[RES+08] *Nicolas Repp, Julian Eckert, Stefan Schulte, Michael Niemann, Rainer Berbner, Ralf Steinmetz*; **Towards Automated Monitoring and Alignment of Service-based Workflows**. In: IEEE International Conference on Digital Ecosystems and Technologies 2008 (IEEE DEST 2008). February 2008. S.235-240.

# Towards Automated Monitoring and Alignment of Service-based Workflows

Nicolas REPP, Julian ECKERT, Stefan SCHULTE, Michael NIEMANN,
Rainer BERBNER and Ralf STEINMETZ, Fellow, IEEE
Multimedia Communications Lab (KOM), Department of Electrical Engineering and Information Technology,
Technische Universität Darmstadt, Germany
e-mail (corresponding author): repp@kom.tu-darmstadt.de

*Abstract* – **Using Web services and Service-oriented Architectures to implement cross-organizational workflows has become state-of-the-art for the realization of collaborations between enterprises. Here, a key issue is the monitoring of workflows and services based on given business requirements and the handling of deviations from those requirements to fulfil Service Level Agreements.**

**In this paper we present an approach to the automated monitoring and alignment of service-based workflows. We describe a policy language for the specification of requirements and deviation handling as well as a distributed architecture supporting automated monitoring and alignment in service ecosystems.**

*Index Terms* – **Distributed Monitoring, Deviation Handling, Quality of Service, Service-oriented Architecture, Governance.**

## I. Introduction

Nowadays, service ecosystems, i.e. ecosystems in which services are used as implementation means for collaborations between the participating parties [1], are gaining more and more importance for enterprises. Here, services of different parties are combined to cross-organizational workflows, i.e., the part of a business process that is supported by software, in order to support outsourcing relationships between business partners.

Opening up the enterprise's workflows and integrating services of third parties lead to various challenges, which have to be addressed for the realization of dependable and trusted cross-organizational workflows. In the first instance, Service Level Agreements (SLA) have to be defined between all participating parties addressing Quality of Service (QoS) requirements or each party's responsibilities in the collaboration. But even if a detailed SLA is present, there is still the need to govern the workflow during runtime.

In practise, there is a large gap between the requirements of business and the IT providing the required services. Requirements defined by business departments can often hardly be mapped on IT and services. Current business and IT alignment approaches are focussing on the automated transformation of business requirements and processes into applications, mainly based on services and workflow technologies. But the simple generation of workflows and their mapping on services is not sufficient. Requirements for monitoring also have to be extracted during the transformation in order to allow automated monitoring of the business requirements at runtime.

In this paper we present an approach to face this challenge. Our approach, named *A*utomated *M*onitoring and *A*lignment of *S*ervices (AMAS.KOM), allows not only the automated monitoring of service-based workflows but also supports the automated deviation handling if given requirements are not met. The monitoring and deviation handling can be distributed between service providers, requesters, and intermediaries, allowing the optimization of reaction times to deviations. As implementation technology of choice we use Web services and the Web Service Business Execution Language (WSBPEL) for the description of workflows and WS-Policy compliant languages to describe business requirements, but our approach also can be applied to other technologies.

The remaining part of this paper is structured as follows. In the next section we present related work to our own approach. Here, we especially focus on state-of-the-art monitoring approaches as well as on languages for the description of monitoring requirements. In the following section, we introduce our monitoring and deviation handling approach. Afterwards, the modelling of requirements with respect to service-based workflows and possible reactions to deviations is discussed in more detail. Therefore, we developed the Web services requirements and reactions policy language (WS-Re2Policy), which is presented in this section. Before the paper closes with a conclusion and outlook, we will present the AMAS.KOM architecture as a proof of concept for our approach and the application of the WS-Re2Policy language.

## II. Related Work

There are various approaches for the monitoring of service-based workflows, which can be divided into monitoring of functional and non-functional requirements.

Robinson discusses monitoring of functional requirements formalized in temporal logic and carried out in parallel to the execution of the workflow [2]. Deviation handling is not part of his approach. Also using logical languages for the description of functional monitoring requirements, Spanoudakis and Mahbub discuss the transformation of BPEL4WS code into an event calculus based language and its subsequent monitoring [3]. Their approach only focuses on monitoring and the reporting of results, not on the deviation handling based upon the monitoring results. Monitoring based on pre- and post-conditions included as annotations in BPEL code is discussed by [4], using a BPEL pre-

processor for the extraction of the monitoring requirements interweaved in the code. Again, no deviation handling is integrated in this approach.

