A Case Study on Managing SLAs in Composite Services with COSMA

André Ludwig, Thomas Hering, Rolf Kluge, Bogdan Franczyk

Information Systems Institute
University of Leipzig
Marschnerstr. 31
04109 Leipzig, Germany
{ludwig,hering,kluge,franczyk}@wifa.uni-leipzig.de

Abstract: COSMA proposes a novel SLA management approach for composite services. It supports a composite service provider in managing SLAs with providers of atomic services, in managing SLAs with requesters of composite services, and aligning both SLA management activities with each other. On this basis, a composite service provider can control and optimize its composite SLA management activities during the entire SLA lifecycle. This includes, in particular, planning and negotiating SLAs, monitoring and evaluating SLAs. In this paper, a case study on managing SLAs in composite services with the COSMA approach is presented in detail.

1 Introduction

Service-oriented computing (SOC) has emerged as the most promising design paradigm for next-generation distributed information systems. The vision that goes along with SOC is that once standards have established themselves and become widely adopted by service providers and requesters, a globally available infrastructure for hosting and accessing services will be created [Pap07]. This infrastructure will allow service providers to offer multiple services with individually adapted service capabilities to their changing customers that can dynamically and on-demand bind these services into their own applications; thus forming a market of services and an Internet of Services, respectively [Rug07]. The advent of service markets on the basis of the SOC paradigm will pave the way for a service-oriented business model which is referred to as composite service provider (CSP) [ACKM04]. A composite service provider requests (atomic) services from atomic service providers (ASP) and provides these services according to a process flow as (composite) service to service requesters (SR). In this constellation, a composite service provider acts as an independent, self-interested business entity, motivated to fulfill own goals, i.e. be profitable and maximize customer satisfaction.

In order to control the interface between service requesters and providers, a contractual basis in form of service level agreements (SLA) is needed. Current SLA management approaches applicable for SOC environments, i.e. WSLA [KL03], WS-Agreement

[ACD+], or WSOL [TPP02], provide extensive SLA language formalizations and management frameworks. However, they focus on bi-lateral service requester/provider constellations neglecting the SLA management requirements of composite service providers, i.e. managing SLAs with atomic service providers and with composite service requesters and aligning both with each other. A SLA management solution for composite services has to consider the contribution of sourced services - formalized in their (atomic) SLAs (ASLA) - in the management of the provided service - formalized in its respective (composite) SLA (CSLA). Since composite services are created on-the-fly also their SLA management must be realized on-the-fly. Manual SLA management is not appropriate for CSPs and automation support is required, i.e. for creation, monitoring and evaluation of SLAs.

The COmposite Sla MAnagement (COSMA) approach provides a novel solution for CSPs in managing ASLAs, CSLAs and their alignment. On this basis, a CSP can control and optimize its composite SLA management activities during the entire SLA lifecycle. COSMA and its constitutional elements COSMAdoc, COSMAframe, and COSMAlife are presented in detail in [LF08a]. Complementary to this presentation, this paper illustrates the application of the COSMA approach on a case study (based on a use case presented in [MGL+07]). With this motivation, it aims at contributing to the overall understanding and enhancement of the approach.

The paper is organized as follows: section 2 gives a brief overview of COSMA and its constitutional elements. Section 3 introduces a general use case and demonstrates the application of COSMA throughout the main phases of the SLA lifecycle. Section 4 refers to a prototypical implementation of the COSMA approach and section 5 concludes the paper and gives an outlook to next steps.

2 Composite SLA Management Approach (COSMA)

The central idea behind the COSMA approach is the integration of all SLAs a composite service provider has to deal with into one composite SLA management document. This composite SLA management document, which is defined as COSMAdoc, contains all contractual information of all involved SLAs and in addition the relationships and dependencies that exist between the different aspects of atomic and composite SLAs. On the basis of a COSMAdoc, the SLA management system of a CSP is able to map atomic SLA contents to contents of the composite SLA. This mapping enables a CSP to control and optimize its SLA management activities in providing a composite service. This includes, in particular, planning and negotiating SLAs, monitoring and evaluating SLAs (cf. SLA lifecycle as outlined in [LBKF05]). The COSMA approach consists of the following three parts:

- COSMAdoc: A generic information model that integrates contractual data, SLA management data, and elements for the expression of dependencies and relationships between SLA elements.

