SALMonADA: A Platform for Monitoring and Explaining Violations of WS-Agreement-Compliant Documents

C. Müller(1), M. Oriol(2), M. Rodríguez(2), X. Franch(2), J. Marco(2), M. Resinas(1), A. Ruiz–Cortés(1)
(1) *University of Seville, LSI, Seville (Spain), ISA research group*
{cmuller,resinas,aruiz}@us.es
(2) *Universitat Politècnica de Catalunya, Barcelona (Spain), GESSI research group*
{moriol,jmarco}@lsi.upc.edu, {marcr,franch}@essi.upc.edu

***Abstract*—Quality assurance techniques have been developed to supervise the service quality (QoS) agreed between service-based systems (SBSs) consumers and providers. Such QoS is usually included in service level agreements (SLAs) and thus, SLA monitoring platforms have been developed supporting violation detection. However, just a few of them provide explanation of the violations caused by observed QoS at monitoring time, but not in an user-friendly format. Therefore, we propose a general monitoring and analysis conceptual reference model and we instantiated it with SALMonADA, a SBS that notifies the clients with violations and their causes in their own easy-to-understand specification terms. In addition, our platform performs an early analysis notification that avoids delays in the client notification time when a violation takes place. Moreover, we have implemented a web application as a SALMonADA client, to prove how it monitors, analyses and reports to their clients the service level fulfillment of real services subject to a SLA specified with WS–Agreement.**

***Keywords*-monitoring; analysis; violation detection; violation explanation;**

I. Introduction and Motivation

Service level agreements (SLAs) establish the service quality (QoS) agreed between service-based systems (SBSs) consumers and providers, and thus, quality assurance techniques are needed to supervise the SLAs fulfillment. These techniques require monitoring platforms enhanced with analysis capabilities to reason about the monitored information in order to extract useful information for the parties.

Many research efforts have been made trying to obtain useful monitoring information, starting from general monitoring framework [1], [2]. Thus, several proposals can be found providing a different kind of information from monitored SLAs, such as: *violation detection* in the SOA testing context [3], [4]; *asynchronous violation detection reports* to subscribed clients that wait for the monitoring information instead of requesting it [5], [6]; *event-based violation explanation in SBSs* [7], [8]; and *dynamic SBS adaptation* when a SLA violation is detected.

In this paper we propose a general monitoring and analysis conceptual reference model in which several agents extract useful information from SLAs at monitoring. For that purpose, three kinds of documents are handled by agents, namely: SLAs, monitoring management documents (MMDs) to configure and manage the monitors, and service level fulfillment (SLF) to report the violations and their causes. In addition, we instantiate the conceptual model with SALMonADA, a service-based system (SBS) that integrates upgraded versions of previously developed SLA monitoring (SALMon [9]) and analysis (ADA [10]) proposals[1] . SALMonADA provides the following contributions to such techniques with the aim of extracting useful information at SLAs monitoring: first, it notifies the client with violations and their causes in their own easy-to-understand specification terms; second, it supports expressive and easy-to-understand SLAs specified with WS–Agreement [11]; and finally it performs an early analysis notification that supports the SLA fulfillment analysis when a violation has just been observed, reducing the client notification time.

Proposals like [12] and [13] assuming the availability of a monitoring and analysis engine, benefit from using SALMonADA since they are provided with such a service level fulfillment information needed to adapt the SBS, renegotiate the SLA, achieve reputation statistics, etc.

The paper is organised as follows. Related work is revised in Section II. WS–Agreement specification is introduced in Section III including an example of our supported SLAs. The conceptual reference model is detailed in IV, while its SALMonADA instantiation is included in V. Section VI and VII detail the monitoring and analysis SALMonADA components, respectively. Section VIII reports an evaluation of our proposal. And Section IX concludes the paper with a discussion of contributions.

II. Related Work

On the one hand, many monitoring frameworks have been proposed, [1], [2], [7] considering monitoring and even analysis agents, but to the best of our knowledge, none of them propose a separation of concerns between SLAs, the monitoring information, and the monitoring analysis result; as we do for SLAs, MMDs, and SLFs.

[1]Both proposals have been widely revised to support the novelties of the proposed conceptual reference model (e.g. the management of SLAs and SLF have been included in ADA; and the MMDs management in SALMon).