Apart from the monitoring of functional requirements, there are various non-functional monitoring approaches. A proprietary monitoring solution, based on the inspection of SOAP messages (e.g., response time and throughput) is part of a commercial software solution by Computer Associates. The monitoring results can be used to manage a network – automatic deviation handling is not yet part of the solution. Another conceptual framework capable of monitoring non-functional requirements and QoS-aware replanning of service-based workflows is described by [5] but without a proof of concept. Schmietendorf et al. are focussing on trustworthy performance and availability measurement and monitoring by independent third parties [6]. Again, no deviation handling is included in the approach.

In our own recent work, we provide a solution to the QoS-aware service selection and replanning of service-based workflows in our WSQoSX system based on centralized monitoring of non-functional requirements [7][8]. Furthermore, we investigated the interdependencies between Web service performance and the underlying network using a cross-layer monitoring approach [9].

A monitoring approach for functional as well as for non-functional requirements using WS-Agreement for the negotiation of requirements specified in various languages is discussed by [10] without a proof of concept. Lazovik et al. use business rules described in a proprietary language for monitoring of functional and non-functional requirements [11].

In addition to the monitoring approaches discussed above, we want to present different approaches to requirements definition. Baresi et al. provide an approach supporting both functional and non-functional requirements [12]. Their Web service constraint language allows the specification of user, provider, and third party requirements and its description in a WS-Policy compliant way. Ludwig et al. also use WS-Policy embedded in WS-Agreement to allow the description of requirements [10]. Each WS-Policy compliant language can be used in their CREMONA architecture.

But there are not only approaches using WS-Policy for requirements description. Similar to the work of Robinson discussed before, Sen et al. use past time linear temporal logic to describe monitoring requirements [13].

## III. Automated Monitoring and Alignment of Service-based Workflows – Our Approach

As we presented in the previous section, current approaches for monitoring do not or only basically support deviation handling. In order to react to deviations from requirements in a timely manner, efficient mechanisms for both monitoring and deviation handling have to be supported by the monitoring platform. Therefore, we propose AMAS.KOM, which supports integrated monitoring and alignment of service-based workflows, i.e., the handling of deviations and the re-fulfilment of SLAs after SLA violations to reach a proper system state again.

### *A. Requirements Analysis*

We had several requirements and restrictions while developing the AMAS.KOM approach in order to integrate the approach in existing service ecosystems. From the technological point of view, existing Web service standards should be used with only a minimum of needed modifications. Especially, the use of different Web service technologies (e.g., SOAP, REST, or XML-RPC) has to be supported. Furthermore, it has to be considered that monitoring itself often creates an overhead both in processing time and network traffic. In order to minimize the network traffic, the approach should support both centralized monitoring units, which are reporting back to a central instance, and decentralized monitoring units working on their own. Additionally, the automatic generation of the proactive monitoring units including deviation-handling mechanisms should be part of AMAS.KOM. Furthermore, flexibility with respect to the subjects of monitoring as well as the support of different requirement specification languages is needed. Here, both functional and non-functional monitoring requirements have to be supported.

From the business perspective, AMAS.KOM should provide a holistic approach supporting every step from the definition of business requirements to the generation and distribution of monitors. Support for different roles as well as views of the monitoring infrastructure and data should be part of the approach.

### *B. Our Approach in Detail*

The core principle of AMAS.KOM is the transformation of a service-based workflow description and related business requirements into a monitored instance of the workflow. Proxies are used to redirect service calls executed by the workflow engine to the monitored instances of a service. The monitoring itself is carried out in parallel to the service execution.