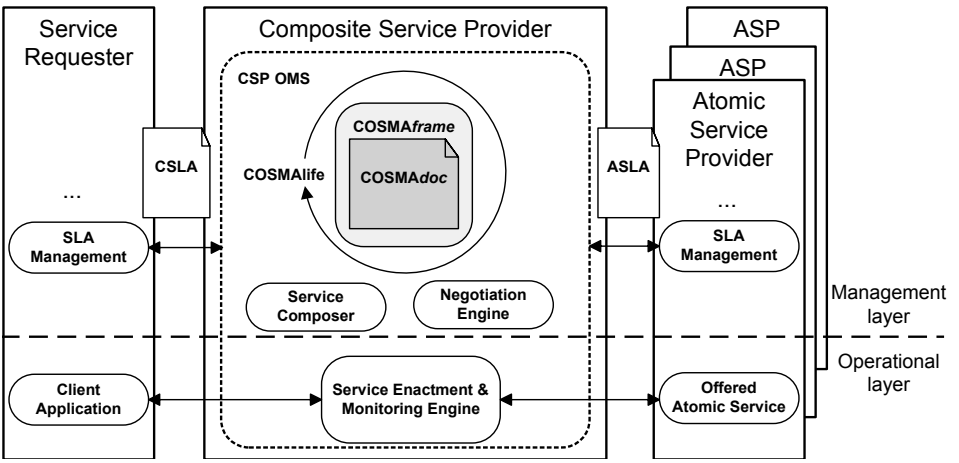


Figure 1: High-level perspective onto a CSP using the COSMA approach

- COSMAframe: A conceptual framework that outlines components that are necessary for the management of the composite SLA lifecycle on the basis of a COSMAdoc instance.
- COSMAlife: An integrated set of SLA management practices that use COSMAdoc instances to cover the phases of the SLA lifecycle.

COSMAframe, COSMAdoc, and COSMAlife are embedded into an operation and management system (OMS) of a CSP. In particular, the OMS provides components for automated run-time service composition (service composer), agent-based negotiation of SLAs (negotiation engine), enactment and monitoring of composite services (service enactment and monitoring engine), service registration, and interaction with service requesters and atomic service providers. In the Adaptive Services Grid project [ASG], all of these components including prototypical implementations were developed. Fig.1 presents the high-level perspective onto the business model composite service provider utilizing the COSMA approach.

3 Use Case

The use case presented in this paper is based on a business-to-business wholesale model of an Internet service provider (ISP). The ISP offers the whole spectrum of Internet services for Webspace provisioning. The ISP acts as a composite service provider. It integrates externally provided atomic services at run-time into end-to-end composite services.

On the supplier side, the ISP uses several atomic services, like domain name checking and registration, Web hosting configuration, operation and maintaining of Domain Name

Server information, handling of payments bundled to end customer products, messaging, and so forth. All of these services are provided by external atomic service providers, i.e. Web hosting ASP or payment ASP. Each service can be provided by different ASPs in different implementations with different functional and non-functional characteristics. It is also possible that an ASP provides the same service implementation with different characteristics depending on the individual requirements to its customers. For example, the service used for domain registration may be provided by several service providers in diverse implementations for different top-level domains (functional characteristic) at different service qualities (non-functional characteristics).

On the sales side, customers of the ISP are business units which request Internet service packages. These requested Internet services are further sold to end customers. Thus, customers of the ISP are resellers of Internet services again. These resellers bundle their services with Internet services sourced from the ISP. Typical examples of reselling customers of the ISP are portal providers like newspapers, TV-stations, or information portals. Portal providers bundle their core products, such as newspapers, with Internet services in order to gain higher profits and customer binding.

The ISP as an integrator of externally provided atomic services into end-to-end composite services has to determine at runtime which atomic service implementations are used in the composite service and has to manage the service provision in an automated fashion. It uses components such as a service composer, a negotiation engine, and a service enactment and monitoring engine. For the management of SLAs involved in the composite service it uses a component utilizing the COSMA approach which will be described in detail below.