Table I
RESPONSETIME<100 IN GETRATE OPERATION USING EC [7]

EC Formula lines	Truth Value
forall t1 : time exists t2 : time	-
Happens(ic:getRate(ID,country2,country1), t1,R(t1,t1))^	True
Happens(ir:getRate(ID),t2,R(t1,t2))	True
oc:self:sub(t2,t1)<100	False

On the other hand, several techniques to extract useful information at SLA monitoring have been developed. Thus, we can find proposals providing *violation detection* for WS–Agreement documents in the SOA testing context [3], [4]. However, such testing proposals monitor the service to detect violations at testing and not while the service is consumed. Other proposals such as [5], [6], dealing the latter with non–WS–Agreement documents, provide *asynchronous violation detection reports* to subscribed clients that wait for the SLA monitoring information instead of requesting it, as commonly performed by other proposals. Moreover, there are proposals such as [14], [15], [16], [17] that go further and when they detect a SLA violation they *dynamically adapt* the SBSs following different strategies, but this dynamic reaction is out of the scope of the paper.

As far as we know, there is only a set of proposals from Mahbub and Spanoudakis that provides *violation explanation* in SBSs [7], [8]. Such proposals use event calculus (EC) and they report an event-based explanation of the SBSs violation as follows: "The `operation` event has violated the EC formula `F` of the term `T`". For instance, Table I depicts a four-lines EC formula that is reported as explanation to a client informing about the response time violation of the `getRate` operation. Such a formula is included inside WS–Agreement document terms[2].

III. WS–AGREEMENT IN A NUTSHELL

The WS–Agreement recommendation [11] describes both an XML–based language and a protocol that facilitates the publication, discovery, and monitoring of SLAs between two parties, usually a service provider and a service consumer. The SLAs are created after a negotiation process and they comprise an *agreement identifier*, an *agreement context* containing information about the involved parties, and *agreement terms* that describe both the characteristics of the services to be provided in *service terms* and the guarantees on such services in *guarantee terms*. Note that WS–Agreement only defines the general structure of a SLA and the kind of terms it may include. However, it does not specify any vocabulary to express the features of the service.

Service terms are divided into two elements: first, *service description terms* that define the features of the service that will be delivered under an agreement; and second,

[2]This sample is included in [7] at page 26.

```
<Agreement AgreementId="1.0"...>
 <Name>SALMonADA-compliant ADA SLA</Name>
 <Context>
  <Initiator>IneedSLAAnalysisCorp.</AgreementInitiator>
  <Responder>ADA Tool (ISA Group)</AgreementResponder>
  <ServiceProvider>AgreementResponder</ServiceProvider>
  <ExpirationTime>2013-01-01T00:00:00</ExpirationTime>
 </Context>
 <Terms Name="ADAService">...
  <ServiceProperties Name="SALMon-compliant metrics"...>
   <Variable Name="AverageResponseTime"
             Metric="metrics/Float">...</Variable>
   <Variable Name="GeneralResponseTime"
             Metric="metrics/Float">...</Variable>
   <Variable Name="AverageAvailability"
             Metric="metrics/Percentage">...</Variable>
  </ServiceProperties>
  <ServiceDescriptionTerm Name="ADA-SDT"
                          ServiceName="ADAService">
   <WebServiceInformation Name="ADAService-WSDL">
    <description>ADA is a SLA analyser</description>
    <wsdl>http://www.isa.us.es:8081/ADAService?wsdl</..>
    <endp>http://www.isa.us.es:8081/ADAService?wsdl</..>
    <operation opName="checkDocumentConsistency">...
    ... more operations are included ...
   </WebServiceInformation>
  </ServiceDescriptionTerm>
  <GuaranteeTerm Name="GeneralAvailability"...>
   <SLO> AverageAvailability >= 95 </SLO>
  </GuaranteeTerm>
  <GuaranteeTerm Name="generalResponseTimeRelations"...>
   <ServiceScope ServiceName="ADAService">
     checkDocumentConsistency, xmlToWSAg4People,
     wsag4PeopleToXML, getMetricFile
   </ServiceScope>
   <SLO> AverageResponseTime<=GeneralResponseTime </SLO>
  </GuaranteeTerm> ... more guarantees are included ...
 </Terms>
</Agreement>
```

Figure 1. Main elements of SALMonADA-compliant ADA SLA

service properties that define named, service–related sets of *variables* that can be used for the specification of guarantee terms and must be therefore considered for agreement monitoring. All variables must include a domain–specific metric definition to specify the semantics and type of a variable. How the service description terms are organized and expressed is left open by the recommendation. Thus, as depicted in the document of Fig. 1, we use our own domain specific language (DSL) for defining the ADA analysis service, including three properties: the average between several response time measures of the same operation (`AverageResponseTime`); the average response time of any operation service (`GeneralResponseTime`); and the average service availability (`AverageAvailability`).

