Monitoring the QoS of Web Services using SLAs

Chrysostomos Zeginis[1], Dimitris Plexousakis[1]

[1]Institute of Computer Science, FORTH-ICS
P.O. Box 1385, GR 71110, Heraklion, Crete, Greece
{zegchris, dp}@ics.forth.gr

Technical Report 404, ICS-FORTH, May 2010

Abstract

Web services are an emerging technology which has been attracting a lot of attention from academia and industry during the recent years. Once Web services become operational, their execution needs to be managed and monitored so that a clear view of how services perform within their operational environment can be gained, informed management decisions can be made and control actions for modifying and adjusting the services' behavior can be performed. Many approaches have been proposed in the literature and have proven that monitoring is crucial for successful Web service deployment.

This technical report analyzes the methods that can be used to monitor the execution of a Web service, in order to comply with the corresponding Service Level Agreement (SLAs). Nowadays, most of Web service providers sign SLA contracts with their clients in order to guarantee the offered functionality of their services. A monitoring system is introduced to check violations in the pre-agreed metric values of SLAs. These results can be very helpful for service providers, who can then take corrective actions to improve their services.

1 Introduction

The monitoring phase is one of the most important parts of the Web service lifecycle, as it enables gaining a clear view of how services perform within their operational environments. It primarily concerns itself with service-level measurement; monitoring is the continuous and closed loop procedure of measuring, monitoring, reporting, and improving the QoS of systems and applications delivered by service-oriented solutions. Service monitoring involves several distinct activities including logging and analysis of service execution details, obtaining business metrics by analyzing service execution data, detecting business situations that require management attention, determining appropriate control actions to take in response to

changing business situations, be it mitigating a risk or taking advantage of an opportunity, and using historical service performance data for continuous service improvement.

The service monitoring phase targets the continuous evaluation of service-level objectives, monitoring the services layer for availability and performance, managing security policies, tracking interconnectivity of loosely coupled components, analyzing the root cause and correcting problems. To achieve this objective, service monitoring requires that a set of QoS metrics be gathered on the basis of SLAs, given that an SLA is an understanding of service expectations. In addition, workloads need to be monitored by the service provider to ensure that the promised performance level is being delivered, and to take appropriate actions to rectify non-compliance with an SLA, such as reprioritizing and reallocating resources.

To determine whether an objective has been met [7], SLA-available QoS metrics are evaluated based on measurable data about a service (e.g. response time, throughput, availability, and so on), performance during specified times, and periodic evaluations. SLAs include other observable objectives which are useful for service monitoring. These include compliance with differentiated service-level offerings, i.e. providing differentiated QoS for various types of customers, individualized service-level offerings and requests policing which ensures that the number requests per customer stays within a predefined limit. All these also need to be monitored and assessed. A key aspect of defining measurable objectives is to set warning thresholds and alerts for compliance failures. For instance, if the response time of a particular service is degrading then the client could be automatically routed to a back up service.

2 Service-Level Agreements (SLAs)

As organizations depend on business units, partners, and external service providers to furnish them with services, they rely on the use of SLAs to ensure that the chosen service provider delivers a guaranteed level of service quality. An SLA sets the expectations between the consumer and the provider. It helps define the relationship between the two parties. It is the cornerstone of how the service provider sets and maintains commitments to the service consumer. A properly specified SLA describes each service offered and addresses:

- how delivery of the service at the specified level of quality will become realized
- which metrics will be collected
- who will collect the metrics and how
- actions to be taken when the service is not delivered at the specified level of quality and who is responsible for performing them
- penalties for failure to deliver the service at the specified level of quality
- how and whether the SLA will evolve as technology changes (e.g., multi-core processors improve the provider's ability to reduce end-to-end latency) [1]

In the definition of an SLA, realistic and measurable commitments are important. Performing as promised is important, but swift and well communicated resolution of issues is even more important.