In order to create a monitored workflow instance a transformation of the workflow description and the business requirements is needed. We can distinguish four steps of the transformation process (cp. Fig. 1):

1. Annotation
2. Modification & Splitting
3. Generation
4. Distribution

The foundation of the process is a workflow description in a standardized form (e.g., WSBPEL) as well as a collection of business requirements in an arbitrary form. In the first step of the process, the *Annotation* step, the business requirements (focussing on the complete workflow) have to be described in form of a single policy document in machine-readable format following a given specification framework. For this, we developed the WS-Re2Policy language, which is discussed in the following section. In the *Modification & Splitting* step, the policy document is used to derive requirements for every single service, which is part of the service-based workflow. Therefore, QoS-aware

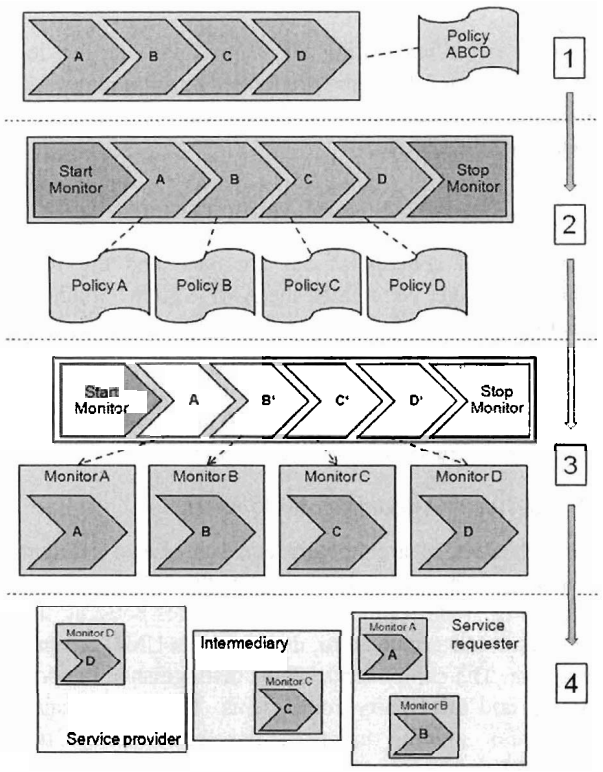


Fig. 1: The AMAS.KOM transformation process

composition algorithms can be used to create an execution plan of the workflow, which is able to fulfil the given requirements (e.g., [5][8]). During the *Generation* step, those policies are used to create both proxies as well as monitoring and alignment units. In order to avoid time-consuming planning for every single service, it is possible to reuse policies of various granularity as well as the resulting monitoring and alignment configurations. Both can be stored in a configuration database. In the last step, the *Distribution* step, the monitoring and alignment units have to be distributed in the infrastructure, based on the results of appropriate planning algorithms.

## IV. Modelling Monitoring Requirements and Possible Reactions to Deviations

In the previous section we discussed the process to generate workflows, which are monitored and support deviation handling. As a foundation of the approach, a description of the monitoring requirements in an adequate format is needed. Current requirements languages do not support deviation handling capabilities, allowing the automated generation of monitoring and alignment units. Those requirements languages often are strongly formalized languages (e.g., temporal logic) and do not support the specification of reactions. Furthermore, they only provide weak support for non-functional requirements. Additionally, strongly formalized languages are hard to use for non-experts.

In order to overcome those shortcomings and to allow the distribution of monitoring and alignment instructions in a service ecosystem, we developed the Web service requirements and reactions policy language (WS-Re2Policy), which will be discussed in the following sections.

### *A. Basic Concepts of the WS-Re2Policy Language*

The WS-Re2Policy language is based on the well-founded Event-Condition-Action (ECA) rules paradigm. We can map the elements of our language to the ECA paradigm as follows:

- Events: subjects to monitor, e.g., a performance figure of a workflow to be monitored.
- Conditions: thresholds for monitoring, e.g., an upper bound for the response time of a service.
- Actions: reactions to deviations, e.g., the triggering of replanning operations after the violation of a threshold.

The WS-Re2Policy language is designed as an extension to the World Wide Web Consortium's (W3C) WS-Policy 1.2 framework and is fully compliant to it. WS-Re2Policy is itself extensible by other WS-Policy compliant languages like WS-Trust or WS-SecurityPolicy. Every WS-Re2Policy compliant document consists of two parts, a requirements and a reactions part. As aforementioned, requirements can be described in any WS-Policy language. Our current approach supports simple QoS-related requirements by default in addition to the existing WS-Policy languages.

Fig. 2 further illustrates the WS-Re2Policy language containing the XML-based data model of our language. Here, the element *RequirementsReactionsSuite* defines an envelope for the requirements and reactions part ensuring the WS-Policy compliance.

In the WS-Re2Policy language, reactions are simple, easy to understand control constructs, which are implementation independent.