The concrete composite service that is presented in the use case is a dynamic supply chain for automated Domain Name registration and provisioning of Webspace (later on referred to as DSC service). Atomic services included in the DSC service are:

- CheckDomain: checks whether a certain domain is available for registration
- CreateWebhostingAccount: creates a Web hosting account to access the Webspace
- CreateDomain: creates a domain for the Webspace
- SetupPhysicalHosting: sets up the physical Web hosting/Webspace which hosts the displayed files of a Website owner
- Nameserver: updates the name server with the new domain connected to the created Webspace
- RegisterDomain: registers a certain domain
- CreditCardPayment: executes a complete credit card authorization and payment process

The DSC service and all atomic services are characterized by service parameters which are defined in equal measures. They include quality parameters such as response time, availability, throughput, and encryption level, and financial terms such as reward. Fig.2

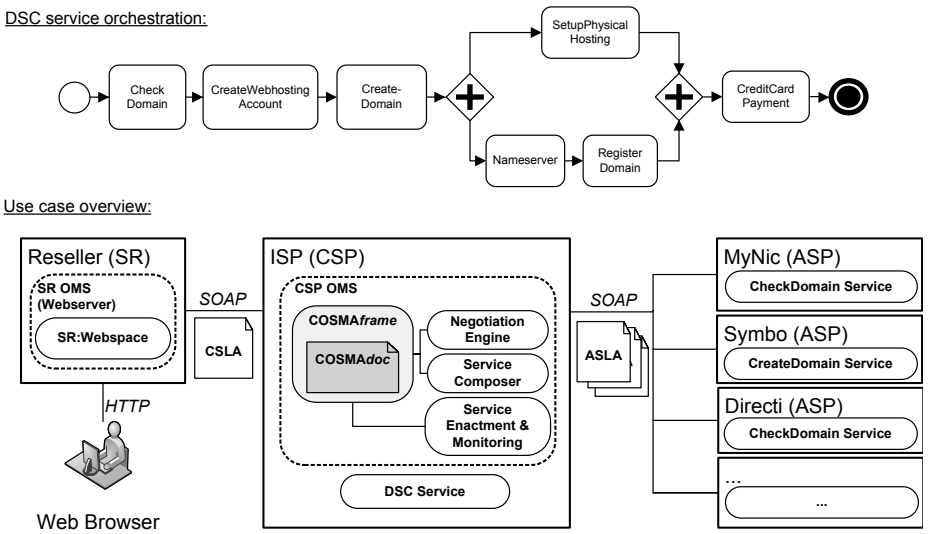


Figure 2: High-level use case perspective

shows the service orchestration composed by the service composer and illustrates the high-level perspective onto the use case.

The use case is based on a number of assumptions and restrictions. Firstly, all parties need to set up a negotiation environment which allows the creation and configuration of negotiation agents that negotiate SLA documents dynamically and that are allowed to sign resulting SLAs. Questions of interoperable negotiation environments, the ability to communicate and understand each other and so on are not tackled. Secondly, it is assumed that all negotiated SLA parameters are defined equally or can at least mapped to a common definition. Thirdly, the whole area of semantic service descriptions, creating them, processing them and using them for registering atomic services or composing service orchestration scripts is blinded out.

3.1 Creation and Integration of a COSMAdoc Instance

The COSMAframe component COSMAdoc Creator is responsible for the creation of a composition-specific instance of the COSMAdoc template. On the basis of a given generic service orchestration script provided by the service composer, the COSMAdoc Creator uses a composition decomposer to atomize the script into its atomic service types. For each atomic service, e.g. CheckDomain, CreateWebhostingAccount, etc., and for the DSC service an empty SLA element is created and embedded into the SlaSetAssembly of the COSMAdoc instance. Initial parameters are set for all SLA documents, e.g. service names, etc.

Afterwards, the COSMAdoc Integrator integrates the COSMAdoc instance. First, pre-

defined settings such as SLA parameters to be used, identifiers for the service requester and the composite service provider are taken from a database. Second, the SlaSetUsageValidation section of the COSMAdoc instance needs to be integrated. The directives for which elements are restricted in which way are taken from a configuration file. Available predicates defined in COSMA include but are not limited to makeMandatory, makeOptional, excludeElement, setNegotiable, or setMask. Exemplary, the following usage validation restrictions are created for the CheckDomain service:

- The predicate makeMandatory enforces that i.e. an end point reference of an atomic service implementation is specified in the SLA document.

```
<constraint action="makeMandatory(//Sla[@SlaId='2']
/.../ServiceReference/anyRef)"'>
```