Guarantee terms describe the *service level objectives* (SLOs) that an *obligated party*, usually the service provider, must fulfill as part of the SLA. The SLO is an assertion defined over monitorable variables defined in the service properties section of the agreement document, and over external factors such as date, time, etc. The SLOs can be expressed using any suitable assertion language. In our example, the provider assures a minimum availability for the service, and a relation assuring that the average response time of 4 particular service operations is less or equal to

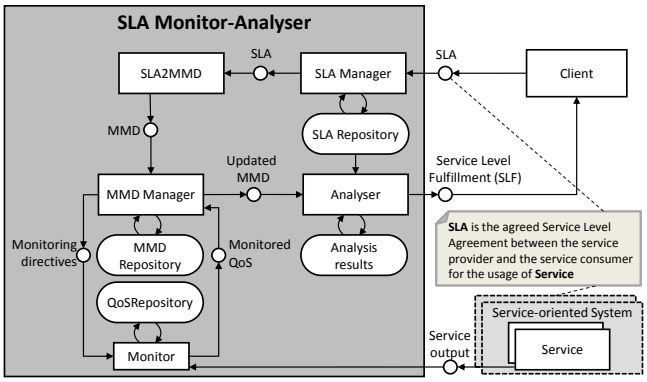

Figure 2. Conceptual Reference Model

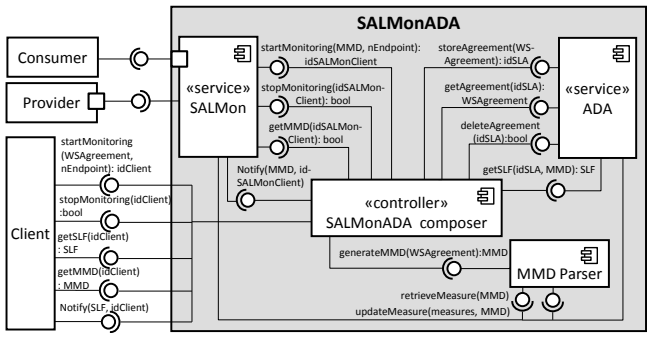

Figure 3. Architectonic Model of SALMonADA

the average response time of any service operation (cf. Section VII for more details about the assertion language).

IV. Conceptual Reference Model

In this section we propose a conceptual architecture as a conceptual reference model. This architecture can be instantiated by several design architectures for monitoring SLAs, ensuring a decoupled structure between the monitoring of the services and the analysis of the SLA compliance. We illustrate the architecture using the SAP-TAM Notation [18].

We briefly describe here the different agents of the system, as depicted in Fig. 2, with their required responsibilities.

Client: is the user of the platform. The responsibilities of the client is to provide the SLA to monitor, and its goal is to retrieve the results of the monitored SLA, structured in a document named Service Level Fulfillment (SLF). It is important not to assimilate the SALMonADA client with the consumer of the service (the SALMonADA client could be either the consumer of the service, the provider or a third party interested in monitoring the assessment of the SLA).

SLA Manager: is an agent responsible for retrieving and managing the monitored SLAs (storage, provisioning, deletion...). The SLAs are stored in a SLA Repository.

SLA2MMD: is an agent that decouples the SLA Manager from the MMD Manager. It works as a bridge between the SLA (a contractual specification understood by SLA-dependent agents) with the MMD (a specification of the monitoring directives to configure a monitor).

MMD Manager: Similarly to the SLA Manager, the MMD Manager is responsible of managing the MMD documents. It provides the Monitoring directives to the Monitor, and it is used to provide an updated MMD with the monitored values. The MMDs are stored in a repository.

Monitor: is the agent responsible of monitoring the services QoS. The observed QoS are stored in a repository.

Analyser: is the agent used to check if the monitored QoS of a service (obtained through the MMD with values) is compliant with the agreed QoS included in the SLA.

The proposed architecture of the conceptual model provides a reference to instantiate the different agents with concrete solutions, establishing a clear separation of concerns on the management of the SLAs, the MMDs and the SLFs.

V. The SALMonADA Platform

In this section we provide a design architecture as an instantiation of the previous conceptual model. We propose SALMonADA, a platform able to monitor SLAs specified in WS–Agreement. The proposed solution combines two existing frameworks that have been extended to realize this project: SALMon[9] and ADA[10]. We also describe in this section the two approaches that SALMonADA has to obtain the results of the analysis.

