[RSS+09] Nicolas Repp, Dieter Schuller, Melanie Siebenhaar, Andre Miede, Michael Niemann, Ralf Steinmetz; **On distributed SLA monitoring and enforcement in service-oriented systems.** In: International Journal On Advances in Systems and Measurements, vol. 2, no. 1, June 2009. Seite 33--43

International Journal On Advances in Systems and Measurements, vol 2 no 1, year 2009, http://www.iariajournals.org/systems_and_measurements/

On distributed SLA monitoring and enforcement in service-oriented systems

Nicolas Repp, Dieter Schuller, Melanie Siebenhaar, André Miede, Michael Niemann, Ralf Steinmetz
Technische Universität Darmstadt
Multimedia Communications Lab
Merckstr. 25, 64285 Darmstadt, Germany
firstname.lastname@KOM.tu-darmstadt.de

Abstract

For the integration of systems across enterprise boundaries, the application of Web service technology and the Service-oriented Architecture (SOA) paradigm have become state of the art. Here, the management of the quality delivered by third party services is crucial. In order to achieve high service quality, requirements therefore need to be initially specified using Service Level Agreements (SLAs), which are later on monitored during runtime. In case of SLA violations, appropriate countermeasures have to be executed.

The paper presents an integrated approach for SLA monitoring and enforcement using distributed autonomous monitoring units. Additionally, the paper presents strategies to distribute those units in an existing service-oriented infrastructure based on mixed integer programming techniques. Furthermore, appropriate framework support is given by the AMAS.KOM framework enabling distributed SLA monitoring and enforcement based on the developed distribution strategies. As a foundation for our approach, the WS-Re2Policy language is presented, which allows the specification of both requirements with respect to service quality and the necessary countermeasures at the same time.

Keywords: Monitoring; SLA enforcement; Location strategies; Service-oriented Architectures.

1. Introduction

In recent years, solution as well as implementation means for all sorts of complex communications systems, e.g., in telecommunications or business automation scenarios, are addressed by propagating Web service technology and the Service-oriented Architecture paradigm. Especially in business automation scenarios, Web services and SOAs are used for the realization of cross-organisational collaborations between enterprises by integrating the business processes and IT systems of the business partners. Here, the Business Process Execution Language (BPEL) allows the composition of services of different business partners to business processes as well as the execution across enterprise boundaries.

Nevertheless, a collaboration of business processes composed of several individual services over the boundaries of an enterprise bears several challenges enterprises have to cope with. In order to build dependable and trusted business relationships QoS and security aspects need to be addressed within the integration of third party services into an enterprise's business processes and IT system. Due to SOA's loose coupling, which permits the selection of third party services at runtime, flexible and permanently changing business relationships have to be considered in particular. Therefore, the participating parties need to define both business requirements and responsibilities of the partners by negotiating contracts and Service Level Agreements (SLA) respectively. From a technological point of view, normally defined XML-based policy documents are used for the definition of SLAs and other requirements.

However, recent approaches for the definition of such policies are insufficient as they do only address the actual requirements. But with regard to SLA enforcement, monitoring of the requirements during runtime is crucial. Therefore, also adequate countermeasures need to be defined within the policy documents in order to restore compliance with the policies in case of deviations from the specified values. Especially in distributed scenarios it is further helpful to provide several independent monitoring units with the information about requirements and countermeasures in order to enforce policies at different locations in an infrastructure.

In order to overcome both issues, we developed the Web service requirements and reactions policy language (WS-Re2Policy) presented in this article, which specifies requirements and reactions in a single policy file. Therefore, it can be deployed to different monitoring units for distributed SLA monitoring and enforcement. Additionally, we present a framework named Automated Monitoring and Alignment of Services (AMAS.KOM), which supports the implementation of WS-Re2Policy in the areas of Web services and SOAs. This article is an extension to our work [1] presented at the ICSNC 2008 conference as well as to our research presented in [2] and [3]. It enhances our previous work with respect to the underlying agent-based monitoring framework as well as to the distribution mechanisms, which are used in our approach to place monitoring units in an infrastructure.

The remaining part of this article is structured as follows. In the next section, the overarching scenario of distributed SLA monitoring and enforcement is discussed in more detail. Subsequently, the WS-Re2Policy language is discussed in depth and explained by the use of an example. Afterwards, the AMAS.KOM framework is presented, which allows the use of WS-Re2Policy for Web service-based SOAs. The following section presents our work towards the optimal distribution of monitoring units in a distributed service-based infrastructure. Here, a distribution strategy is developed and evaluated by the use of simulation. Before the article closes with a conclusion and outlook, the related work to our research is presented.

2. Scenario and approach

In cross-organisational collaborations on the basis of an integration of business processes and IT systems, the "classical" scenario consists of a single enterprise and different business partners, in which the enterprise wants to use different third party services provided by the partners in a *1 to n* client-server style [4].

Here, a centralised monitoring and deviation handling approach is performed and carried out by the service requester (SR) itself (cf. Figure 1a). Thus, all other components, i.e., the monitoring units (MU) and decision making components, are located at the service requesters', even if monitoring data is sometimes collected by distributed probes. However, in large-scale SOA scenarios a centralised approach is not applicable, due to a vast amount of service requesters and providers (e.g., *m to n*) which bears scalability and complexity issues. But also unclear responsibilities between participating partners or a lack of privacy lead to legal and governance issues. Furthermore, the collection and availability of monitoring data required for decisions is hindered due to the existence of different spheres of control representing domains which belong only to single partners. Consequently, there exists no sufficient quality and amount of monitoring data, so that adequate decision making and timely handling of SLA violations cannot be performed.