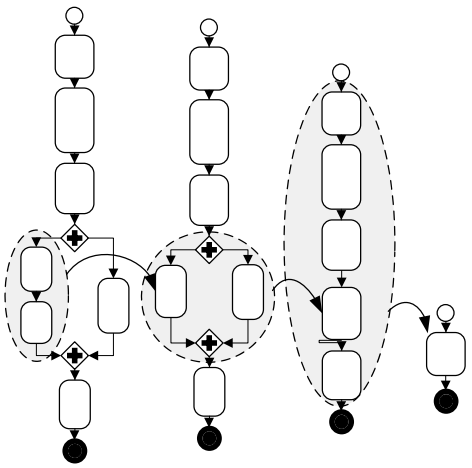
- The predicate excludeElement excludes i.e. the possibility to specify penalties in the SLA with the service requester. The predicate is bound to an equal condition (=). Thus, the predicate is only used if the atomic service provider of the SLA 1 is "MyNic". This restriction may be caused by the fact that MyNic does not accept the specification of penalties.

```
<conditional>
  <context condition="//Sla[@SlaId='2']/.../
    ServiceProvider='MyNic'">
    <constraint action="excludeElement(//
     Sla[@SlaId='1']/.../FinancialTerms/Penalty)"/>
  </context>
</conditional>
```

- The predicate setNegotiable is used i.e. to explicitly state that the reward for invocation of the CheckDomain service is negotiable.

```
<constraint action="setNegotiable(//Sla[@SlaId='2']
/.../ClearingUnitPrice/ValueExpr)">
```

Finally, the SlaSetDataValidation section needs to be integrated. Values for static data validation restrictions are taken from a database similar to usage restrictions above, i.e. availability must always be below hundred percent or a certain service requires a minimal response time retrieved from experiences. Possible predicates include i.e. setValue, setMaxValue, setMinValue, or setValueRange. Before dynamic data validation restrictions can be integrated, the COSMA Integrator analyzes the service orchestration script and used SLA parameters of the SlaSetAssembly. Dynamic data validation restrictions usually result from the structure of the service orchestration script and depend on the actual service parameters. Thus, the COSMA Integrator decomposes the service orchestration script into its atomic composition patterns, i.e. sequence, parallel split, and loop. Afterwards,



Aggregation formulas (extract)

Pattern	CSLA parameter	Aggregation formula
Sequence	Response time	$RT_{CS} = \Sigma(RT_{S\text{-}i})$
	Availability	$A_{CS} = \Pi(A_{S\text{-}i})$
	Throughput	$T_{CS} = \min\{T_{S\text{-}i}\}$
	Encryption Level	$E_{CS} = \min\{E_{S\text{-}i}\}$
Parallel split	Response time	$RT_{CS} = \max(RT_{S\text{-}i})$
	Availability	$A_{CS} = \min(A_{S\text{-}i})$
	Throughput	$T_{CS} = \min\{T_{S\text{-}i}\}$
	Encryption Level	$E_{CS} = \min\{E_{S\text{-}i}\}$
Loop	Response time	$RT_{CS} = k\ RT_S$
	Availability	$A_{CS} = A_S$
	Throughput	$T_{CS} = T_S$
	Encryption Level	$E_{CS} = E_S$

Notation:
CS = composite service
S-i = atomic services I
k = assumed number of repetitions in a loop

Composition-specific aggregation formula for RT_{CS} in the DSC service:

$$RT_{CS} = \Sigma(RT_{checkDomain}, RT_{createWebhostingAccount}, RT_{createDomain}, \max(RT_{setupPhysicalHosting}, \Sigma(RT_{nameServer}, RT_{registerDomain})), RT_{creditCardPayment})$$

Figure 3: Decomposition of the DSC service and creation of composition-specific aggregation formulas

composition-specific aggregation formulas are created from generic aggregation formulas in a reverse order, i.e. as proposed in [JRGM04]. Fig.3 illustrates the decomposition of the DSC service orchestration script, summarizes generic aggregation formulas for different SLA parameters and shows the resulting composition-specific aggregation formula for response time of the DSC service.

