Nicolas Repp, André Miede, Michael Niemann, Ralf Steinmetz:
WS-Re2Policy: A policy language for distributed SLA monitoring and enforcement. In: IEEE Computer Society: Proceedings of the Third International Conference on Systems and Networks Communications , p. 256-261, IEEE Press, October 2008. ISBN 978-0-7695-3371-1.

WS-Re2Policy: A policy language for distributed SLA monitoring and enforcement

Nicolas Repp, André Miede, Michael Niemann, Ralf Steinmetz
Technische Universität Darmstadt
Multimedia Communications Lab
Merckstr. 25, 64285 Darmstadt, Germany
repp@KOM.tu-darmstadt.de

Abstract

Web service technology and the Service-oriented Architecture (SOA) paradigm have become state of the art for the integration of systems across enterprise boundaries. Here, a strong need for policies exists, which describe the Quality of Service delivered by third parties.

Current policy languages in the area of Web services and SOAs allow the specification of requirements with respect to the Quality of Service as well as the parameters, which should be monitored. They do not cover the countermeasures needed and accepted in case of requirement violations. Especially, in distributed scenarios it is helpful to provide the monitoring units with information about possible reactions to violations in order to enforce policies at the monitoring units. Therefore, we developed the Web service requirements and reactions policy language (WS-Re2Policy), which overcomes those issues by specifying requirements and reactions in a single policy to be distributed to independent monitoring units.

1. Introduction

In recent years, Web service technology and the Service-oriented Architecture (SOA) paradigm are propagated as both solution and implementation means for all sorts of complex communication systems, e.g., in telecommunications or business process automation scenarios. Especially in the latter, Web services and SOAs are used to build up cross-organisational collaborations between business partners by integrating the business processes and IT systems of the partners. Here, services of different business partners can be composed to business processes and executed across enterprise boundaries, e.g., using the Business Process Execution Language (BPEL).

Nevertheless, a collaboration based on the integration of business processes and IT systems also creates various challenges enterprises have to cope with. The integration of third party services into an enterprise's business processes and IT systems needs to address Quality of Service (QoS) as well as security aspects to build dependable and trusted business relationships. Especially, the dynamics of business relationships resulting from the selection of third party services at runtime, which is possible due to SOA's loose coupling, have to be considered. Therefore, contracts and Service Level Agreements (SLA) respectively have to be negotiated between the participating parties defining both business requirements and responsibilities of the partners involved. From a technological point of view, SLAs and other requirements are normally defined as XML-based policy documents.

However, it is not sufficient just to specify such requirements in form of policies as recent approaches do. In order to enforce SLAs, monitoring of the requirements during runtime is crucial. Moreover, in case of deviations from the values specified in the policies adequate countermeasures have to be selected and applied to restore compliance with the policies. Therefore, there is a need to specify accepted countermeasures as well in parallel with the requirements to monitor. Furthermore, in distributed scenarios like large-scale SOAs it is helpful to provide not only a single monitoring unit with the information about requirements and countermeasures but different independent monitoring units in order to enforce policies at different locations in an infrastructure.

In this paper, we present the Web service requirements and reactions policy language (WS-Re2Policy), which overcomes both issues by specifying requirements and reactions in a single policy and therefore can be deployed to different monitoring units for distributed SLA monitoring and enforcement. In addition, a supportive architecture to implement WS-Re2Policy in the areas of Web services and SOAs is presented. This paper is an extension of our pre-

vious work (cp. [11] [12]) with respect to a stronger focus on the policy language as well as improvements to it in comparison to our former work. Furthermore, the evolution of our distributed proxy architecture towards a mobile software agent-based approach is discussed.

The remaining part of this paper 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 Automated Monitoring and Alignment (AMAS.KOM) architecture is presented, which allows the use of WS-Re2Policy for Web service-based SOAs. Before the paper closes with a conclusion and outlook, the related work in policies for service monitoring is presented.

2. Scenario and approach

The "classical" scenario for cross-organisational collaborations based on an integration of business processes and IT systems consists of a single enterprise, which wants to use third party services, and different business partners providing those services in a *1 to n* client-server style [6].