The challenge [9] for a new service and its associated SLA is to ensure that there exists a direct relationship between the architecture and the maximum levels of availability. Thus, an SLA cannot be created in a vacuum. An SLA must be defined with the infrastructure in mind. An exponential relationship exists between the levels of availability and the related cost. Some customers need higher levels of availability and are willing to pay more. Therefore, having different SLAs with different associated costs is a common approach.

An SLA may contain the following parts:

- *Purpose:* This field describes the reasons behind the creation of the SLA.
- *Parties:* This field describes the parties involved in the SLA and their respective roles, e.g. service provider and service consumer (client).
- *Validity period:* This field defines the period of time that the SLA will cover. This is delimited by start and end time of the agreement term.
- *Scope:* This field defines the services covered in the agreement.
- *Restrictions:* This field defines the necessary steps to be taken in order for the requested service levels to be provided.
- *Service-level objectives:* This field defines the levels of service that both the service customers and the service providers agree on, and usually includes a set of service level indicators, like availability, performance, and reliability. Each of these aspects of the service level will have a target level to achieve.
- *Penalties:* This field defines what sanctions should apply in case the service provider underperforms and is unable to meet the objectives specified is the SLA.
- *Optional services:* This field specifies any services that are not normally required by the user, but might be required in case on an exception.
- *Exclusion terms:* These specify what is not covered in the SLA.
- *Administration:* This field describes the processes and the measurable objectives in an SLA and defines the organizational authority for overseeing them.

SLAs can be either static or dynamic in nature. A static SLA is an SLA that generally remains unchanged for multiple service time intervals. Service time intervals may be calendar months for a business process that is subject to an SLA, or may be a transaction or any other measurable and relevant period of time for other processes. They are used for assessment of the QoS and are agreed between a service provider and service consumer. A dynamic SLA is an SLA that generally changes from service period to service period, to accommodate changes in provision of service.

To evaluate a Web service's SLA, specific QoS metrics that are measured over a time interval to a set of defined objectives, should be employed. Measurement of QoS levels in an SLA will ultimately involve tracing Web services through multi-domain (geographical, technological, application, and supplier) infrastructures. In a typical scenario, each Web service may interact with multiple Web services, switching between the roles of being a service provider in some interactions to being a consumer in other interactions. Each of these interactions could potentially be governed by an SLA. The metrics imposed by SLAs should correlate with the overall objectives of the services being provided. Thus, an important function that an SLA

accomplishes is addressing QoS at the source. This refers to the level of service that a particular service provides. [7]

The credibility [5] of SLAs is essential for the functioning of a service market. Unreliable SLA advertisements decrease the overall welfare of the market, since clients do not have accurate information for planning their business. As clients are usually required to pay for an SLA before receiving the requested service, providers have an opportunity to cheat. They may provide lower QoS than advertised, and thus save costs. It is therefore necessary to create incentives for service providers to respect their advertised SLAs by stating penalties that must be paid when the delivered QoS is less than promised.

2.1 Why is a Service Level Agreement Important?

A good SLA is important [9] because it sets boundaries and expectations for the following aspects of service provisioning.

- *Customer commitments*. Clearly defined promises reduce the chances of disappointing a customer. These promises also help staying focused on customer requirements and assuring that the internal processes follow the right direction.
- *Key performance indicators for the customer service*. By having these indicators established, it is easy to understand how they can be integrated in a quality improvement process. By doing so, improved customer satisfaction stays a clear objective.
- *Key performance indicators for the internal organizations*. A SLA drives internal processes by setting a clear, measurable standard of performance. Consequently, internal objectives become clearer and easier to measure.
- *The price of non-conformance*. If the SLA has penalties, non-performance can be costly. Therefore, by having penalties defined, the customer understands that the provider truly believes in its ability to achieve the set performance levels. It makes the relationship clear and positive.

2.2 SLA Life Cycle

Service Level Agreements have a certain life cycle, which they are running through from the initial preparation of the templates for an agreement up to the evaluation whether the agreement has been fulfilled, or, partly or completely violated. W. Sun et al [2] describe that the SLA life cycle consists of six phases shown in Figure 2.1.