The composition-specific aggregation formulas are then embedded into the Aggregation-Formulas section of the COSMAdoc instance, given a unique identifier and referenced by data validation predicates of the SlaSetDataValidation section. For example, the created aggregation formula for composite service response time would be embedded as follows:

```
<AggregationFormulas>
  <GuaranteeTerms>
    <Formula Id="789">
    '//Sla[@SlaId='2']/Terms/GuaranteeTerms/.../
    ServiceLevelObjective/Value'+
    '//Sla[@SlaId='3']/Terms/GuaranteeTerms/.../
    ServiceLevelObjective/Value'+
    '//Sla[@SlaId='4']/Terms/GuaranteeTerms/.../
    ServiceLevelObjective/Value'+ max(
    '//Sla[@SlaId='5']/Terms/GuaranteeTerms/.../
    ServiceLevelObjective/Value',
    '//Sla[@SlaId='6']/Terms/GuaranteeTerms/.../
    ServiceLevelObjective/Value'+
```

```
      '//Sla[@SlaId='7']/Terms/GuaranteeTerms/.../
      ServiceLevelObjective/Value') +
      '//Sla[@SlaId='8']/Terms/GuaranteeTerms/.../
      ServiceLevelObjective/Value'
      </Formula>...
    </GuaranteeTerms>
    <FinancialTerms>...
    <MonitoringTerms>...
</AggregationFormulas>
```

The according data validation restriction that connects the aggregation formula with a SLA parameter specified in GuaranteeTerms of the CSLA would be stored in the SlaSetDataValidation section as follows:

```
<constraint action="setMinValue(//Sla[@SlaId='1']/.../
  ServiceParameter[@Name='ResponseTime']/
  ServiceLevelObjective/Value,//AggregationFormulas/.../
  Formula[@Id='789'])"/>
```

Analogously, aggregation formulas and data validation restrictions are created for all existing SLA parameters. They will be used in the following to express relationships and dependencies between ASLA and CSLA contents.

3.2 Negotiation of a COSMAdoc Instance

The negotiation of SLAs involved in the COSMAdoc instance's SlaSetAssembly is executed in three steps: preparation of the negotiation process, iterative SlaSetAssembly negotiation, and final ASLA/CSLA conclusion. The high-level perspective on the negotiation of the DSC service is illustrated in Fig.4.

For the preparation of the negotiation process, a negotiation engine needs to be configured. For each atomic service involved in the service orchestration script and the composite service itself, a negotiation agent is created that is responsible for negotiating the according SLA (agents 1-8 in Fig.4). In the use case, seven atomic SLAs and the resulting DSC CSLA need to be negotiated; hence, eight negotiation agents are created in the negotiation engine. The configuration of each negotiation agent in terms of negotiation protocols and negotiating parties is made according to the NegotiationTerms individually specified in respective SLAs. The negotiation strategy of each negotiation agent depends on policies and decision rules that were given to a negotiation engine for its agents by the ISP.

After the preparation, an iterative negotiation process is executed by the negotiation engine in combination with the COSMAdoc instance. For this, first, an initial validation of all ASLAs is executed by the COSMA Manager on the validation interface. Since no negotiations were made so far, all usage and data validation restrictions are violated and according errors are set into all SLA documents. Each negotiation agent receives only the

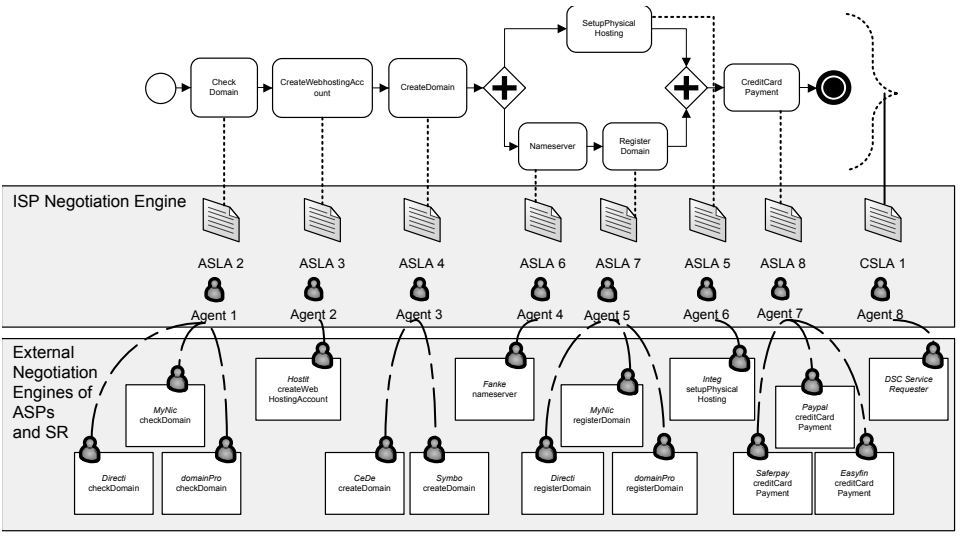


Figure 4: Setting of agent-based SLA negotiations for services involved in the DSC service