A. Design Level Architecture

As shown in Fig. 3, we have developed SALMonADA as a SBS with the following elements:

Client: provides the SLA to monitor expressed in WS–Agreement. It is able to retrieve either the MMD or the SLF, and if desired, it can also receive notifications when the SLA has been violated (see subsection V-B).

SALMonADA composer: is the service that composes the internal services of the platform. It provides the interface to the client and manages the execution process of the system. It also adds an independence layer on the interaction required between the analysis of the SLAs (performed by ADA) and the monitoring of the services QoS (performed by SALMon). Such a decoupled structure allows to add or modify the internal components in a very flexible manner. (i.e. allows to replace the monitor or the analyzer without affecting the other elements of the platform).

SALMon: is the service responsible for monitoring the services QoS. It acts as both the MMD Manager and Monitor agents of the conceptual model. A detailed description of the SALMon behavior is included in section VI.

ADA: is the service responsible of managing and analyze the different WS–Agreement documents. It supports the analysis of WS–Agreements with expressive assertions inside guarantee terms. It acts as both the SLA Manager and Analyzer of the conceptual model. A detailed description of the ADA behavior is included in section VII.

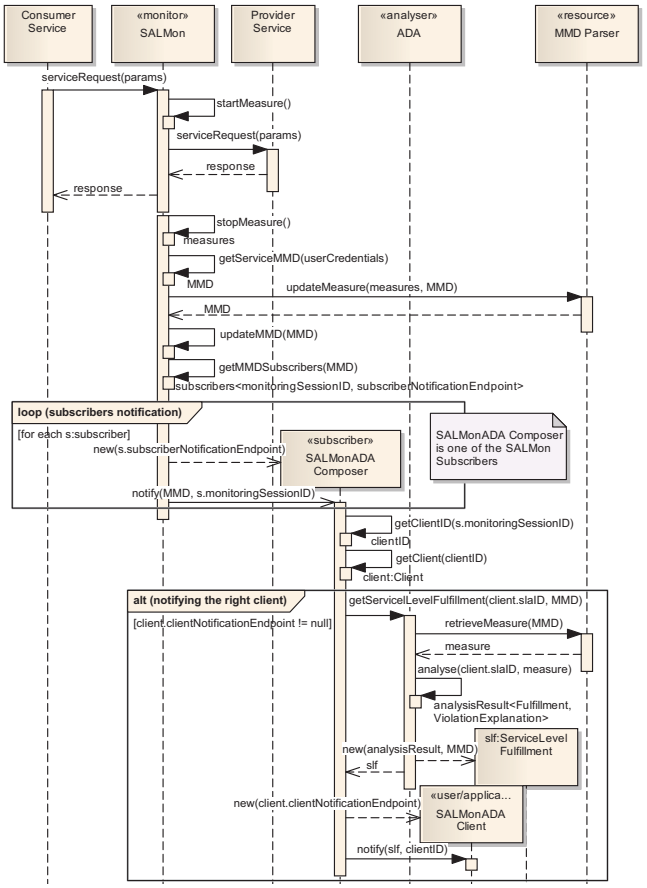

Figure 4. Asynchronous SALMonADA approach.

MMD Parser: is the service that implements the SLA2MMD agent extracting the monitoring information, and it also implements the functionality to interact with the MMDs (retrieve or update values). Thus, the MMD structure, whose information is used by all platform components, is decoupled from both ADA and SALMon. Therefore, different MMD structures can be developed, if needed.

B. The Asynchronous and Synchronous Approaches

SALMonADA platform is designed and developed to support asynchronous and synchronous interaction styles with their clients. Thus, a client, based on its own benefit, may choose its preferred approach. Independently of the selected approach a client must start and stop the SALMonADA monitoring to be subscribed/unsubscribed as client. The start process requires a WS–Agreement document to monitor its fulfillment and such a process slightly varies for clients using asynchronous approach because they must also provide where the notification is awaited (notification endpoint).