Here, both monitoring and handling of SLA violations is centralised and carried out by the service requester (SR) itself (cp. Fig. 1). Monitoring units (MU) and decision making components are located at the service requesters, even if monitoring data is sometimes collected by distributed probes. However, a centralised approach cannot be used in large-scale SOA scenarios including a vast number of service requesters and providers (e.g., *m to n*) due to scalability and complexity issues. Furthermore, legal and governance issues do exist for example based on unclear responsibilities between the participating partners or a lack of privacy. Additionally, different spheres of control, i.e., "areas" which belong only to single partners, hinder the availability of monitoring data for decision making. That results into an insufficient quality and amount of monitoring data, which is needed for decision making and the timely handling of SLA violations.

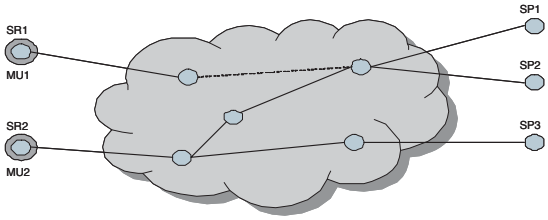

Figure 1. Monitoring of services in classical scenarios [11]

In order to overcome the information deficit and to improve the speed of information provisioning, we propose a distributed approach to SLA monitoring and enforcement. The basic idea, which is the motivation for the development of the WS-Re2Policy language as well as the foundation of our AMAS.KOM architecture, is the distribution of both monitoring requirements and the specification of countermeasures to decentralised monitoring and alignment agents (MAA) located at various places in an infrastructure (cp. Fig. 2). Here, no fully decentralised approach (i.e., Peer-to-Peer) is taken but a hybrid approach supporting a mixture of centralised and decentralised interaction styles.

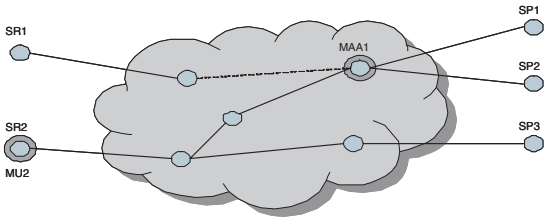

Figure 2. Monitoring of services in a distributed fashion [11]

Monitoring and alignment agents can manage single services as well as service compositions in a semi-autonomous way according to rules defined by a WS-Re2Policy compliant policy file. A policy describes boundaries of the MAA's behaviour as a collection of agreed countermeasures in case of SLA violations. Policies and therefore the behaviour of MAAs can be split into sub-policies, which again form the basis for new MAAs. It is also possible to recombine policies and MAAs to reduce the amount of MAAs up to a single instance. MAAs can interact with each other using specialised communication mechanisms. Therefore, tasks can also be delegated between MAAs.

3. Web service requirements and reactions policy language

In this section, we discuss various aspects of the WS-Re2Policy language in its most recent version. Based on a theoretical foundation, the core elements of the language are discussed and explained by a basic example. For a discussion of a preliminary version of the WS-Re2Policy language we refer to [12].

3.1. Theoretical foundation of the WS-Re2Policy language

The WS-Re2Policy language is based on the well-founded Event-Condition-Action (ECA) rules paradigm, which was first discussed in the area of active databases (e.g., [18]). As the name ECA already explains, the main concepts of ECA rules are the following:

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

WS-Re2Policy makes direct use of those concepts. The elements of our language 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 a 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 for our language has different advantages. In the first instance, ECA rules are easy to understand and to use. Our goal is to allow the generation of policy files even by non-experts with and without tool support. Furthermore, a rule-based system support the separation of control logic from the real implementation of an MAAs and therefore allows adaptability, which is crucial for our approach. Finally, there is a broad theoretical foundation, 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 [17]. The applicability of π-calculus to WS-Re2Policy is currently under investigation.

3.2. Core elements of the WS-Re2Policy language

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

The two main parts of a WS-Re2Policy compliant policy are the requirements and the reactions part (cp. Fig. 3). Requirements can be described in any WS-Policy compliant language. Currently, our approach natively supports some basic QoS parameters like throughput and response time. Further QoS parameters will follow in future versions of the language.

Reactions on the other hand are simple and easy to understand control structures, describing possible countermeasures in case of SLA violations. The following reactions are currently supported by the WS-Re2Policy language and also supported by the AMAS.KOM architecture:

- Restarting of a service, which violated a SLA.