Currently, the following reactions are supported by the

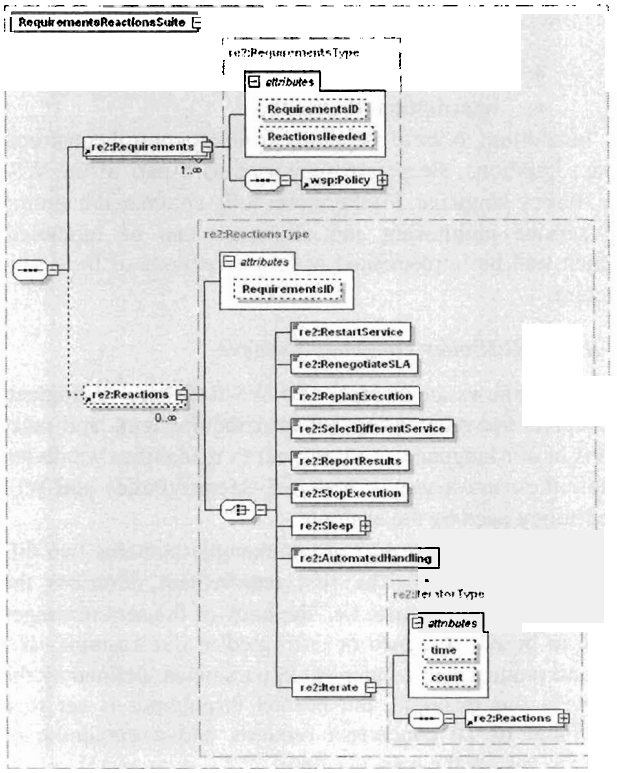


Fig. 2: The WS-Re2Policy data model

```
<?xml version="1.0" encoding="UTF-8"?>
<re2:RequirementsReactionsSuite ... >
  <re2:Requirements ReactionsNeeded="true" RequirementsID="1138">
    <wsp:Policy wsu:Id="ID_236" Name="SecurityQoSCombined">
      <wsp:All>
        <wsp:ExactlyOne>
          <wsp:All>
            <sp:SignedParts>
              <sp:Body/>
            </sp:SignedParts>
          </wsp:All>
          <wsp:All>
            <sp:EncryptedParts>
              <sp:Body/>
            </sp:EncryptedParts>
          </wsp:All>
        </wsp:ExactlyOne>
        <wsp:All>
          <qos:Throughput>10</qos:Throughput>
          <qos:ResponseTime>23.5557</qos:ResponseTime>
        </wsp:All>
      </wsp:All>
    </wsp:Policy>
  </re2:Requirements>
  <re2:Reactions RequirementsID="1138">
    <re2:Sleep time="10.0"/>
    <re2:Iterate time="0.0" count="2">
      <re2:RestartService/>
    </re2:Iterate>
    <re2:ReportResults/>
  </re2:Reactions>
</re2:RequirementsReactionsSuite>
```

Fig. 3: Example of a WS-Re2Policy document

WS-Re2Policy language:

- Restart of selected services
- Renegotiation of Service Level Agreements
- Replanning of execution plans
- Selection of different services based on various criteria
- Report results to caller or different third parties
- Interruption of execution

In addition to these reactions, different control constructs (e.g., iterations, sleep) are supported as a part of the WS-Re2Policy language. Furthermore, fully automated handling of service monitoring and alignment can be modelled, which will be implemented in future versions of the architecture.

### *B. WS-Re2Policy Language Example*

Fig. 3 shows an example of a WS-Re2Policy document. It depicts the relationship between requirements and reactions in our language. With respect to readability we do not quote the namespaces of both WS-SecurityPolicy and WS-Re2Policy used by the example.

The *requirements part* in the example contains two different requirements. The first requirement describes the needed security features, i.e., the body of the sent messages have to be either signed or encrypted in our example. The second requirement contains QoS parameters defined by the user. In our example, the needed throughput is set to a minimum of 10 concurrent requests and a maximum of 23.5557 ms for the response time.

The alignment instructions are described in the *reactions part* of the document. In our example, the monitor restarts the service twice after waiting for 10 ms following the detection of a deviation from the predefined requirements. Afterwards, the results of monitoring are always reported back to the caller of the service.