SLA document it negotiates. Thus, the agent does not control how its individual negotiation results influence the overall CSLA negotiation. This mapping is controlled by the COSMA Manager using data validation restrictions and aggregation formulas. Negotiation agents must only ensure that all usage and data validation errors in their individual SLA document are eliminated. Example usage and data validation restrictions on an ASLA are as follows:

```
<Sla SlaId="2" ServiceName="CheckDomain"> ...
  <AnyRef usage-error="makeMandatory"/> ...
</Sla>

<Sla SlaId="7" ServiceName="RegisterDomain"> ...
  <ValueExpr dataValidation-error=
   "recommendValues(10.00)"> ...
  <Value dataValidation-error="setMinValue(128)"/> ...
</Sla>
```

The produced error messages are used by negotiation agents for negotiation of ASLAs. Each negotiation agent (agents 1-7) is now able to negotiate with its opponent the contents of the ASLA considering the initially created data and usage validation errors. Based on the negotiation agent's decision making strategy, the agent selects an atomic service implementation for each service and fills the respective ASLA with the required contents.

On this basis, an initial CSLA validation can be executed on the validation interface of the COSMA Manager and usage and data validation errors are set into the DSC service's SLA. Data errors are created using the aggregation formulas specified for the DSC service. One

data validation error corresponding to the data validation restriction presented above is:

```
<Sla SlaId="1" ServiceName="DSCService"> ...
  <ServiceParameterSet>
    <ServiceParameter Name="ResponseTime">
      <ServiceLevelObjective>
        <Value dataValidation-error=
         "setMinValue(80000)"/> ...
      </ServiceLevelObjective>
    </ServiceParameter> ...
  </ServiceParameterSet> ...
</Sla>
```

The value of 80,000 milliseconds is a dynamically calculated value (Formula Id="789") based on the negotiation results of the ASLAs. The negotiation agent (agent 8) can then execute the negotiation of the CSLA using the validation errors as support. Once the CSLA is filled, the validation interface is invoked to validate the CSLA contents and add remaining validation errors. For example, if the service level objective of response time was 60,000 milliseconds, then the dataValidation-error="setMinValue(80000)" would remain in the CSLA and would need to be fixed by the agent.

Depending on the used negotiation protocol, the process of ASLA validation-negotiation and CSLA validation-negotiation is continued in order to optimize conditions of service consumption and provision. For every CSLA validation, the data validation restrictions change dynamically depending on the ASLA negotiation outcomes. Thus, a stepwise optimization can be realized with adaptive restrictions and harmonized SLAs.

If no validation errors remain in ASLAs and the CSLA and if all negotiating parties can accept proposed SLAs, their representing negotiation agents are allowed to conclude their SLA. Signing SLAs is only possible if the validation process does not produce any validation errors. This ensures that all SLAs are harmonized with each other and comply with the goals of the CSP.

A description of SLA negotiations based on the COSMA approach is published in [LF08b].

3.3 Monitoring and Evaluation of a COSMAdoc Instance

Monitoring and evaluation of the DSC service COSMAdoc instance include (1) determining the actual service levels of atomic services, (2) evaluation of measured service levels with regard to the atomic and composite service, (3) determination of corrective actions that support the composite service provision according to the CSLA, and (4) explanation of service level violations and reasoning for future composite service provisions.

In order to determine actual service levels of atomic services, the MonitoringTerms defined for every service and every service parameter in a COSMAdoc instance need to be processed by the COSMA Manager. The MonitoringTerms defined for the service Directi:CheckDomain and the service parameter response time are as follows:

```
<MonitoringTerms>
  <ServiceParameterSet>
    <ServiceParameter Name="ResponseTime">
      <Obligated>Directi</Obligated>
      <AccessPattern>
        <AccessMechanism>Pull</AccessMechanism>
        <DataSource>rdbms.directi.com:4040</DataSource>
        <AssessmentInterval>
          <TimeIntervall>
            <StartTime>2007-10-15T14:00:00.000+01:00
            </StartTime>
            <Duration>2008-01-05T00:00:00.000+01:00
            </Duration>
          </TimeIntervall>
          <Count>02:30:00.000</Count>
        </AssessmentInterval>
      </AccessPattern>
      <ServiceLevelMeasurement>2000
      </ServiceLevelMeasurement>
    </ServiceParameter> ...
  </ServiceParameterSet>
</MonitoringTerms>
```