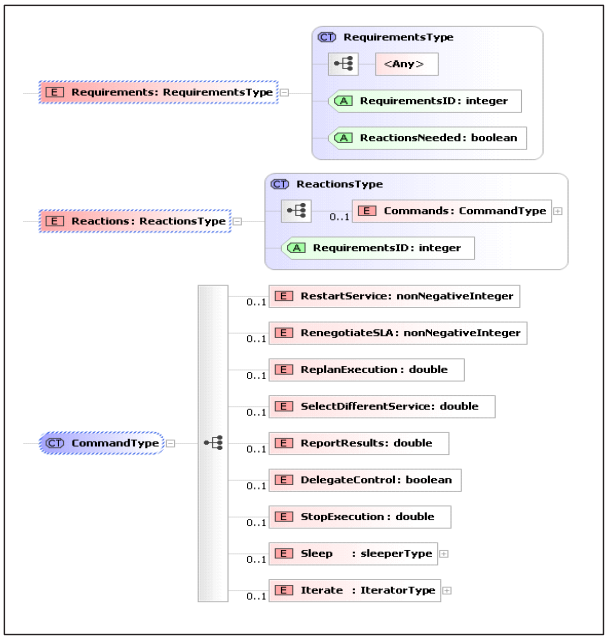

Figure 3. Core elements of the language

- 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 or to the central control instance without raising exceptions.
- Interruption of execution and passing back control by raising exceptions.

Furthermore, WS-Re2Policy supports additional control structures like loops (so called iterations).

Finally, the connection between the ECA structures discussed in the last section and the parts of the WS-Re2Policy language should be mentioned. The requirements parts contain the events, i.e., the subjects to monitor, as well as the conditions to be met. Actions are defined in the reactions part of the policy. Both are connected by a reference key, allowing reuse of reactions in different requirements parts.