Asynchronous Approach: it is the most convenient way to interact with the platform due to the asynchronous nature of SALMonADA service monitoring and analysing. In this sense, the platform incorporates an *early analysis notification* that support the SLA fulfillment analysis as soon as a violation is observed. Thus, the SLF notification is sent to the client without more delay than the analysis time. Therefore, SALMonADA notifies their clients only when the monitored service has just been used by a service consumer and a SLA violation is incurred. As depicted in sequence diagram of Fig. 4, once the client has started to monitor, the *provider service* included in the reported WS–Agreement document is monitored by the *SALMon* component. Next, the MMD created from the monitored WS–Agreement document is sent to the *MMD Parser* with monitored measures to be updated. Finally, the new MMD is notified to the *SALMonADA composer* that sends it to the *ADA* component to analyse the service level fulfillment of the corresponding WS–Agreement document (cf. Section VII for more details). Then, the client is notified about such SLF informing about: the WS–Agreement document fulfillment or not; and in the latter case, both: the specific violated WS–Agreement terms and the violating metrics, are included as violation explanation. Section VIII includes an example of how this SLF is reported to users. Note that SALMonADA supports the same endpoint acting as different clients, for instance, one of them to get the SLF, other to store reputation analytics of the service consumer and provider, or even to perform self-adaptation strategies.

Synchronous Approach: it allows the client to control when SALMonADA operations are requested to get: the current MMD with the most recent monitoring information obtained by *SALMon*; or the current SLF of the WS–Agreement document analysed by *ADA*. However, the availability of new monitoring information is not assured by the platform due to the aforementioned asynchronous nature of its monitoring and analysis. Thus, it is possible for the client to get the same monitoring information in consecutive MMD requests. Only if the SALMonADA client is acting as consumer or provider service, the availability of new monitoring information is known.

VI. SALMon Component in a Nutshell

SALMon is a framework aimed at monitoring the QoS of services [9]. It has been developed as a SBS itself, providing hence an easy integration on frameworks developed as a service-oriented system, such as Self-Adaptive SBS [19] or Cloud monitoring [20] frameworks. In this work, SALMon has been enhanced with the *MMD Manager Service*, which can be invoked through standard SOAP-based web service protocols. The *MMD Manager Service*, in turn, invokes the already existing *Monitor Service* to configure the monitor. Fig. 5 depicts the extended SALMon components.

MMD Manager: is the service that stores the MMDs in the repository and configures the Monitor accordingly. To do so, it parses the MMD using the MMD Parser component.

Monitor Service: The *Monitor Service* is responsible for retrieving the QoS of the services. To do so, it creates the required *Measure Instruments* to obtain the *QoS Data*.

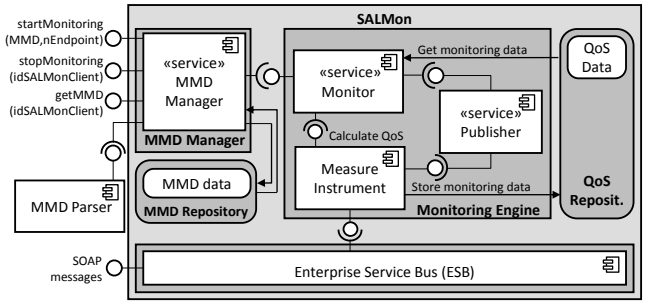

Figure 5. Technical Architecture of the extended SALMon

Monitoring is performed by means of an *Enterprise Service Bus (ESB)* (i.e., instead of invoking the services directly, all requests and responses are sent through the ESB) which in turn feeds the *Measure Instruments*.

Measure Instrument: is the component that obtains the values of a basic quality metric, whereas derived metrics are calculated by computing the required formula (e.g. average). The *Measure Instruments* are activated depending on the quality metrics to measure. Once activated, they receive the SOAP messages through the ESB that intercepts them.

Publisher Service: implements the Observer pattern for services in a SBS. It notifies any relevant state change to any subscribed service. This pattern requires that the subscribed services (the observers) implement the required interface to receive such a notification. This is achieved by defining a common WSDL-interface with the notify method.

VII. ADA Component in a Nutshell

ADA is an *Agreement Document Analysis* framework aimed at extracting useful information from agreement documents at any SLA life-cycle stage [10]. It has been developed based on our previous theoretical works on applying the *constraint satisfaction problem* (CSP) [21] paradigm to the automated procurement of web services [22] and explanation of WS–Agreement document inconsistency and non-compliance situations [23], [24]. The main ADA features are: (1) *interoperability* through a triple distribution model, namely: as a Java library, as an OSGi[3] service and as a web service; and (2) *solver independent* through the use of a semantic mapping between WS–Agreement documents and CSP paradigm, that protects our design from the possible variations derived from using different solvers. In this work, ADA has been enhanced with the ADA Manager and several analysis facilities depicted in Fig. 6 and detailed as follows.