## V. The AMAS.KOM Architecture

As a proof of concept of our approach and the WS-Re2Policy language, we created the AMAS.KOM architecture (following the broker architectural style discussed by [14]) and implemented the architecture based on current Web service standards. The following sections discuss the overall architecture as well as its most important component, the *Monitoring & Alignment Manager*.

### *A. Architectural Overview of AMAS.KOM*

The AMAS.KOM architecture consists of six different components, which are needed to realize the specified functionalities. Fig. 4 shows the interdependencies between the components in our architecture, depicted as a UML component diagram. The components can be distinguished in core, supporting, and third party components. The core components contain unique functionalities realized by the AMAS.KOM architecture. The core components are:

- *AMAS Controller*: The controller provides all the transformation logic to create a monitored workflow and create service specific policy documents by splitting the global policy.
- *AMAS Repository*: A repository is used to store both policies and configurations of monitoring and alignment units based on XML documents.
- *Monitoring & Alignment Manager*: The component is responsible for the generation of monitoring and alignment units and their distribution in the service ecosystem. It will be discussed in detail in the following section.

In addition to the core components, the *(User) Interface* defines a supporting component of AMAS.KOM. The com-

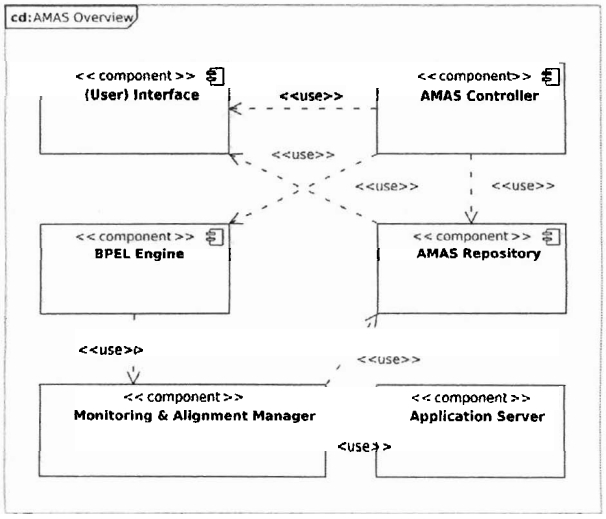


Fig. 4: Overview of the AMAS.KOM components

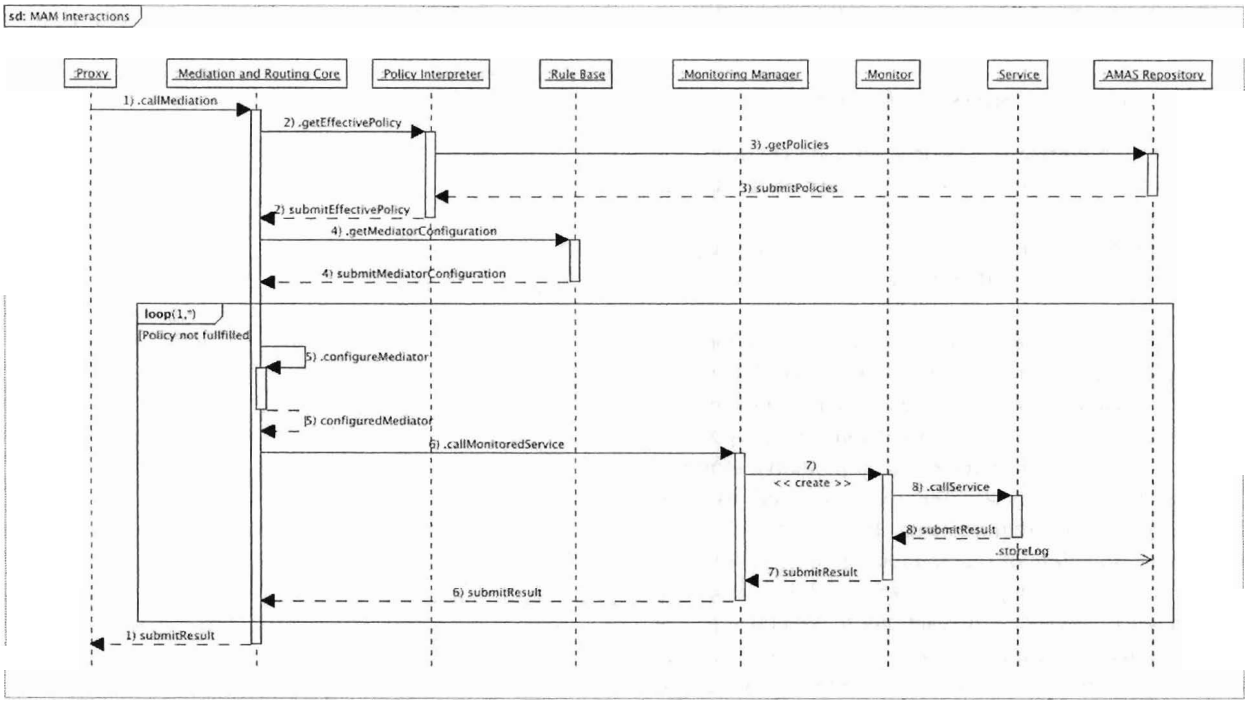


Fig. 5: Monitoring & Alignment Manager in detail

ponent offers different types of technical interfaces. It provides both a user interface to enter workflow descriptions as well as policies and an interface to interact with other systems, allowing the automated deployment of workflows.

Finally, the architecture includes third party components, providing state-of-the-art technologies for service provision and workflow execution:

- *BPEL Engine*: Workflows implemented as WSBPEL code can be executed using this execution engine.

  *Application Server*: Various servers are responsible for the provision of services used in the deployed workflow.

We are using standard software components in order to realize parts of our AMAS.KOM architecture, i.e., Apache AXIS and Synapse for Web service handling and mediation. This further allows us to easily integrate plenty of different Web service standards and related specifications in our architecture, i.e., WSBPEL 2.0 as workflow description, Web service description language (WSDL) 1.1, SOAP 1.2, and the REST approach as transport mechanisms. Additionally, WS-Policy 1.5 and WS-SecurityPolicy 1.1 are supported as policy formats.

### *B. Monitoring & Alignment Manager*

The previous section discussed the components of AMAS.KOM. In the current section, we present the *Monitoring & Alignment Manager* component in more detail as it provides the main functionalities of the architecture apart from the transformation realized by the *AMAS Controller*. Again, the *Monitoring & Alignment Manager* consists of a collection of interacting building blocks (*Proxy*, *Mediation and Routing Core*, *Policy Interpreter*, *Rule Base*, *Monitoring Manager*, and *Monitor*), which are depicted in Fig. 5. Services as well as the AMAS Repository are not part of the *Monitoring & Alignment Manager*.