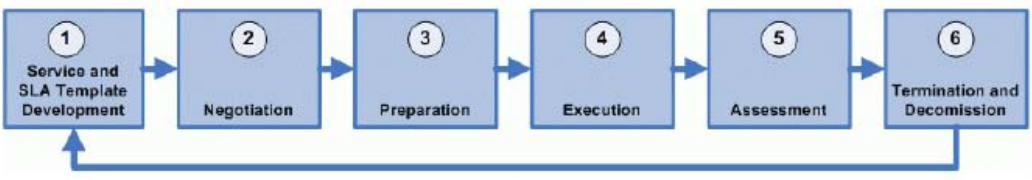

Figure 2.1: SLA Life Cycle

1) *Service and SLA Template Development:* This phase includes the identification of service consumer needs, the identification of appropriate service characteristics and parameters that can be offered given the service execution environment, and the preparation of standard SLA templates.

2) *Negotiation***:** This phase includes the negotiation of the specific values for the defined service parameters, the costs for the service consumer, the costs for the service provider when the SLA is violated, and the definition and periodicity of reports to be provided to the service consumer.

3) *Preparation***:** The service (or a specific instance of it) is prepared for consumption by the service consumer. This phase may require the reconfiguration of the resources that support service execution in order to meet SLA parameters.

4) *Execution***:** This phase is the actual operation of the service. It includes service execution and monitoring, real-time reporting, service quality validation, and real-time SLA violation processing.

5) *Assessment***:** This phase has two parts:
   a) Assessment of the SLA and the QoS that is provided to an individual consumer. QoS, consumer satisfaction, potential improvements, and changing requirements are reviewed periodically for each SLA.
   b) Assessment of the overall service. This assessment can be tied to an internal business review. Elements to be covered in this review are the QoS provided to all consumers, the need for the realignment of service goals and operations, the identification of service support problems, and the identification of the need for different service levels.
6) *Termination and Decommission:* This phase deals with termination of the service for reasons such as contract expiration or violation of contract, as well as, with the decommission of discontinued services.

2.3 Expressing SLAs in the WSLA Language

In this section, we provide a brief overview of the parts of the WSLA language [12] that relate to the definition of SLA parameters and the way they are monitored (Figure 2.2).

The *Parties* [4] section identifies the contractual parties and contains the technical properties of a party, i.e. their address and interface definitions (e.g. the ports for receiving notifications). The *Service Description* section of the SLA specifies the characteristics of the service and its observable parameters as follows: For every *Service Operation***,** one or more *Bindings***,** i.e., the transport encoding for the messages to be exchanged, may be specified. In addition, one or more *SLA Parameters* of the service may be specified. Examples of such SLA parameters are service *availability, throughput, or response time.* Every SLA parameter refers to one *Metric***,** which, in turn, may aggregate one or more other (composite or raw) metrics, according to a measurement directive or a function. Examples of composite metrics are *maximum response time of a service, average availability of a service, or minimum throughput of a service.* Examples of raw metrics are: *system uptime,*

service outage period, number of service invocations. Measurement Directives elements are used when the value of a Metric should be measured directly from a resource by probing or instrumentation of the system. There are seven types of measurement directives in the WSLA specification. Typical examples of measurement directives [6] are the uniform resource identifier of a hosted computer program, a protocol message (e.g., an SNMP GET message), the command for invoking scripts or compiled programs, or a query statement issued against a database or data warehouse. *Function* elements are used for a metric if the metric value is derived from the value of other metrics or constants. There are eighteen types of functions used in WSLA document. Examples of functions are formulas of arbitrary length containing mean, median, sum, minimum, maximum, and various other arithmetic operators, or time series constructors. For every function, a *Schedule* is specified. It defines the time intervals during which the functions are executed to retrieve and compute the metrics. These time intervals are specified by means of *start time, duration,* and *frequency.* Examples of the latter are *weekly, daily, hourly,* or *every minute. Obligations***,** the last section of an SLA, define the SLOs, guarantees and constraints that may be imposed on the SLA parameters.