For reasons already stated, we propose a distributed approach to SLA monitoring and enforcement, which overcomes the information deficit and improves the speed of information provisioning. Our approach is based on the application of decentralised monitoring and alignment agents (MAAs) which obtain both monitoring requirements and the specification of countermeasures. The MAAs are placed within the infrastructure at various places (cf. Figure 1b). Here, a hybrid approach, i.e., a mixture of centralised and decentralised interaction styles, is taken, instead of a fully decentralised approach (i.e., Peer-to-Peer).

A WS-Re2Policy compliant policy file enables the monitoring and alignment agents to manage single services as well as service compositions in a semi-autonomous way

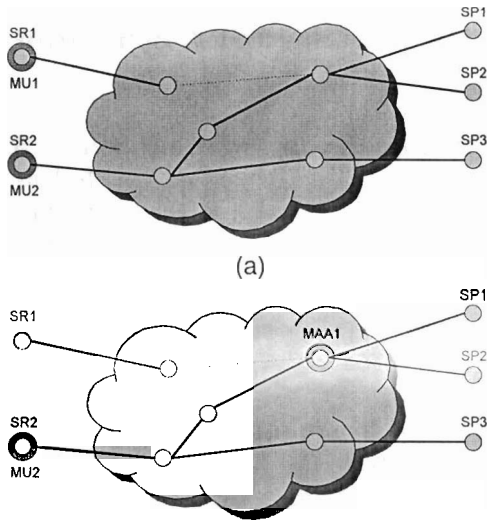

Figure 1. Monitoring styles (cf. [2])

according to the specified rules. Within a policy file agreed countermeasures in case of SLA violations are defined representing boundaries of the MAA's behaviour. Given that policies can be split into sub-policies, the corresponding subdivision of the MAA's behaviour forms the basis for new MAAs. Policies and MAAs can also be recombined in order to reduce the amount of MAAs up to a single instance. Using specialised communication mechanisms, MAAs can interact with each other, so that tasks can also be delegated between MAAs. Here, the communication protocols of the selected agent platform are facilitated (e.g., the Agent Communication Language). An example of cooperating agents based on our AMAS.KOM framework can be found in [5].

3. Web service requirements and reactions policy language

This section addresses various aspects of the WS-Re2Policy language in its most recent version. Starting from a theoretical point of view, a basic example is then used to discuss and explain the core elements of the language. For a discussion of a preliminary version of the WS-Re2Policy language we refer to [3].

3.1. Theoretical foundation of the WS-Re2Policy language

The well-founded Event-Condition-Action (ECA) rules paradigm, first discussed in the area of active databases (e.g., [6]), represents the basis of the WS-Re2Policy language. The

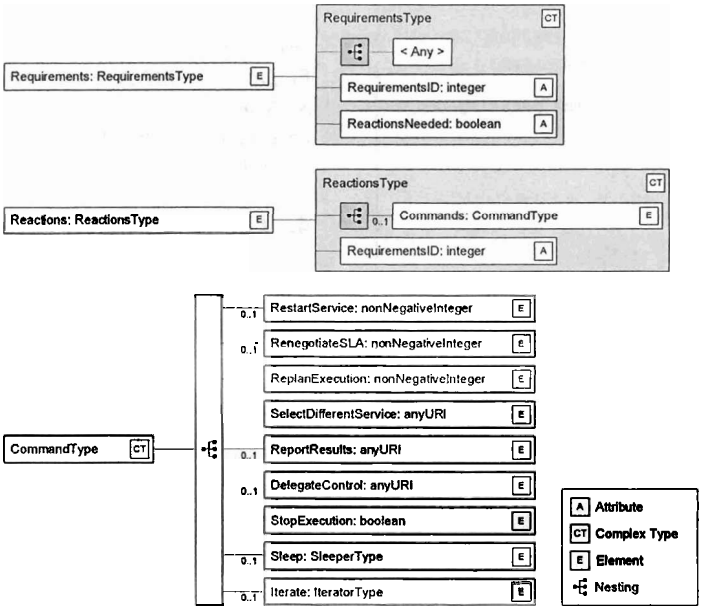

Figure 2. Core elements of the language

name ECA already states the main concepts of the ECA rules:

```
ON Event
IF Condition
DO Actions
```

Those concepts are directly used by the WS-Re2Policy language. Its elements can be mapped to those ECA concepts as follows:

Event: current measure of a monitoring subject, e.g., the response time of a service composition.

- Condition: threshold for the monitoring subject, e.g., the upper bound of the service's response time.
- Actions: reactions to an SLA violation aiming at the enforcement of the SLA, e.g., the restart of a service after a failure or time-out.

The use of ECA-styled rules as a basis for our WS-Re2Policy language has different advantages. In the first instance, the application of ECA rules does not require a very deep understanding which allows the generation of policy files even by non-experts with tool support. Furthermore, using a rule-based system the separation of control logic from the real implementation of an MAA is supported, thus allowing adaptability, which is crucial for our approach. Finally, a broad theoretical foundation exists, ranging from possible optimisations of rule-based systems to distributed problem solving strategies of autonomous units in distributed systems using ECA in combination with π-calculus [7].

3.2. Core elements of the WS-Re2Policy language

The WS-Re2Policy language was designed as an extension to the World Wide Web Consortium's WS-Policy 1.2 framework in order to use existing standards and be compliant to it. Therefore, the WS-Re2Policy can be extended by other WS-Policy compliant languages, e.g., WS-SecurityPolicy.

Basically, a WS-Policy compliant policy consists of two main parts: the requirements and the reactions part as depicted in Figure 2. For the description of requirements any WS-Policy compliant language can be used. Currently, some basic QoS parameters, e.g., throughput and response time, are natively supported by our approach. In future versions of the language, further QoS parameters will follow.