The *Monitoring & Alignment Manager* is responsible for the interception of every single Web service call during runtime. For this, a *Proxy* receives calls of the workflow engine and redirects them to our *Mediation and Routing Core*, which will carry out further processing of the service invocation. The *Mediation and Routing Core* decides which service to call and what monitoring to use based on policy and configuration information stored in the *AMAS Repository* and the local *Rule Base*. After the retrieval of all policies with impact on the invoked service, the *Policy Interpreter* decides the effective policy for the current service invocation. The effective policy is the intersection of all policies, which are related to the selected service. Subsequently, the configuration of the *Mediation and Routing Core* is completed, so that the service invocation can be routed to an according *Monitoring Manager*. The *Monitoring Manager* is responsible for the generation of customized *Monitors*. A *Monitor* can be created individually for every service and its corresponding requirements, but the real benefit results from the reuse of monitoring configurations stored in the *AMAS Repository*. The *Monitor* calls the service and tries to fulfil the policy connected with the service. Monitoring results are again stored in the repository and can be used for further analysis. After policy fulfilment the control is passed back to the *Proxy*, which submits the result back to the workflow engine. A result is always passed back to the invoking party, even in case the service invocation failed completely. A policy is fulfilled even when the service invocation was not successful. In this special situation the alignment steps have to be applied successfully (but without

any effect to the result of the service invocation itself), in order to fulfil the given policy.

## VI. Conclusions and Outlook

Nowadays, enterprises collaborate in different scenarios by integrating third party services into their own business processes. In order to support the operation of so called cross-organizational workflows based on services, adequate monitoring mechanisms and support for SLA management is needed.

In this paper we presented an approach to the automated monitoring and alignment of service-based workflows, i.e., support for deviation handling in exceptional situations as well as the handling of SLA violations. Our approach AMAS.KOM can be characterized as a proactive monitoring approach and supports functional as well as non-functional monitoring requirements. In order to allow the simultaneous modelling of requirements as well as reactions to deviations, we developed the WS-Re2Policy language offering a set of pre-defined reactions to several types of possible deviations. Furthermore, the AMAS.KOM approach supports the distribution of monitoring and alignment units in a service ecosystem.