The example shows that the ASP Directi is obligated to monitor the response time and provide the aggregated monitoring data in the defined database interface which can be pulled by the ISP between the specified dates every 2.5 hours. The last service level measurement for response time which is stored in the ServiceLevelMeasurement element is 2,000 milliseconds.

The monitoring process is executed for all service parameters and all atomic services of the DSC service COSMAdoc instance. The results of these parallel monitoring processes are stored in the service level measurement elements. The evaluation of the measured service levels is executed by the COSMA Validator and Violation Detector and can be invoked on its validateCOSMAdoc interface. First, measured service levels are compared with service level objectives. For example, for the Directi:checkDomain service, the measured response time is 2,000 milliseconds whereas the service level objective and qualifying condition could be “LESSTHAN” and “3000” milliseconds. Hence, the service level objective is fulfilled and not violated. This result is stored in the StateTerms of the according SLA.

After the atomic service level evaluation, service levels of the DSC service must be evaluated. Therefore, first, the anticipated service levels of the composite service must be calculated and compared with the service level objectives of the CSLA. The calculation is based on an aggregation formula that was created during COSMAdoc integration and that is stored in the AggregationFormulas section. The formula for calculating the service level measurement of response time is almost the same as the one presented in Fig.3. The only difference is that the terms of the formula (paths to ASLA elements) are the service level measurements of the ASLAs for response time rather than the service level objectives.

The predicate defined in the data validation section for the service level measurement of response time is setValue (not setMinValue) since the response time of the DSC service results directly from response times of atomic services.

In general, there are three possible situations. First, all ASLAs are fulfilled and no violation of the CSLA occurs. Second, one or more ASLAs are violated but do not cause a violation of the CSLA. Third, one or more ASLAs are violated and cause a violation of the CSLA. It should be noted that managing the evaluation process for more than one service level and for constantly changing service level measurements is a complex task. The elements of COSMA enable a full automation of this evaluation process. The aspect of dealing with CSLA violations in terms of claiming penalties, re-negotiating ASLAs, changing service implementations, or even re-planning the service orchestration are not covered in this paper. For the explanation of service level violations and reasoning for future composite service provisions, the service orchestration script and sources of violation need to be analyzed. Dynamic service profiles are a useful mechanism to store service fulfillment or violation experiences [AKKZ06].

4 Prototypical Implementation

To prove the applicability of the COSMA approach in managing SLAs of composite services, a prototypical implementation was developed. This prototypical implementation serves as a demonstrator for the realization of the key elements of COSMA. The demonstrator is far from being restrictive. It can be adapted and extended depending on a certain usage area or scenario. The use case scenario presented above is implemented in a demonstrator. Although the COSMA approach represents an automated approach towards composite SLA management and does not require human intervention in executing the management steps, the demonstrator provides a graphical user interface for human interaction (see screenshot in Fig.5). This graphical user interface allows a human user to manually trigger individual COSMA management tasks and view the results of every single operation (creation, negotiation, validation, etc.). Additionally, the results of single operations can be changed to allow testing and probing alternatives, i.e. in restricting values and altering aggregation formulas.

5 Conclusions and Outlook

The paper presented a use case for managing SLAs in composite services with the COSMA approach. It aimed at contributing to the overall understanding and enhancement of COSMA. It was shown how relationships and dependencies between SLAs can be maintained at a central point of information. On this basis, a CSP can control and optimize its SLA management activities, pro-actively plan financial consequences, and dynamically calculate the expected service level objectives from dynamically varying service orchestration scripts. By using the SlaSetDataValidation restrictions, a CSP can dynamically calculate

Figure 5: Screenshot of the COSMA demonstrator during COSMAdoc monitoring and evaluation

the expected service level objectives of a composite service and limit the values contracted in all SLAs to meaningful levels. A CSP can restrict the penalty payments that are due in case of service level violations. Aggregation formulas can be used to split up imposed penalty functions (of the composite service) to penalty functions that a CSP imposes to the atomic service provider. These penalty functions can then be used to define minimal penalties claimed from the atomic service provider. On this basis, a CSP can evaluate the impact of service level variations on composite service levels and trigger corrective actions if necessary and possible.