In the WS-Re2Policy language, reactions are simple and easy to understand control structures describing possible countermeasures in case of deviations from SLAs. At this stage, the WS-Re2Policy language as well as the AMAS.KOM framework support the following reactions:

- Restarting of a service, which violated an SLA.
- Renegotiation of SLA parameters for a single service or composition.
- Replanning of an existing execution plan for the composition a unit is responsible for.
  Selection of different services, which offer comparable functions.
  Reporting of results to caller or third parties.
- Delegation of control to other units on the same level

```
<?xml version="1.0" encoding="UTF-8"?>
<re2:RequirementsReactionsSuite ... >
 <re2:Requirements ReactionsNeeded="true"
      RequirementsID="3428">
  <wsp:Policy wsu:Id="ID_236" Name="Security-QoS">
   <wsp:All>
    <wsp:All>
     <sp:EncryptedParts>
      <sp:Body/>
     </sp:EncryptedParts>
    </wsp:All>
    <wsp:All>
     <qos:Throughput>10</qos:Throughput>
     <qos:ResponseTime>23.55ms</qos:ResponseTime>
    </wsp:All>
   </wsp:All>
  </wsp:Policy>
 </re2:Requirements>
 <re2:Reactions RequirementsID="3428">
  <re2:Sleep time="10.0ms"/>
  <re2:Iterate time="0.0ms" count="2">
   <re2:RestartService/>
  </re2:Iterate>
  <re2:DelegateControl>caller</re2:DelegateControl>
 </re2:Reactions>
</re2:RequirementsReactionsSuite>
```

Figure 3. A WS-Re2Policy compliant example

or to the central control instance without raising exceptions.

- Interruption of execution and passing back control by raising exceptions.

Additionally, the WS-Re2Policy language supports further control structures, e.g., loops (so-called iterations).

Finally, with regard to the connection between the parts of the WS-Re2Policy language and the ECA structures, the requirements parts contain the events, i.e., the subjects to monitor, and their corresponding conditions. The reactions part of the policy contains the specified actions. A reference key is used to interconnect reactions and requirements, so that reactions can be reused in different requirement parts.

3.3. WS-Re2Policy – a basic example

A simple example of a WS-Re2Policy compliant policy document is depicted in Figure 3. The namespace declarations of both WS-SecurityPolicy and WS-Re2Policy were removed in order to improve readability.

Within the requirements part of the example, requirements in two different WS-Policy compliant policy languages are defined. The first requirements element contains WS-SecurityPolicy information (cf. the following tag: $< sp : EncryptedParts >$) representing the required security features for interaction with the service in this case. The second requirement element of the example defines QoS parameters specifying a minimum of 10 concurrent service calls and a maximum of 23.55 ms for the response time.

The reactions part of the document specifies the countermeasures in case of SLA violations. In the example in Figure 3, the MAA restarts the service twice 10 ms after an SLA violation occurred. If no normal service operation can be established, the MAA raises no exception. Instead, the control is passed back to the caller for further handling.

4. The AMAS.KOM framework

As a proof of concept, we designed the AMAS.KOM framework and its underlying architecture. By supporting all phases ranging from the modelling of requirements to the enforcement of SLAs by MAAs, AMAS.KOM aims towards a holistic SLA monitoring and enforcement approach. For this purpose, AMAS.KOM offers a transformation of a business process description and associated requirements into a monitored process. Within the transformation, an indirection of service calls to the MAA infrastructure is integrated by analysing and adapting an existing process description in form of a Web service composition.

The transformation process consists of the four steps *modelling and annotation*, *modification and splitting*, *generation*, and *distribution*. In the first step, *modelling and annotation*, the requirements concerning the complete business process are specified by manually enhancing a description of a business process with SLA assertions. Here, the business process is described in the Business Process Modelling Notation (BPMN – cf. [8]) which represents a graphical specification of a business process. Later on, the SLA assertions are transformed into policy documents describing the requirements and countermeasures for the complete process using the WS-Re2Policy language. In order to support the user with the annotation, AMAS.KOM provides a Web-based wizard. Within the second step called *modification and splitting*, separate policies for each service or sub-process, depending on the planned granularity of MAAs, are derived using the global policy document. Here, various QoS-aware planning algorithms can be used for planning purposes in order to generate feasible partitions, e.g., as discussed in [9], [10]. This process step is carried out automatically and results in policy documents and execution plans. Both contain simple Web service calls in combination with a policy document and execution plans for sub-processes in combination with the related policy documents. The third step, *generation*, includes the creation of MAAs on the basis of the predefined policies. Afterwards, the MAAs are distributed within the monitoring and alignment infrastructure during the *distribution* step. Due to a plug-in concept, MAAs are highly extensible. Therefore, only the configuration of the plug-ins needed, as specified by the requirements in the policy before their distribution, is necessary. For distribution purposes, various algorithms can be used in AMAS.KOM, e.g., a random distribution algorithm or the use of heuristics for the solution of the MAA location problem. An optimal

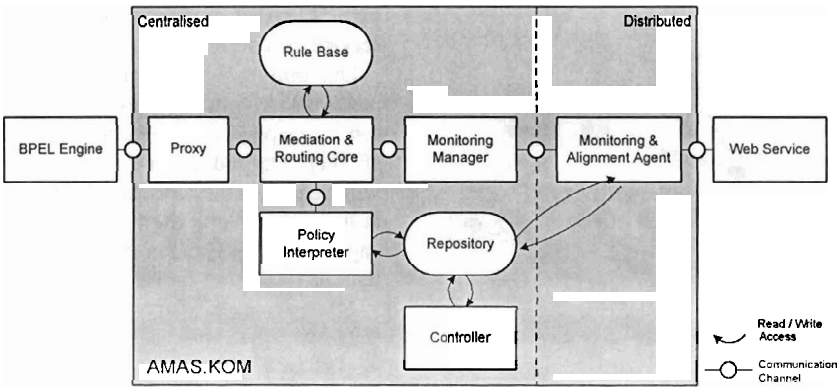

Figure 4. Overview of the AMAS.KOM framework

distribution strategy based on the analysis of given SLAs is presented in section 5.