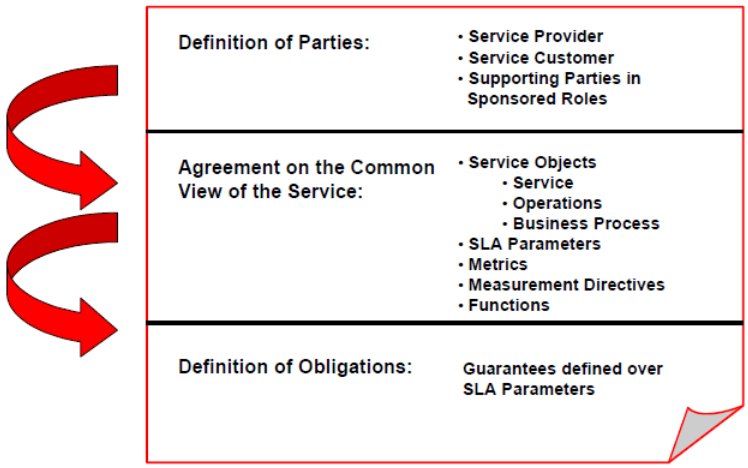

Figure 2.2: SLA structure as defined in WSLA

3 Monitoring of Web services

3.1 Services involved in SLA-compliance monitoring

During the contracting process, after the main elements of the SLA are agreed upon, customer and provider may define third parties[1] to which SLA monitoring tasks may be delegated.

[1] In the WSLA context we refer to these as *supporting parties.*

When the SLA is finalized, both provider and customer make the SLA document available for deployment. The *deployment service* is responsible for checking the validity of the SLA and distributing it either in full or in part to the supporting parties.

Figure 3.1 [3] illustrates the services involved in compliance monitoring when multiple parties are involved. Services that may be outsourced to third parties are either measurement services or condition evaluation services.

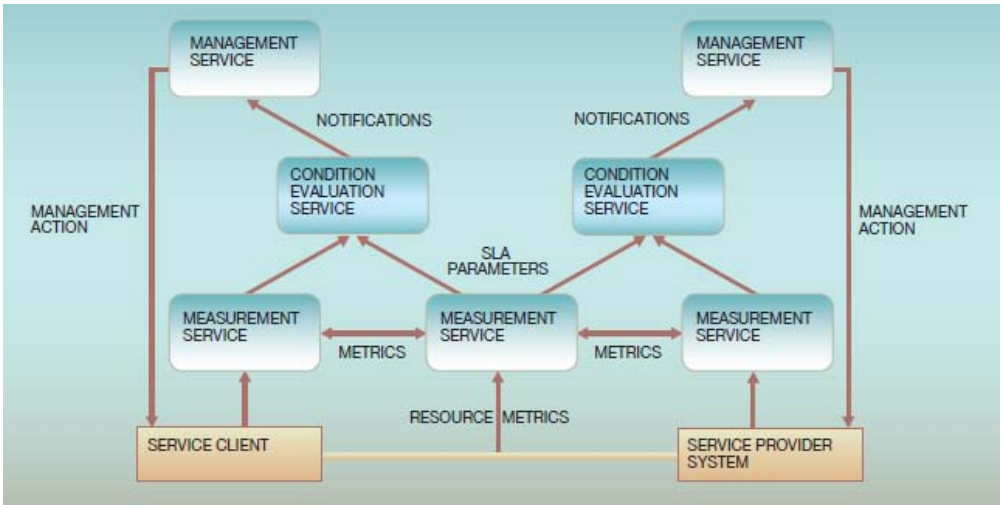

Figure 3.1: Services involved in SLA-compliance monitoring with multiple parties

The *measurement* service maintains information on the current system configuration and runtime information on the metrics that are part of the SLA. It measures SLA parameters, such as availability or response time, either from inside, by retrieving resource metrics directly from managed resources, or from outside the service provider's domain, for example, by probing or intercepting client transactions. A measurement service may measure all or a subset of the SLA parameters. Multiple measurement services may simultaneously measure the same metrics, e.g., a measurement service may be located within the provider's domain while another measurement service probes the service offered by the provider across the Internet from various locations.