ADA Manager: is responsible for SLA storage and retrieval from the repository; as well as the translation between several SLA models to a WS–Agreement-based normalised one that ADA is able to analyse.

[3]www.osgi.org

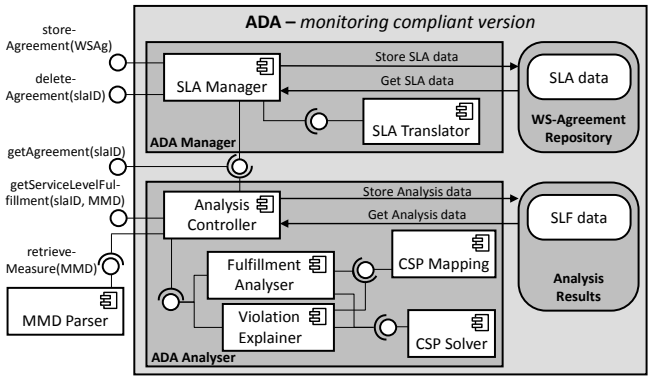

Figure 6. Technical Architecture of the monitoring-compliant ADA.

ADA Analyser: is responsible for the fulfillment analysis between the WS–Agreement document and the MMD with monitored measures, as well as the creation of violation explanation when an unfulfillment is detected. This component is also in charge of the SLF storage and retrieval. Note that the use of the CSP paradigm allows ADA to supports the following easy-to-understand for non-technical users expressive assertion language inside WS–Agreement documents (cf. SLOs of Fig. 1):

$P ::= P\ op_L\ P \mid T$ – predicate, $op_L \in \{\wedge \mid \vee \mid \neg \mid \Rightarrow \mid \Leftrightarrow\}$
$T ::= E\ op_C\ E$ – term, $op_C \in \{ = \mid \neq \mid > \mid \geq \mid < \mid \leq \}$
$E ::= ID\ op_A\ ID \mid ID \mid lit$ – expression, op_A is an algebraic operator
– defined on the domain of the service
– properties, variables, and literals

VIII. Explaining Violations with SALMonADA

For demonstration purposes, we have implemented a web application[4] as a SALMonADA client in order to specify or upload the WS–Agreement documents to monitor, execute SALMonADA and receive the results. In this web application, we have introduced the WS–Agreements of ADA and SALMon themselves. By monitoring the SLAs of these services, we assess on the one hand, the functionality of SALMonADA, and on the other, the non-functional aspects of its main components. Moreover, as part of the demonstration, we have simulated the consumers that execute ADA and SALMon services. We describe here the asynchronous SALMonADA approach through monitoring the ADA service and analysing the service level fulfillment with the WS–Agreement document in order to report violation explanations.

To monitor the WS–Agreement, the SALMonADA client invokes the startMonitoring method specifying the ADA WS–Agreement and the endpoint for the notification. In the demonstrated scenario, it is the same web application, but any other client can be subscribed to the notification

[4]SALMonADA web application can be tried at www.isa.us.es/ada.source/SLAnalyzer/ and a screencast is available at gessi.lsi.upc.edu/salmon/ada/

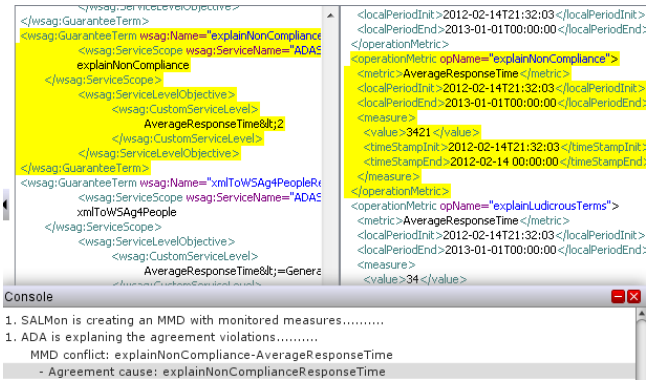

Figure 7. Reporting a violation with SALMonADA

and explaining of the violations of the same ADA WS–Agreement, such as a service reputation agent, service adaptation frameworks, etc. Using this asynchronous approach, as soon as a violation is detected, it is automatically analysed, and then reported to all the subscribed agents. As Fig. 7 depicts, the web application highlights as violation explanation that the `AverageResponseTime` of `explainNonCompliance` operation is the violating metric because it was measured as `3.421` seconds, while the guarantee term obligates the provider to respond in less than `2` seconds.