The AMAS.KOM framework contains five core and various supportive components, like data storages for requirements, system configurations as well as monitoring data (cf. Figure 4). Subsequently, we will shortly discuss the core components:

- *Mediation & Routing Core*: combines both configuration information for MAAs as well as the applicable policies to specification sets and is furthermore responsible for routing both Web service call and specification set to Monitoring Managers.
- *Policy Interpreter*: calculates the effective policy, i.e., the policy out of a set of possible policies, which represents the current monitoring situation best.
- *Monitoring Manager*: generates MAAs based on a given specification set and distributes them in the infrastructure.
- *Controller*: provides the logic to build monitored processes by transforming complete requirement sets into service or sub-process specific policy documents.
- *Monitoring & Alignment Agent*: responsible for the actual monitoring as well as the execution of countermeasures.

The actual process of executing a monitored instance of a Web service call is described in the following. Generally, the Web service calls of the *BPEL Engine* are forwarded to a *Proxy* which redirects them to the *Mediation & Routing Core* component. For each Web service invocation, this component requests the effective policy from the *Policy Interpreter* which retrieves all applicable policies from the *Repository* and determines the effective one. Subsequently, the *Mediation & Routing Core* retrieves the associated configuration information of the service invocation from the *Rule Base*. Afterwards, the service invocation is routed to an appropriate *Monitoring Manager* which performs the generation of tailor-made monitoring units. The *Monitoring & Alignment Agent* performs the actual service call and tries to comply with the associated policy. For further evaluation purposes, the results are stored in the *Repository*. After the fulfilment of the policy, the result is passed back to the *BPEL Engine* via the *Mediation & Routing Core*. In case of an unsuccessful service invocation, appropriate alignment measures have to be accomplished.

We prototypically implemented the AMAS.KOM framework as a proof-of-concept for our distributed SLA monitoring and enforcement approach and the WS-Re2Policy language. Therefore, we used the JADE agent development framework to realise the MAAs, Apache Axis2 for Web service integration as well as the WSBPEL 2.0 standard for the specification of Web service-based collaborations. WS-Policy 1.5 and WS-SecurityPolicy 1.1 are also supported via the WS-Re2Policy language.

5. Distribution strategy for monitoring and alignment units

In this section, we introduce an approach to the distribution of MAAs in an infrastructure. As the installation of those monitoring and alignment components is associated with expenses, it should be avoided to distribute them randomly. Instead, we recommend to distribute the MAAs in a way that minimises the total costs, which can be divided in setup costs and costs for the communication between nodes.

5.1. Modelling basics and prerequisites

In order to model the distribution problem of MAAs, some assumptions have to be made.

The relationships between service requester, service providers, and potential intermediaries form a network, which can be modelled as an undirected graph. Intermediaries are all nodes in the communication path between service requester and providers. The communication between

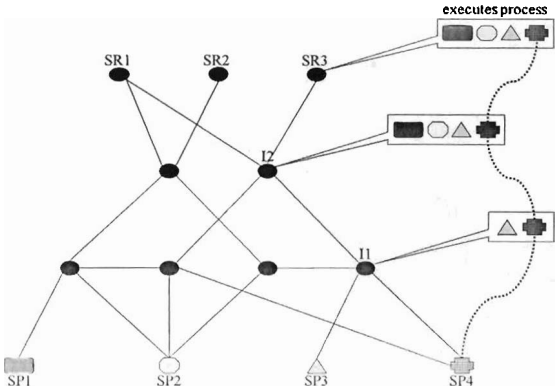

Figure 5. Example topology and routing

requester and providers forms a spanning tree, normally based on the shortest path from the requester to one provider. All intermediaries as well as the root itself can be seen as possible candidates for the execution of a monitoring and alignment unit.

The goal of our optimisation problem is the selection of those nodes, where the setup of monitoring and alignment units minimises the overall costs consisting of setup and communication costs. We call such an optimisation model *Monitoring Unit Location Problem* (MULP).

As constraints, we need to ensure that the alignment demands of all providers are satisfied. Here, the alignment demand defines the need for corrective actions with respect to a monitored service if an SLA is violated during service execution. The alignment demand is an abstract concept used for modelling, which can contain various elements like availability of a service or its error rate. As input for both the alignment demand calculation and the costs for communication as well as setup costs, the SLAs between the communicating parties are used.

In order to clarify the idea of the MULP as well as the concept of alignment demands, we present a simple example in Figure 5. As depicted in Figure 5, *Service Requester 3* (SR3) executes a process which is composed of services from different service providers – *Service Providers 1* to *4* (SP1 to SP4). If SR3 calls the service provided by SP4, the request and its corresponding response pass the *Intermediaries 1* and *2* (I1 and I2). Assume, that a MAA is installed on I1, which controls SP3 and SP4. In case the service provided by SP4 violates its current SLA, the MAA at I1 is able to detect this violation long before it is noted by the calling party. Furthermore, the MAA tries to correct the violation, e.g., by restarting the service, if the service was unavailable before. In the best case, the calling party does not even notice the problems. The alignment demand, which emerges from the SLA violation of the service, is satisfied by the MAA at I1, as it is responsible for the service provided by SP4. If I1 is unable to align the service, the control of the service execution is passed back to the calling party SR3.

Our optimisation problem, which determines the placement of MAAs in an infrastructure by reducing overall costs, can be mapped to a Warehouse Location Problem (WLP – cf. [11]). Depending on the existence of capacity restrictions with regard to the amount of supported alignment demands at an intermediary, the problem can be mapped to either a capacitated WLP or an uncapacitated WLP. In both cases, the corresponding optimisation problems are NP-hard.

Mixed integer programming techniques (cf. e.g., [12], [13]) can be applied for the purpose of solving the WLP-based *Monitoring Unit Location Problem* for arbitrary topologies. The resulting models can be solved using Branch-and-Bound algorithms afterwards.