As depicted in Figure 3.1, measurement services may be cascaded, that is, a third measurement service may be used to aggregate data computed by other measurement services. In this way, we refer to metrics that are retrieved directly from managed resources as *resource metrics*. *Composite metrics*, in contrast, are created by aggregating several resource (or other composite) metrics according to a specific algorithm, such as averaging one or more metrics over a specific amount of time or by breaking them down according to specific criteria (e.g., top 5 percent, minimum, maximum, mean, median etc.). This is usually done by a measurement service within a service provider's domain, but can be outsourced to a third-party measurement service as well.

The *condition evaluation service* is responsible for monitoring compliance of the SLA parameters at runtime with the agreed-upon service level objective (SLO) by comparing measured parameters against the thresholds defined in the SLA and notifying the management services of the customer and the provider. It obtains measured values of SLA parameters from one or more measurement services and tests

them against the guarantees given in the SLA. This can be done each time a new value is available, or periodically.

Finally, both service customer and provider have a *management service*. Upon receipt of a notification, the management service takes appropriate actions to correct a problem, as specified in the SLA. The main purpose of the management service is to execute corrective actions on behalf of the managed environment if a condition evaluation service discovers that a term of an SLA has been violated.

3.2 Monitoring and Managing SLAs

Figure 3.2 shows a conceptual architecture for an SLA monitoring and management infrastructure. Instrumentation is added to the service user and to the service provider to generate metrics. The measurement subsystem receives the metrics from instrumentation on one or more components and computes or combines data to produce values for the parameters specified in the SLA (i.e., the service level parameters). Figure 3.2 also shows measurement subsystem components on the service user and service provider sides, but there are also other alternatives. The measurement subsystem could exist only on the service user side or only on the service provider side, but the party providing measurement needs to be trusted by the other party. Measurement could also be partially or totally implemented by a third-party component running on a separate machine. As Figure 3.2 illustrates, the values of the SLA parameters are input for the evaluation procedure, which can run on

- either the service user or service provider
- both the service user and service provider
- a third-party machine

Figure 3.2: Conceptual Architecture for SLA Monitoring and Management

The evaluation procedure checks the values against the guaranteed conditions of the SLA. If any value violates a condition in the SLA, predefined actions are invoked. Action-handler components are responsible for processing each action request. While the functionality they execute will vary, it will likely include some form of violation notification or recording. Action handlers may exist on the service user machine, service provider machine, and/or third-party machines.

An important output of SLA management not shown in Figure 3.2 consists of reports that contain analyses of the metrics and thresholds. These reports can guide planning and implementation of improvements to the service infrastructure.

The same service may be invoked by different service users under different service level agreements. In that case, each service call may need to identify the agreement that applies to that invocation. Alternatives to agreement identification include tagging service calls with agreement identifiers, identifying the service user based on information exchanged at binding time, and selecting agreements based on the user.

4 Implementation

4.1 Implementation Tools

For the purposes of our work, we have chosen to use some tools for the implementation, among many alternative solutions. The criteria for our selection were the connectivity and the efficient collaboration between them. These tools are:

- Sun GlassFish Application Server
- Eclipse
- WSLA plug-in for Eclipse
- PostgreSQL

Eclispe and PostgreSQL are very well-known tools and we'll not provide more information for them.

4.1.1 Sun GlassFish Application Server

GlassFish Application Server[2] is an open-source application server implementation of Java EE 5. In project GlassFish, Web services are first-class objects that can be easily monitored and managed.