COSMA and the presented use case are to be interpreted as a starting point that must be extended, adapted to individual requirements of further usage scenarios, and tested on them. This may lead to a widespread evaluation and assessment of the approach and will result in a broader range of practical experiences.

Currently, COSMAlife describes mechanisms for the most important SLA lifecycle phases: creation, integration and negotiation of SLAs and monitoring, evaluation and termination of SLAs. For an overall SLA lifecycle management, support of all phases of the lifecycle would be necessary. In the preliminary phase of finding a potential agreement partner, this includes mechanisms to advertise provided services and their SLAs and find suitable service providers based on certain criteria. After the termination of SLAs involved in a managed composite service, the experienced SLA behaviour should be stored in dynamic service profiles which can be used during SLA negotiations and creation of usage and data restrictions. These and other steps which would iteratively improve the behavior of a COSMAframe implementation and connect multiple COSMAlife cycles with each other need to be added and integrated in the proposed approach.

Another important aspect regards the more detailed investigation on penalties and bonuses in the management of SLAs. Instead of aggregating quality of service parameters and the financial parameter reward, penalties would require a decomposition of a received (multi-dimensional) penalty from a service requester and the allocation of sub-penalties to atomic SLAs. With different penalties on service parameters and complex service composition patterns, this task is an extremely complicated task which is however vital for a broad adoption of the COSMA approach by composite service providers.

References

[ACD+] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu. Web Service Agreement Specification (WS-Agreement). `http://www.gridforum.org/documents/GFD.107.pdf`.

[ACKM04] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web services: concepts, architectures and applications*. Springer, Berlin, New York, 2004.

[AKKZ06] W. Abramowicz, M. Kaczmarek, M. Kowalkiewicz, and D. Zyskowski. *Architecture for Service Profiling. In: Modeling, Design, and Analysis for Service-oriented Architecture Workshop (MDA4SOA 2006), pp. 121–130*. Chicago, 2006.

[ASG] ASG. Integrated Project Adaptive Services Grid (ASG). `http://www.asg-platform.org`.

[JRGM04] M.C. Jaeger, G. Rojec-Goldmann, and G. Muehl. *QoS aggregation for Web service composition using workflow patterns. In: 8th International IEEE Enterprise Distributed Object Computing Conference (EDOC 2004), pp. 149-159*. Monterey, 2004.

[KL03] A. Keller and H. Ludwig. *The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services, pp. 57-81*. J. of Network and Systems Management 11, 2003.

[LBKF05] A. Ludwig, P. Braun, R. Kowalczyk, and B. Franczyk. *A Framework for Automated Negotiation of Service Level Agreements in Service Grids. In: Bussler, C., Haller, A. (eds.) Business Process Management Workshops. LNCS, vol. 3812, pp. 89–101*. Springer, Berlin, Heidelberg, 2005.

[LF08a] A. Ludwig and B. Franczyk. *COSMA - An Approach for Managing SLAs in Composite Services. In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) 6th International Conference on Service Oriented Computing (ICSOC 2008). LNCS, vol. 5364, pp. 626-632*. Springer, Berlin, Heidelberg, 2008.

[LF08b] A. Ludwig and B. Franczyk. *Managing Dynamics of Composite Service Level Agreements with COSMA. In: 5th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2008)*. Jinan, 2008.

[MGL+07] M. Momotko, M. Gajewski, A. Ludwig, R. Kowalczyk, M. Kowalkiewicz, and J.Y. Zhang. *Towards adaptive management of QoS-aware service compositions. J. of Multiagent and Grid Systems 3, pp. 299-312*. J. of Multiagent and Grid Systems 3, 2007.

[Pap07] M.P. Papazoglou. *Web Services: Principles and Technology*. Prentice Hall, Essex, 2007.

[Rug07] R. Ruggaber. *Internet of Services SAP Research Vision. In: 16th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2007), pp. 3*. Rome, 2007.

[TPP02] V. Tosic, K. Patel, and B. Pagurek. *WSOL - Web Service Offerings Language. In: Web Services, E-Business, and the Semantic Web. LNCS. pp. 57-67*, volume 2512. Springer, Berlin, Heidelberg, 2002.