5.2. Modelling the distribution strategy

Service requester, service providers and intermediaries are described by nodes of the network topology. Here, we can distinguish between the set of nodes that represent service providers (nodes $j \in M = \{1, ..., m\}$) and the set of nodes that represent service requesters and intermediaries (nodes $i \in N = \{1, ..., n\}$). By definition, only the service providers have alignment demands d_j, as already stated in the section before. Service providers are not allowed to directly execute monitoring and alignment units, because we only consider the requester perspective in our model. Service providers probably will carry out their own monitoring. For this, it is sufficient only to consider the nodes $i \in N$ when assigning setup costs. The actual costs for installing a MAA on node i are indicated by c_i^s. The cost of communication between node i (service requester or intermediary) and node j (service provider) are labelled with c_{ij}^c. In this context it is important to mention that the existence of an edge between two arbitrary nodes is not mandatory. Therefore, c_{ij}^c is defined as follows:

$$c_{ij}^c := \begin{cases} c_{ij}^{c*}, & \text{in case } (i,j) \text{ exists} \\ \min(c_{kj}^c + c_{ik}^{c**}), & \text{in case } (i,k) \text{ exists} \\ \infty, & \text{else} \end{cases}$$

Thereby, c_{ij}^{c*} describes the communication costs between intermediary i and provider j ($i \in N, j \in M$). Furthermore, c_{ik}^{c**} describes the communication costs between intermediary i and intermediary k ($i, k \in N$). These definitions ensure that c_{ij}^c specifies the minimum possible communication costs between intermediary i and provider j. The variables a_i constitute the capacities for nodes i. As already mentioned before, load restrictions could be seen as an example for such capacities. The decision variables y_i state, whether or not to install a MAA on node i, while the decision variables x_{ij} describe the communication at a given path assuring that the alignment

demands are satisfied. The definition of the required variables is also depicted in Table 1 and Table 2.

According to [11], the corresponding mathematical model of the MULP can be described as denoted in the following optimisation model:

Target function

$$\text{minimise } F(x,y) = \sum_{i=1}^{n} c_i^s y_i + \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}^c x_{ij} \quad (1)$$

subject to

$$\sum_{i=1}^{n} x_{ij} = d_j \quad \forall j \in M \quad (2)$$

$$\sum_{j=1}^{m} x_{ij} \leq a_i y_i \quad \forall i \in N \quad (3)$$

$$y_i \in \{0,1\}, x_{ij} \geq 0 \quad \forall i \in N, \forall j \in M \quad (4)$$

Finally, it is important that the MAA setup costs at the service requesters' are zero, i.e., monitoring and alignment can be done at the service requester's process engine without additional costs. Here, centralised monitoring is always enabled and taken into account.

As stated in the previous section, we use Branch-and-Bound algorithms to solve the WLP. For large topologies, this would require strong computational effort. In such a case, we propose to relax the integrity conditions and apply heuristics afterwards – as for example H1_RELAX_IP discussed in [9] – to get a valid solution with integer values for y_i. This heuristic does not perform significantly worse compared with the optimal solution with respect to its solution quality (cf. [9]).

In any case, the partitioning and distribution of the optimisation process improves the scalability of the complete system (e.g., the load situation at the service requester's QoS management system), because the computation of the distribution schemes is not only carried out by a single system but by all existing monitoring units for the areas they are responsible for. Furthermore, the distribution of the monitoring units and hence the computation of the distribution schemes into control spheres of third parties with no external access allows the application of our approach, e.g., in scenarios with high security demands.

6. Evaluation of the distribution strategy

In this section, we present the evaluation of the distribution strategy presented before. For this, different infrastructure types and configurations are simulated. As prerequisites, we present the scenarios used for simulation as well as the simulation setup in the following sections.

Table 1. Definition of variables

i:	service requester, intermediary
j:	service provider
d_j:	**alignment demand of provider j**
c_s^i:	costs for installing a MAA on node i
c_{ij}^c:	communication costs between i and j

Table 2. Definition of decision variables

y_i:	whether or not to install a MAA on i
x_{ij}:	alignment demand of j satisfied by i

6.1. Simulation scenarios and assumptions

The overall goal of the simulation is to show that the distribution of MAAs can lead to cost savings for a given distributed scenario. The following aspects were taken into account during the evaluation:

- A cost comparison of the optimal distributed solution with the centralised solution.
- The comparison of the number of MAAs with the overall nodes and service providers.
- The analysis of the execution time behaviour of our distribution strategy.

During simulation different parameters were adjusted and their impact on the overall performance was measured. The following parameters were used in the evaluation:

- *Topology parameters*, containing the number of nodes and connectivity parameters of the network, which are used to configure the Waxman algorithm.
- *Percentage of service providers*, defining the amount of service providers in relation to the overall nodes. The service providers are randomly distributed in the given topology.
- *Cost ratio*, which is defined as the ratio of setup costs for installing a MAA on a node to the maximum of the communication costs per link.

Based on those simulation parameters, several simulation scenarios can be specified. For this article, we focus on scenarios in which the topologies (as well as the related matrices) are sparse, i.e., they have a high degree of unconnected links. We assume different cost ratios in order to test extreme configurations of the simulation. Here, cost ratios of setup costs and maximum communication costs of 1:4, 1:1, and 4:1 were used. Furthermore, every scenario contains random topologies including 10, 20, ..., 100 nodes. Every type of scenario is executed 10 times with different configurations. For the analysis of the results the mean of all calculated values is used.