The WS–Agreement of ADA, already shown in Fig. 1, includes more guarantee terms to monitor, some of them including SLOs involving more than just one quality metric. Hence, an appropriate explanation identifying not only the guarantee term that are involved in the violation of the SLA, but also the concrete violating metrics, is required. For instance, some operations have a higher priority and are required to be faster than the average response time of the different methods of the service (`AverageResponseTime <= GeneralResponseTime`). In this case, our explanation would identify if the violating metric is either `AverageResponseTime` or `GeneralResponseTime` because a simple identification of the violated term is not enough to grasp the violation cause. Similarly, SALMonADA supports the explanation of violations of more expressive SLOs as follows. The provider may guarantee a different average response time limit for the slower service operations, depending on the general response time of the service: ((`GeneralResponseTime >= 0 AND GeneralResponseTime < 2`) `IMPLIES` (`AverageResponseTime < 3`)) `AND` ((`GeneralResponseTime >= 2 AND GeneralResponseTime <= 4`) `IMPLIES` (`AverageResponseTime < 5`)).

IX. Conclusions and Discussion

In this paper we present a conceptual reference model of a monitoring and analysis framework, and SALMonADA as one of it possible instantiations.

The proposed conceptual reference model provides a reference architecture to develop a platform for monitoring SLAs. The proposed architecture ensures and provides the following set of features to implement a concrete platform:

- A flexible and highly decoupled architecture is presented. The different agents of the system deal exactly with the information required in separate documents: SLA, MMD and SLF.
- We propose the MMD as a unique document to manage the monitors, in order to (1) specify the monitoring directives to retrieve the different metrics and (2) report the measured results over the specified metrics.
- We introduce the SLF as the document that explains clearly the violations of the SLA. Other approaches [7], [8] support an event-based violations explanations based on Event Calculus. However, it has in our consideration, the following drawbacks: (1) the client (end-user) must be an expert in EC to specify the guarantee terms with EC formulas, but also to understand the violation explanations; and (2) it is difficult for the client (application or end-user) to grasp the violation origin specially when more than one metric are related in the violating event and/or the violated EC formula.

We present as an instantiation of conceptual reference model, the SALMonADA platform. As Section V-B describes, the platform extracts the monitoring information from the client WS–Agreement document as other authors propose [5], [1]. Our proposal differs from these works since we store it in an independent MMD that will be updated with the monitored information when the service is used at service provisioning time. Such updated MMD is used to analyse the service level fulfillment in order to report to the clients an easy-to-understand explanation including the violated SLA terms and its violating metrics. The features that provides our platform can be summarized as follows:

- SBS: It is a SBS by itself, and because of its decoupled structure, the inherent services can be replaced by others if they just implement the required interface.
- WS–Agreement compliant: it is able to analyze expressive SLOs in WS–Agreement documents. Moreover, the clients do not need to be experts in any reasoning paradigm: neither EC nor CSP, because we support an assertion language (detailed in Section VII) inside WS–Agreement documents that is easier to understand.
- Early notification: We provide an early notification mechanism based on the observer pattern useful for self-adaptive SBS, and other interested parties such as service reputation agents.

However, the platform presented in [7], [8] has two key points that differs from our proposal and makes it very appealing: (1) they consider violations of an expected behavioral of the SBS to monitor by adding assumptions inside the WS–Agreement document; and (2) as they monitor events with EC, its proposal is able to deal with metrics depending on a time interval. Both are part of the possible improvement of our work, the first in terms of expected operations execution flow (it is possible by defining a precedence order for service operations inside the SLA); and the latter using the temporal analysis of ADA that is currently not considered in the proposal.

ACKNOWLEDGMENT

This work has been partially supported by: S–Cube, the European Network of Excellence in Software Services and Systems; the European Commission (FEDER); the Spanish Government under the CICYT projects SETI (TIN2009–07366) and ProS–Req (TIN2010–19130–C02–01); and by the Andalusian Government under the projects THEOS (TIC–5906) and ISABEL (P07–TIC–2533).

REFERENCES

[1] A. D. et al., "Web services on demand: WSLA-driven automated management," *IBM Systems Journal*, vol. 43, no. 1, pp. 136–158, 2004.

[2] A. Keller and H. Ludwig, "The WSLA Framework : Specifying and Monitoring Service Level Agreements for Web Services," *Network*, vol. 11, no. 1, pp. 57–81, 2003.

[3] M. Palacios, J. Garcia-Fanjul, J. Tuya, and C. De La Riva, "A Proactive Approach to Test Service Level Agreements," *Fifth International Conference on Software Engineering Advances*, pp. 453–458, 2010.