Currently, our research focus is on the efficient distribution of monitoring and alignment units in a service ecosystem. We are investigating where to place monitoring and alignment units in order to minimize the needed amount of time to react in case of a deviation. Therefore, we are currently working on the definition and solution of an optimization problem enabling a cost and response-time efficient distribution of those units.

Of similar importance to our work is the continuous enhancement of our policy language in order to support additional QoS requirements. In this context, we are planning to create a dedicated policy language for the specification of QoS requirements in service ecosystems, which would be useful not only for our own research and projects.

## VII. Acknowledgments

This work is supported in part by the E-Finance Lab e.V., Frankfurt am Main, Germany (http://www.efinancelab.com).

## VIII. References

[1] N. Repp, S. Schulte, J. Eckert, R. Berbner and R. Steinmetz, "An Approach to the Analysis and Evaluation of an Enterprise Service Ecosystem", in *Proceedings of the ICSOFT'07 Workshop on Architectures, Concepts and Technologies for Service Oriented Computing*, 2007, pp. 42-51.

[2] W.N. Robinson, "A requirements monitoring framework for enterprise systems", *Journal of Requirements Engineering*, vol.11, no. 1, 2005, pp.17-41.

[3] G. Spanoudakis and K. Mahbub, "Non Intrusive Monitoring of Service Based Systems", *International Journal of Cooperative Information Systems*, vol. 15, no. 3, 2006, pp. 325-358.

[4] L. Baresi and S. Guinea, "Towards Dynamic Monitoring of WS-BPEL Processes", in *Proceedings of the 3rd International Conference on Service oriented computing*, 2005, pp. 269-282.

[5] G. Canfora, M. Di Penta, R. Esposito and M.L. Villani, "QoS-aware replanning of composite Web services", in *Proceedings of the 3rd IEEE International Conference on Web Services*, 2005, pp. 121-129.

[6] A. Schmietendorf, R. Dumke and S. Stojanov, "Performance aspects in Web Service-based Integration Solutions", in *Proceedings of the 21st UK Performance Engineering Workshop*, 2005, pp. 137-152.

[7] R. Berbner, M. Spahn, N. Repp, O. Heckmann and R. Steinmetz, "Heuristics for QoS-aware Web Service Composition", in *Proceedings of the 4th IEEE International Conference on Web Services*, 2006, pp. 72-79.

[8] R. Berbner, T. Grollius, N. Repp, J. Eckert, O. Heckmann, E. Ortner and R. Steinmetz, "Management of Service-oriented Architecture (SOA)-based Application Systems", *Enterprise Modelling and Information Systems Architectures*, vol. 1, no. 2, 2007, pp. 14-26.

[9] N. Repp, R. Berbner, O. Heckmann and R. Steinmetz, "A Cross-Layer Approach to Performance Monitoring of Web Services", in *Proceedings of the IEEE ECOWS'06 Workshop on Emerging Web Services Technology*, 2006, pp. 19-30.

[10] H. Ludwig, A. Dan and R. Kearney, "Cremona: An Architecture and Library for Creation and Monitoring of WS-Agreements", in *Proceedings of the 2nd International Conference on Service oriented computing*, 2004, pp. 65-74.

[11] A. Lazovik, M. Aiello and M. Papazoglou, "Planning and monitoring the execution of web service requests", *International Journal on Digital Libraries*, vol. 6, no. 3, 2006, pp. 235-246.

[12] L. Baresi, S. Guinea and P. Plebani, "WS-Policy for Service Monitoring", in *Proceedings of the 6th Workshop Technologies for E-Services*, 2006, pp. 72-83.

[13] S. Sen, A. Vardhan, G. Agha and G. Rosu, "Efficient Decentralized Monitoring of Safety in Distributed Systems", in *Proceedings of the 26th International Conference on Software Engineering*, 2004, pp. 418-427.

[14] T.S. Dillon, C. Wu and E. Chang, "Reference Architectural Styles for Service-Oriented Computing", in *Proceedings of the IFIP International Conference of Networked and Parallel Computing*, 2007, pp. 543-555.