Additionally, we assume the capacity of the nodes to be infinite for the evaluation, so we apply an uncapacitated WLP. Furthermore, we only take one service requester into account. In addition, it is possible to imagine a completely

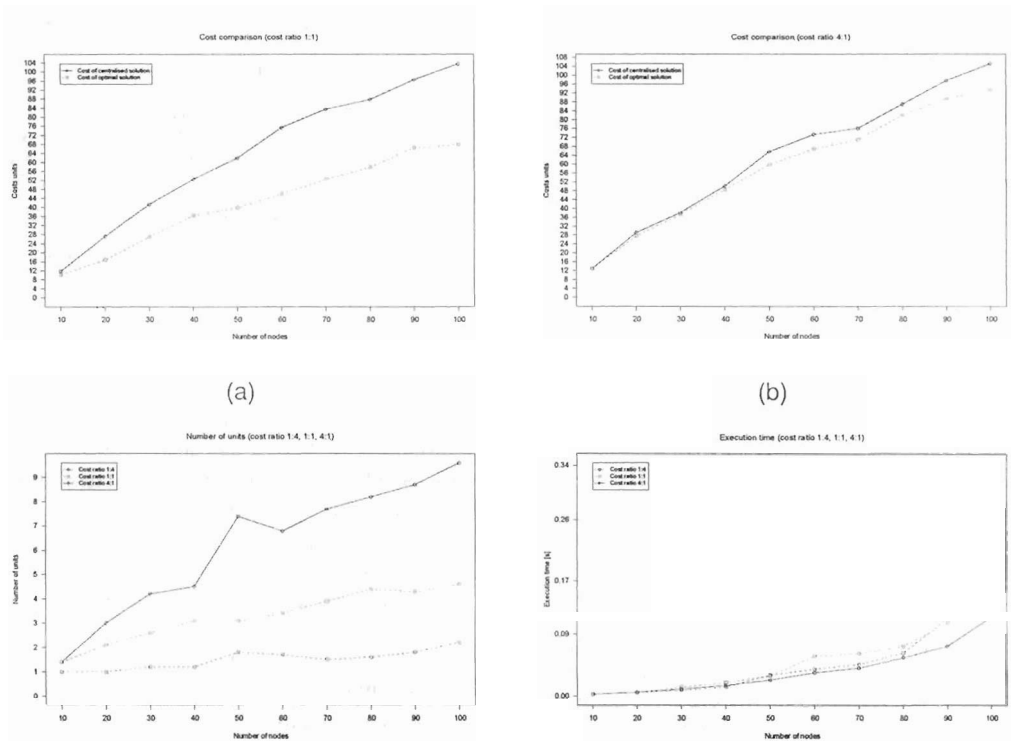

Figure 6. Cost savings, number of MAAs, and execution time samples

decentralised scenario in order to reach a high degree of scalability and robustness. However, concerning our simulation, we postulate that the alignment demand of one provider j has to be satisfied by a single node i having a MAA installed, which is located on the "shortest path" (with respect to our minimisation problem) from this provider j to the requester or by the requester itself. That means in the context of our work: $x_{ij} = d_j$.

6.2. Simulation setup

For simulation purposes, a workbench was set up which integrates topology and cost model generation with a solver capable of solving mixed integer programs as well as appropriate visualisation functionalities.

In detail, the topology generator BRITE (Boston University Representative Internet Topology Generator – cf. [14]) was modified in order to support the data formats needed by our model generator and the solver. BRITE supports the generation of topologies based on the Waxman- and the Barabasi-Albert methods – both of them well established in the research community. We used Waxman-generated topologies for the evaluation of our approach (cf. [15] for in-depth discussion of Waxman topologies), but our approach is not limited to those types of topologies. After topology generation, the topology description file is complemented by cost and demand vectors, which are generated randomly by our .NET based model generator. In order to generate real random values, the randomiser methods of the .NET cryptography API are used. The solution to the mixed integer program is calculated with the commercial XPress-MP solver (release 2007B). Results of the solver are prepared and visualised using the R statistics package (version 2.7.1). In order to automate the simulation runs, all components were linked together using shell scripts.

The simulation itself was run on an Intel Core 2 Duo 2.33 GHz computer with 2 GByte RAM running the Windows Vista operating system.

6.3. Discussion of results

In this section, we will discuss some of the results which were generated during the simulation runs.

Figure 6a shows a comparison of the overall costs of monitoring and alignment for different topologies ranging from 10 to 100 nodes. Here, 20 % of the nodes are service

providers and the cost ratio of setup costs to maximum communication costs is 1:1. As Figure 6a shows, almost every configuration has some optimisation potential by distributing monitoring and alignment units. The maximum is at 60 nodes with a 38.9 % cost reduction. At this point, 6 to 7 MAAs are needed on average to manage 12 service providers, which is depicted in Figure 6c.

Figure 6b again shows a comparison of overall costs, but in this case for a cost ratio of 4:1. Again, 20 % of the nodes are service providers and 100 random topologies ranging from 10 to 100 nodes were investigated. Figure 6b shows that cost savings are only marginal in this scenario. A maximum cost saving of 11.1 % on average can be achieved at 100 nodes. For this, at most two MAAs are needed as Figure 6c shows. Scenarios, which only place one MAA are centralised ones.

A drawback of our approach is the runtime behaviour of the mixed integer program. As noted before, a WLP like optimisation problem is NP-hard. As we can see in Figure 6d, the runtime of the solver for the mixed integer program grows exponentially – even for relatively small problem sets. Therefore, an application to real-time scenarios is only possible for small topologies – e.g., a topology of 1000 nodes takes more than 30 minutes to solve on our simulation system – or by applying heuristics, which are in scope of our further research.

7. Related work

Due to the amount of research areas touched in this article, related work out of different areas has to be taken into account. First, we discuss the area of specification languages as well as the corresponding monitoring systems, followed by the presentation of a selection of distributed monitoring approaches. Finally, we discuss the related work to the distribution of monitoring units in an infrastructure.

