, M., Kazhamiakin, R., Bucchiarone, A., eds.: Integration Framework Baseline. S-Cube project deliverable (2009) S-Cube project deliverable: CD-IA-3.1.1. http://www.s-cube-network.eu/achievements-results/s-cube-deliverables.

Previously published ICB - Research Reports

2009

No 33 (May 2009)

Heimo Adelsberger, Andreas Drechsler, Tobias Bruckmann, Peter Kalvelage, Sophia Kinne, Jan Pellinger, Marcel Rosenberger, Tobias Trepper: Einsatz von Social Software in Unternehmen – Studie über Umfang und Zweck der Nutzung

No 32 (April 2009)
Barth, Manfred; Gadatsch, Andreas; Kütz, Martin; Rüding, Otto; Schauer, Hanno; Strecker, Stefan: Leitbild IT-Controller/-in – Beitrag der Fachgruppe IT-Controlling der Gesellschaft für Informatik e. V.

No 31 (April 2009)
Frank, Ulrich; Strecker, Stefan: Beyond ERP Systems: An Outline of Self-Referential Enterprise Systems – Requirements, Conceptual Foundation and Design Options

No 30 (February 2009)
Schauer, Hanno; Wolff, Frank: Kriterien guter Wissensarbeit – Ein Vorschlag aus dem Blickwinkel der Wissenschaftstheorie (Langfassung)

No 29 (January 2009)
Benavides, David; Metzger, Andreas; Eisenecker, Ulrich (Eds.): Third International Workshop on Variability Modelling of Software-intensive Systems

2008

No 28 (December 2008)
Goedicke, Michael; Striewe, Michael; Balz, Moritz: „Computer Aided Assessments and Programming Exercises with JACK“

No 27 (December 2008)
Schauer, Carola: “Größe und Ausrichtung der Disziplin Wirtschaftsinformatik an Universitäten im deutschsprachigen Raum - Aktueller Status und Entwicklung seit 1992”

No 26 (September 2008)
Milen, Tilev; Bruno Müller-Clostermann: “ CapSys: A Tool for Macroscopic Capacity Planning”

No 25 (August 2008)
Eicker, Stefan; Spies, Thorsten; Tschersich, Markus: “Einsatz von Multi-Touch beim Softwaredesign am Beispiel der CRC Card-Methode”

No 24 (August 2008)
Frank, Ulrich: “The MEMO Meta Modelling Language (MML) and Language Architecture – Revised Version”

No 23 (January 2008)
Sprenger, Jonas; Jung, Jürgen: “Enterprise Modelling in the Context of Manufacturing – Outline of an Approach Supporting Production Planning”

No 22 (January 2008)
Heymans, Patrick; Kang, Kyo-Chul; Metzger, Andreas, Pohl, Klaus (Eds.): “Second International Workshop on Variability Modelling of Software-intensive Systems”

2007

No 21 (September 2007)
Eicker, Stefan; Annett Nagel; Peter M. Schuler: "Flexibilität im Geschäftsprozess-management-Kreislauf"

No 20 (August 2007)
Blau, Holger; Eicker, Stefan; Spies, Thorsten: "Reifegradüberwachung von Software"

No 19 (June 2007)
Schauer, Carola: "Relevance and Success of IS Teaching and Research: An Analysis of the ‚Relevance Debate'

No 18 (May 2007)
Schauer, Carola: "Rekonstruktion der historischen Entwicklung der Wirtschaftsinformatik: Schritte der Institutionalisierung, Diskussion zum Status, Rahmenempfehlungen für die Lehre"

No 17 (May 2007)
Schauer, Carola; Schmeing, Tobias: "Development of IS Teaching in North-America: An Analysis of Model Curricula"

No 16 (May 2007)
Müller-Clostermann, Bruno; Tilev, Milen: "Using G/G/m-Models for Multi-Server and Mainframe Capacity Planning"

No 15 (April 2007)
Heise, David; Schauer, Carola; Strecker, Stefan: "Informationsquellen für IT-Professionals – Analyse und Bewertung der Fachpresse aus Sicht der Wirtschaftsinformatik"

No 14 (March 2007)
Eicker, Stefan; Hegmanns, Christian; Malich, Stefan: "Auswahl von Bewertungsmethoden für Softwarearchitekturen"

No 13 (February 2007)
Eicker, Stefan; Spies, Thorsten; Kahl, Christian: "Softwarevisualisierung im Kontext serviceorientierter Architekturen"