[4] M. Di Penta, G. Canfora, G. Esposito, V. Mazza, and M. Bruno, "Search-based testing of service level agreements," in *Proc. of the 9th annual conference on Genetic and evolutionary computation*, pp. 1090–1097, ACM, 2007.

[5] M. Comuzzi and C. Kotsokalis, "Establishing and monitoring SLAs in complex service based systems," *Web Services*, 2009.

[6] A. Michlmayr, F. Rosenberg, P. Leitner, and S. Dustdar, "Comprehensive qos monitoring of web services and event-based sla violation detection," in *Proc. of the 4th International Workshop on Middleware for Service Oriented Computing*, MWSOC '09, pp. 1–6, ACM, 2009.

[7] K. Mahbub and G. Spanoudakis, "Monitoring *ws-agreement* s: An event calculus-based approach," in *Test and Analysis of Web Services*, pp. 265–306, Springer, 2007.

[8] 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.

[9] M. Oriol, X. Franch, J. Marco, and D. Ameller, "Monitoring adaptable soa-systems using salmon," in *Workshop on Service Monitoring, Adaptation and Beyond (Mona+)*, pp. 19–28, 2008.

[10] C. Müller, M. Resinas, and A. Ruiz-Cortés, "A Framework to Analyse WS–Agreement Documents," in *Proc. of The 4th Workshop on Non-Functional Properties and SLA Management in Service-Oriented Computing (NFPSLAM-SOC'10)*, 2010.

[11] A. Andrieux et al., "Web Services Agreement Specification (WS-Agreement) (v. gfd-r.192)," 2011. OGF - Grid Resource Allocation Agreement Protocol WG.

[12] M. Papazoglou, V. Andrikopoulos, and S. Benbernou, "Managing evolving services," *Software, IEEE*, vol. 28, no. 3, pp. 49 –55, 2011.

[13] O. F. Rana, M. Warnier, T. B. Quillinan, F. Brazier, and D. Cojocarasu, "Managing Violations in Service Level Agreements," in *Grid Middleware and Services Chapter Title - Managing Violations in Service Level Agreements*, pp. 349–358, 2008.

[14] M. Comuzzi and G. Spanoudakis, "Dynamic set-up of monitoring infrastructures for service based systems," in *SAC*, pp. 2414–2421, ACM, 2010.

[15] B. Pernici, "Adaptation of web services based on QoS satisfaction," *Service-Oriented Computing*, pp. 65–75, 2011.

[16] R. Kazhamiakin, B. Wetzstein, D. Karastoyanova, M. Pistore, and F. Leymann, "Adaptation of service-based applications based on process quality factor analysis," in *ICSOC/ServiceWave Workshops*, vol. 6275 of *LNCS*, pp. 395–404, 2009.

[17] O. Moser and F. Rosenberg, "Non-intrusive monitoring and service adaptation for WS-BPEL," in *Proc. of the 17th international conference in WWW*, pp. 815–824, 2008.

[18] S. AG, *Standardized Technical Architecture Modeling: Conceptual and Design Level. Version 1.0.* SAP, 2007.

[19] O. Sammodi, A. Metzger, X. Franch, M. Oriol, J. Marco, and K. Pohl, "Usage-based online testing for proactive adaptation of service-based applications," in *COMPSAC*, pp. 582–587, IEEE Computer Society, 2011.

[20] A. Kertesz, G. Kecskemeti, M. Oriol, A. C. Marosi, X. Franch, and J. Marco, "Integrated monitoring approach for seamless service provisioning in federated clouds," in *PDP*, 2012.

[21] E. Tsang, *Foundations of Constraint Satisfaction*. Academic Press, 1995.

[22] A. Ruiz-Cortés, O. Martín-Díaz, A. Durán, and M. Toro, "Improving the Automatic Procurement of Web Services using Constraint Programming," *Int. Journal on Cooperative Information Systems*, vol. 14, no. 4, 2005.

[23] C. Müller, A. Ruiz-Cortés, and M. Resinas, "An Initial Approach to Explaining SLA Inconsistencies," in *Proc. of the 6th Int. Conf. on Service-Oriented Computing (ICSOC)*, pp. 394–406, 2008.

[24] C. Müller, M. Resinas, and A. Ruiz-Cortés, "Explaining the Non-Compliance between Templates and Agreement Offers in WS-Agreement*," in *Proc. of the 7th International Conference on Service-Oriented Computing (ICSOC)*, pp. 237–252, 2009.