There are various approaches to specify monitoring requirements with respect to SLAs in the area of Web services and SOAs, almost all with appropriate system support. Robinson specifies functional monitoring requirements in temporal logic, without any support for the specification of countermeasures [16]. Again, using logic to specify functional monitoring requirements, Spanoudakis and Mahbub present a transformation of BPEL into event calculus, in which the requirements can be specified [17]. The specification of countermeasures is again not part of their approach. Sen et al. also use past time linear temporal logic for the description of monitoring requirements, without any support for countermeasure specification [18]. Furthermore, all of the logic-based approaches lack an easy readability by non-expert users.

Monitoring assertions, which are integrated in the form of pre- and post-conditions in BPEL, are discussed by Baresi and Guinea [19]. Also an approach by Baresi et al. is the Web service constraint language for the specification of functional and non-functional requirements [20]. It uses the WS-Policy language to specify requirements of users, providers, and third parties. Both approaches do not cover the handling of deviations or the specification of countermeasures, but offer framework support for integration. Lazovik et al. use business rules for the same purpose and with the same limitations [21].

The approaches presented above primarily focus on the specification of monitoring aspects. All of the examined policy and requirements languages above do not support the specification of countermeasures and therefore are not fully applicable to our scenario. An approach, which also covers basic policy enforcement aspects, is discussed by Ludwig et al. [22]. The authors use WS-Policy as a part of WS-Agreement to specify requirements in their CREMONA architecture. A focus of their work is on the initial creation and subsequent adaptation of agreements between different parties (i.e., during the negotiation of parameters), which include policy elements and their enforcement or renegotiation. A different policy language named CIM-SPL is presented in [23], which also supports the specification of countermeasures and therefore enables an enforcement of policies. CIM-SPL integrates the elements of the Common Information Model (CIM), an industry standard provided by the Distributed Management Task Force, into a policy language. The application of a heavy-weight standard like CIM as the foundation of a distributed decision making approach is currently under research.

An additional area of application for policy enforcement is the area of security. Sattanathan et al. discuss an architecture, which allows the securing of Web services by the use of adaptive security policies defining e.g., the security levels of incoming and outgoing Web service messages [24]. Here, security policies can be changed during execution time without changing the implementation. Furthermore, their architecture allows negotiation and reconciliation of security policies. Ardagna et al. also address policy enforcement issues with respect to the security of Web services [25]. In their work, an approach for the access control of Web services based on policies is presented, which also supports basic policy enforcement strategies.

Of further interest is the approach followed by Oracle with their Web Services Manager [26]. The Oracle Web Service Manager integrates centralised monitoring and policy enforcement with distributed information gathering of basic QoS parameters (e.g., response time), exceptions, and security aspects. Therefore, an architecture containing non-intrusive (i.e., gateways) and intrusive elements (i.e., agents running at the same application server as the Web service) was developed, which allows both basic reporting of monitoring results as well as the automatic selection and activation of policies based on given measurements.

With respect to distributed monitoring, there are different

approaches, which are in some parts comparable to our work, especially in their application of agent technology for the distribution of the monitoring logic. The integration of SNMP into an agent-based architecture for network management is discussed in [27]. Here, agents are responsible to collect the monitoring data from SNMP-capable data sources. Again for network management purposes, the work of [28] discusses a scalable framework based on mobile software agents. The approaches differ from our AMAS.KOM approach by not supporting deviation handling mechanisms. A further representative of agent-based monitoring approaches is discussed in [29], in which software agents act as area managers responsible for the monitoring and control of dedicated parts of a network. Area assignment is dynamic, allowing agents to adapt their zones during runtime and to migrate into the selected zone using agent mobility features.

Finally, some approaches to the distribution of monitoring units exist, which are related to our work presented in this article. All of the existing approaches are focusing on overall network monitoring, but not on monitoring of services in a SOA. An overview presenting applicable models and problem definitions for the location of network monitoring units is discussed in [30]. The authors analyse in detail what types of monitors can be matched to what kind of optimisation problems. Possible solutions to those problems are presented in addition. Furthermore, [31] present methods for the optimal positioning of monitoring units for network performance assessment. Hereby, the authors focus on the minimisation of the number of devices as well as finding their optimal location. Another approach to calculate the optimal number of monitoring units based on BGP (Border Gateway Protocol) data is presented by [32], where the authors give some theoretical foundation to define boundaries for the number of monitoring units needed in an Internet-like scenario. As a result, the authors claim that one third of all nodes in an arbitrary topology should execute monitoring functionalities in order to manage the complete infrastructure.

8. Conclusion and outlook

In this article, we presented an integrated approach for distributed SLA monitoring and enforcement in service-oriented systems. For this, we introduced the policy language WS-Re2Policy to specify requirements and countermeasures to SLA violations simultaneously. Additionally, the AMAS.KOM framework, supporting distributed monitoring and enforcement of SLAs in Web service-based cross-organisational collaborations as well as a distribution strategy to find the optimal locations of monitoring units in distributed scenarios were presented.

The AMAS.KOM framework utilises the mobile software agent paradigm to define autonomous monitoring and alignment units, which are distributed in a service-oriented system by applying our optimisation approach. The corresponding monitoring and alignment units are pre-configured using the WS-Re2Policy language. First performance evaluations showed that the overhead introduced by our agent-based approach is almost insignificant in a Web service scenario.

The evaluation of the distribution strategy proposed in this article, which calculates the optimal position of a monitoring unit with respect to the total cost, shows that cost improvements can be reached by distributing monitoring units in almost every scenario we investigated. Here, a realistic cost model is crucial because the cost ratio of setup costs to communication costs can limit potential benefits.

Currently, the WS-Re2Policy language exists in its second version and is implemented in a prototypical implementation of our AMAS.KOM framework. Nevertheless, the language is under continuous development. One of the planned major enhancements of WS-Re2Policy is the native support of various additional QoS-related parameters, as a common definition of a QoS policy is currently missing. Another focus of our ongoing work is on the improvement of the distribution strategy. As noted before, the current strategy is not applicable to large topologies under real-time conditions. At the moment, we are working on heuristics to improve the planning process.