3.3. WS-Re2Policy – a basic example

Fig. 4 contains a simple example of a policy document, which is compliant to the WS-Re2Policy language specification. In order to improve readability, we removed

```
<?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 4. A WS-Re2Policy compliant example

the namespace declarations of both WS-SecurityPolicy and WS-Re2Policy.

The requirements part of the example contains requirements defined in two different policy languages, which are both WS-Policy compliant. The first requirements element contains WS-SecurityPolicy information (cp. the following tag: $< sp : EncryptedParts >$), in this case the security features needed for the interaction with the service. The second requirements element defines QoS parameters, which are natively supported by WS-Re2Policy. Here, a minimum of 10 concurrent service calls as well as a maximum of 23.55 ms for the response time are specified.

The countermeasures are specified in the reactions part of the document. In the example depicted in Fig. 4, the MAA restarts the service twice after waiting for 10 ms following a SLA violation. In case no normal service operation could be established, the MAA will pass the control back to the caller for further handling without raising an exception.

4. WS-Re2Policy – overarching architecture

In order to implement our distributed SLA monitoring and enforcement approach, we designed the Automated Monitoring and Alignment (AMAS.KOM) architecture. AMAS.KOM aims towards a holistic SLA monitoring and enforcement by supporting all phases beginning with the modelling of requirements to the enforcement of SLAs by MAA. Therefore, a process transforming both business process descriptions and requirements into monitored process instances is offered by AMAS.KOM. In more detail, an existing process description in form of a Web service composition is analysed and adapted in order to integrate an indirection of service calls to the monitoring and alignment infrastructure.

The steps of the transformation process are called ”modelling and annotation”, ”modification and splitting”, ”generation”, and ”distribution”. In the ”modelling and annotation” step, a graphical representation of a business process conforming to the Business Process Modelling Notation (BPMN) is manually enhanced by SLA assertions and later on transformed into policy documents describing the requirements and countermeasures for the complete process using the WS-Re2Policy language. Therefore, AMAS.KOM offers a Web-based wizard to support the user with the annotation. During ”modification and splitting”, requirements for single services or sub-processes are derived depending on the planned granularity of MAAs. Here, for planning purposes various QoS-aware planning algorithms can be used to generate feasible partitions, e.g., as discussed by [5] [7]. This process step is carried out automatically. Its results are policy documents and execution plans, which contain both simple Web service calls in combination with a policy document and execution plans for sub-processes in combination with the related policy documents. In the ”generation” step, MAAs are created based on the policy information by the monitoring and alignment infrastructure and are distributed in the infrastructure during the ”distribution” step. The generation of MAAs during the ”generation” step depends on the requirements defined in the policy documents. MAAs are highly extensible based on a plug-in concept, which allows the configuration of only the plug-ins needed (as specified by the requirements in the policy) before their distribution. 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.

The AMAS.KOM architecture contains five core components, enhancing our previous work in [11]:

- AMAS Modeller: allows the annotation of both BPMN and BPEL process descriptions by SLA assertions using a Web-based user interface.
- AMAS Controller: provides the logic to build monitored processes by transforming complete requirement sets into service or sub-process specific policy documents.
- AMAS Repository: stores policies and configurations of the overall system and the MAAs.
- Monitoring and Alignment Manager: generates MAAs and distributes them in the infrastructure.
- Monitoring and Alignment Agents: responsible for monitoring and the execution of countermeasures.

We prototypically implemented the AMAS.KOM architecture as a proof-of-concept for our distributed SLA monitoring and enforcement approach and the WS-Re2Policy language. Therefore, we use the JADE agent development framework to realise the MAA, Apache Axis 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.

For an in-depth discussion of the overall architecture we refer to [11] and [12].

5. Related work

There are various approaches to specify monitoring requirements with respect to SLAs in the area Web services and SOAs. Robinson specifies functional monitoring requirements in temporal logic, without any support for the specification of countermeasures [13]. 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 [16]. 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 [15]. 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 [3]. Also an approach by Baresi et al. is the Web service constraint language for the specification of functional and non-functional requirements [4]. 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. Lazovik et al. use business rules for the same purpose and with the same limitations [9].

The approaches presented above primarily focused 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. [10]. 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 [1], 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 [14]. 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 [2]. 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 in their Web Services Manager [8]. 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.

6. Conclusion and outlook

In this paper, we presented the WS-Re2Policy language, which can be used to specify monitoring requirements and countermeasures to SLA violations simultaneously. Its area of application is the distributed monitoring and enforcement of SLAs in Web service-based cross-organisational collaborations. Furthermore, we presented AMAS.KOM as an overarching architecture, which facilitates the WS-Re2Policy language in the pre-mentioned Web services scenarios.

Currently, the WS-Re2Policy language exists in its second version and is implemented in a prototypical implementation of our AMAS.KOM architecture. 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. Furthermore, the splitting, recombination, and syncing of policies is under strong research in order to allow a conflict-free distribution of policies in an infrastructure.

Finally, we are currently investigating the performance

of AMAS.KOM as well as the overhead introduced by our agent-based approach and potential improvements.

Acknowledgements

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

Finally, we thank the reviewers of this paper for their excellent feedback.

References

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

[2] 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*, pages 47–62, 2006.

[3] L. Baresi and S. Guinea. Towards dynamic monitoring of ws-bpel processes. In *Proceedings of the 3rd International Conference on Service oriented computing*, pages 269–282, 2005.

[4] L. Baresi, S. Guinea, and P. Plebani. Ws-policy for service monitoring. In *Proceedings of the 6th Workshop Technologies for E-Services*, pages 72–83, 2006.

[5] 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*, pages 72–79, 2006.

[6] M. Bichler and K.-J. Lin. Service-oriented computing. *IEEE Computer*, 39(3):99–101, March 2006.

[7] 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*, pages 121–129, July 2005.

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

[9] A. Lazovik, M. Aiello, and M. Papazoglou. Planning and monitoring the execution of web service requests. *International Journal on Digital Libraries*, 6(3):235–246, 2006.

[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*, pages 65–74, 2004.

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

[12] 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*, pages 235–240, 2008.

[13] W. Robinson. A requirements monitoring framework for enterprise systems. *Journal of Requirements Engineering*, 11(1):17–41, 2005.

[14] 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*, pages 93–99, 2006.

[15] 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*, pages 418–427, 2004.

[16] G. Spanoudakis and K. Mahbub. Non intrusive monitoring of service based systems. *International Journal of Cooperative Information Systems*, 15(3):325–358, 2006.

[17] 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*, pages 456–465, London, UK, 2002. Springer-Verlag.

[18] J. Widom and S. Ceri. *Active Database Systems*. Morgan-Kaufmann, 1995.