No 12 (February 2007)
Brenner, Freimut: "Cumulative Measures of Absorbing Joint Markov Chains and an Application to Markovian Process Algebras"

No 11 (February 2007)
Kirchner, Lutz: "Entwurf einer Modellierungssprache zur Unterstützung der Aufgaben des IT-Managements – Grundlagen, Anforderungen und Metamodell"

No 10 (February 2007)
Schauer, Carola; Strecker, Stefan: "Vergleichende Literaturstudie aktueller einführender Lehrbücher der Wirtschaftsinformatik: Bezugsrahmen und Auswertung"

No 9 (February 2007)
Strecker, Stefan; Kuckertz, Andreas; Pawlowski, Jan M.: "Überlegungen zur Qualifizierung des wissenschaftlichen Nachwuchses: Ein Diskussionsbeitrag zur (kumulativen) Habilitation"

No 8 (February 2007)
Frank, Ulrich; Strecker, Stefan; Koch, Stefan: "Open Model - Ein Vorschlag für ein Forschungsprogramm der Wirtschaftsinformatik (Langfassung)"

2006

No 7 (December 2006)
Frank, Ulrich: "Towards a Pluralistic Conception of Research Methods in Information Systems Research"

No 6 (April 2006)
Frank, Ulrich: "Evaluation von Forschung und Lehre an Universitäten – Ein Diskussionsbeitrag"

No 5 (April 2006)
Jung, Jürgen: "Supply Chains in the Context of Resource Modelling"

No 4 (February 2006)
Lange, Carola: "Development and status of the Information Systems / Wirtschaftsinformatik discipline: An interpretive evaluation of interviews with renowned researchers, Part III – Results Wirtschaftsinformatik Discipline"

2005

No 3 (December 2005)
Lange, Carola: "Development and status of the Information Systems / Wirtschaftsinformatik discipline: An interpretive evaluation of interviews with renowned researchers, Part II – Results Information Systems Discipline"

No 2 (December 2005)
Lange, Carola: "Development and status of the Information Systems / Wirtschaftsinformatik discipline: An interpretive evaluation of interviews with renowned researchers, Part I – Research Objectives and Method"

No 1 (August 2005)
Lange, Carola: „Ein Bezugsrahmen zur Beschreibung von Forschungsgegenständen und -methoden in Wirtschaftsinformatik und Information Systems"

Research Group	Core Research Topics
Prof. Dr. H. H. Adelsberger Information Systems for Production and Operations Management	E-Learning, Knowledge Management, Skill-Management, Simulation, Artificial Intelligence
Prof. Dr. P. Chamoni MIS and Management Science / Operations Research	Information Systems and Operations Research, Business Intelligence, Data Warehousing
Prof. Dr. F.-D. Dorloff Procurement, Logistics and Information Management	E-Business, E-Procurement, E-Government
Prof. Dr. K. Echtle Dependability of Computing Systems	Dependability of Computing Systems
Prof. Dr. S. Eicker Information Systems and Software Engineering	Process Models, Software-Architectures
Prof. Dr. U. Frank Information Systems and Enterprise Modelling	Enterprise Modelling, Enterprise Application Integration, IT Management, Knowledge Management
Prof. Dr. M. Goedicke Specification of Software Systems	Distributed Systems, Software Components, CSCW
Prof. Dr. T. Kollmann E-Business and E-Entrepreneurship	E-Business and Information Management,E-Entrepreneurship/ E-Venture, Virtual Marketplaces and Mobile Commerce, Online-Marketing
Prof. Dr. B. Müller-Clostermann Systems Modelling	Performance Evaluation of Computer and Communication Systems, Modelling and Simulation
Prof. Dr. K. Pohl Software Systems Engineering	Requirements Engineering, Software Quality Assurance, Software-Architectures, Evaluation of COTS/Open Source-Components
Prof. Dr.-Ing. E. Rathgeb Computer Networking Technology	Computer Networking Technology
Prof. Dr. A. Schmidt Pervasive Computing	Pervasive Computing, Uniquitous Computing, Automotive User Interfaces, Novel Interaction Technologies, Context-Aware Computing
Prof. Dr. R. Unland Data Management Systems and Knowledge Representation	Data Management, Artificial Intelligence, Software Engineering, Internet Based Teaching
Prof. Dr. S. Zelewski Institute of Production and Industrial Information Mangement	Industrial Business Processes, Innovation Management, Information Management, Economic Analyses

For more information visit us on the Web: **http://www.icb.uni-due.de**

ISSN 1860-2770 (Print)
ISSN 1866-5101 (Online)