Acknowledgements

This work is supported in part by E-Finance Lab Frankfurt am Main e.V. (http://www.efinancelab.com). In addition, parts of the research are carried out in the THESEUS TEXO project funded by means of the German Federal Ministry of Economy and Technology under the promotional reference "01MQ07012". The authors take the responsibility for the contents.

References

[1] N. Repp, A. Miede, M. Niemann, and R. Steinmetz, "Ws-re2policy: A policy language for distributed sla monitoring and enforcement," in *Proceedings of the 3rd International Conference on Systems and Networks Communications*, October 2008, pp. 256–261.

[2] N. Repp, "Monitoring of services in distributed workflows," in *Proceedings of the Third International Conference on Software and Data Technologies*, July 2008.

[3] N. Repp, J. Eckert, S. Schulte, M. Niemann, R. Berbner, and R. Steinmetz:, "Towards automated monitoring and alignment of service-based workflows," in *Proceedings of the IEEE International Conference on Digital Ecosystems and Technologies 2008*, 2008, pp. 235–240.

[4] M. Bichler and K.-J. Lin, "Service-oriented computing," *IEEE Computer*, vol. 39, no. 3, pp. 99–101, March 2006.

[5] A. Miede, J.-B. Behuet, N. Repp, J. Eckert, and R. Steinmetz, "Cooperation mechanisms for monitoring agents in service-oriented architectures," in *Proceedings of the 9th International Conference Wirtschaftsinformatik*, Februar 2009, pp. 749–758.

[6] J. Widom and S. Ceri, *Active Database Systems*, 1st ed. Morgan-Kaufmann, 1995.

[7] Y. Wei, S. Zhang, and J. Cao, "Coordination among multi-agents using process calculus and eca rule," in *Proceedings of the First International Conference on Engineering and Deployment of Cooperative Information Systems*, 2002, pp. 456–465.

[8] S. A. White and D. Miers, *BPMN Modeling and Reference Guide*, 1st ed. Future Strategies, 2008.

[9] 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.

[10] G. Canfora, M. D. Penta, R. Esposito, and M. L. Villani, "Qos-aware replanning of composite web services," in *Proceedings of the IEEE International Conference on Web Services*, July 2005, pp. 121–129.

[11] W. Domschke and G. Krispin, "Location and layout planning: A survey," *OR Spektrum*, vol. 19, pp. 181–194, 1997.

[12] Y. Pochet and L. A. Wolsey, *Production Planning by Mixed Integer Programming*, 1st ed. Springer, 2006.

[13] R. Sridharan, "The capacitated plant location problem," *European Journal of Operational Research*, vol. 87 (2), pp. 203–213, December 1995.

[14] A. Medina, A. Lakhina, I. Matta, and J. Byers, "Brite: An approach to universal topology generation," in *Proceedings of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems*, 2001.

[15] B. M. Waxman, "Routing of multipoint connections," *Journal on Selected Areas in Communications*, vol. 6, no. 9, pp. 347–352, December 1988.

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

[17] G. Spanoudakis and K. Mahbub, "Non intrusive monitoring of service based systems," *International Journal of Cooperative Information Systems*, vol. 15, no. 3, pp. 325–358, 2006.

[18] 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.

[19] 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.

[20] 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.

[21] 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, pp. 235–246, 2006.

[22] 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.

[23] D. Agrawal, S. Calo, K.-W. Lee, and J. Lobo, "Issues in designing a policy language for distributed management of it infrastructures," in *Proceedings of the 10th IFIP/IEEE International Symposium on Integrated Network Management*, 2007, pp. 30–39.

[24] S. Sattanathan, N. C. Narendra, Z. Maamar, and G. K. Mostéfaoui, "Context-driven policy enforcement and reconciliation for web services," in *Proceedings of the Eighth International Conference on Enterprise Information Systems: Databases and Information Systems Integration*, 2006, pp. 93–99.

[25] C. A. Ardagna, E. Damiani, S. D. C. di Vimercati, and P. Samarati, "A web service architecture for enforcing access control policies," in *Proceedings of the First International Workshop on Views on Designing Complex Architectures*, 2006, pp. 47–62.

[26] K. Chu, O. Cordero, M. Korf, C. Pickersgill, and R. Whitmore, *Oracle SOA Suite Developer's Guide*, Oracle, 2006.

[27] M. Zapf, K. Herrmann, and K. Geihs, "Decentralized snmp management with mobile agents," in *Proceedings of the 6th IFIP/IEEE International Symposium on Integrated Network Management*, 1999, pp. 623–635.

[28] D. Gavalas, D. Greenwood, M. Ghanbari, and M. O'Mahony, "Using mobile agents for distributed network performance management," in *Proceedings of the 3rd International Workshop on Intelligent Agents for Telecommunication Applications*, 1999, pp. 96–112.

[29] A. Liotta, G. Pavlou, and G. Knight, "A self-adaptable agent system for efficient information gathering," in *Proceedings of the 3rd International Workshop on Mobile Agents for Telecommunication Applications*, 2001, pp. 139–152.

[30] K. Suh, Y. Guo, J. Kurose, and D. Towsley, "Locating network monitors: Complexity, heuristics and coverage," *Computer Communications*, vol. 29, no. 10, pp. 1564–1577, June 2006.

[31] C. Chaudet, E. Fleury, I. G. Lassous, H. Rivano, and M.-E. Voge, "Optimal positioning of active and passive monitoring devices," in *Proceedings of the 2005 ACM conference on Emerging network experiment and technology*, 2005, pp. 71–82.

[32] J. D. Horton and A. López-Ortiz, "On the number of distributed measurement points for network tomography," in *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, 2003, pp. 204–209.