GlassFish can track and graphically display operational statistics, such as the number of requests per second, the average response time and the throughput. One

[2] https://glassfish.dev.java.net/

can enable monitoring for each of the Web services within an application and set the monitoring level to one of the following:

- **HIGH** — GlassFish monitors the response times, throughput, and the total number of requests, the faults, and the details of the SOAP message trace.
- **LOW** — GlassFish monitors only the response times, throughput, the total number of requests, and the faults.
- **OFF** — GlassFish collects no monitoring data.

If monitoring is on for a Web service, it applies to all the operations in that Web service.

4.1.2 WSLA plug-in for eclipse

There's an active community of third party Eclipse plug-in developers, both open source and commercial. As an Eclipse user, you're regularly rewarded with great new features from official Eclipse releases and from the plug-in development community. Such a plug-in is the WSLA plug-in[3] for eclipse developed by SOA-Blog.net. It is a free Eclipse plug-in that adds Web Service Level Agreement (WSLA) Language Support to the Eclipse platform. This tool is very useful as it provides a graphical interface to create WSLA documents with all its elements, including the Parties, the SLA Parameters and the Obligations.

The **WSLA plug-in** itself can be grouped into five main components:

- **WSLA Core Model:** provides access to WSLA model objects by mapping the XML document to a semantic model.
- **WSLA Multipage Editor:** a user-friendly form based editor to modify service level agreements that hides the complexity of XML. However, the editor also allows you to edit the XML source code directly, providing advanced editing capabilities such as syntax highlighting and outline views.
- **WSLA Report Generator:** service level agreements reports can be printed out for review or audition from a WSLA model. This is especially beneficial to communicate SLAs with non technical users.
- **WSLA Wizards:** assist in the creation and management of WSLA documents.
- **WSLA Help System:** a comprehensive and extensive help system familiarizes with the usage of the plug-in. The help system can be accessed via the packaged eclipse help, the invocation of context sensitive help or online by visiting the WSLA Information Center. In addition, the entire WSLA specification is included as a reference in the help system.

[3] http://soa-blog.net/index.php?/archives/29-WebService-Level-Agreement-WSLA-Eclipse-Plugin.html

Figure 4.1 shows a sample WSLA document, as extracted by the WSLA plug-in of eclipse. It is like an XML-editor, which discriminates with different colors the elements, the values and the types.

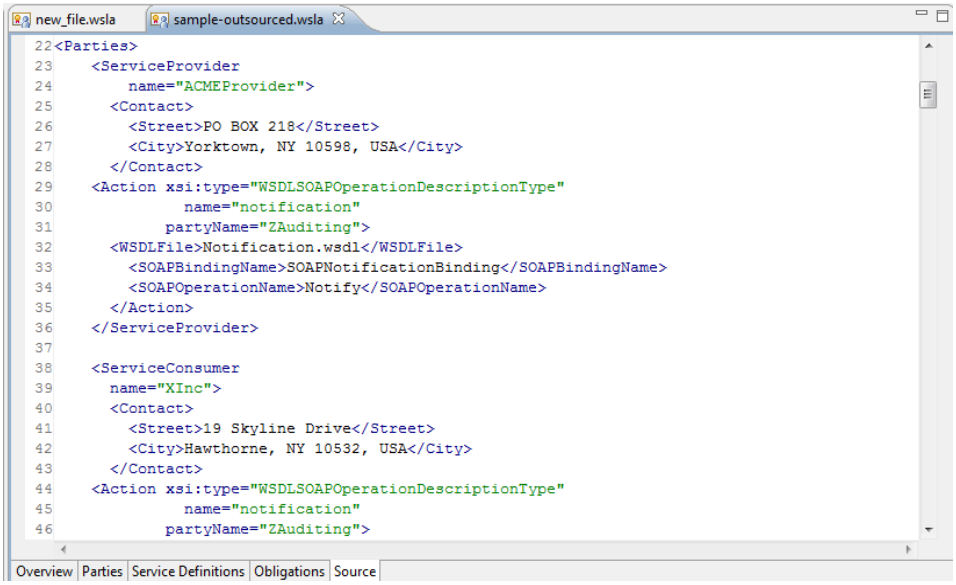

Figure 4.1: A sample WSLA document extracted from the eclipse plug-in

4.2 Implementation procedure

In this section we analyze the implementation procedure of our work. It consists of six steps that are given in detail below:

1. Creation of WSLA documents using WSLA Plug-in for Eclipse.
2. Parsing of the WSLA document.
3. Storing of the ServiceLevelObjectives in a PostgreSQL database.
4. Execution of the Web Service using Sun GlassFish Application Server and extraction of the available metrics.
5. Computing of further metrics using the metrics, provided by GlassFish.
6. Comparing the agreed metrics values of SLA with the metrics extracted by the monitoring system

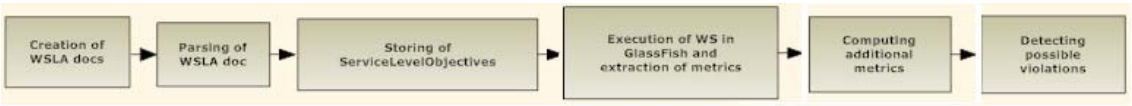

Figure 4.2: Steps of the implementation procedure

4.2.1 Creation of WSLA documents using WSLA Plug-in for Eclipse

For the purposes of our work it is essential to create WSLA documents, as it is difficult to have the real SLA documents the provider signs with the client. They are usually private documents that are not publicly available. For this reason, we used a WSLA plug-in for eclipse to create our WSLA documents.

4.2.2 Parsing of the WSLA document

As described in the previous section, a WSLA document has many elements, such as parties, service definition, obligations etc. From all of these, we are interested in the obligations that define the metrics values that were agreed between the service provider and the client. In order to collect this information, we made a parser that is basically an XML parser. This parser collects the following elements of the WSLA document:

- The Web service name
- The obliged party
- The start of validity period
- The end of validity period
- The predicate that applies for the specific value
- The name of the SLA parameter
- The agreed value of the SLA parameter and
- The EvaluationEvent

4.2.3 Storing of the ServiceLevelObjectives in a PostgreSQL database

The previous collected elements must be saved permanently, in order to be used later. For this reason, we have used a PostGreSQL database to save them. In this database, we have created a table with nine columns (plus one for the ID), each of which store the corresponding value. The web service name, the obliged party, the predicate, the SLA parameter name and the Evaluation Event are stored as text values, the start and validity period as date values and the agreed value of the SLA Parameter as a numeric value.

4.2.4 Execution of the Web Service on Sun GlassFish Application Server and extraction of the available metrics.

As described in the tools subchapter, Sun GlassFish Application Server is a very useful tool that allows us to manipulate web services uploaded on it. We use the monitoring feature to collect all the metrics that are available. The metrics values are also depicted with graphs along with the time (figure 4.3), but we use the asadmin console to collect them via command line. As mentioned below, there are three levels of monitoring in Glassfish. We set the monitoring level to HIGH, in order to have as many metrics as possible.

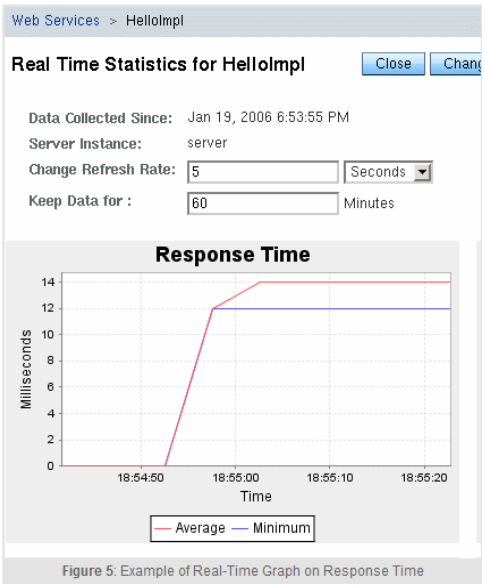

Figure 4.3: Graphical depiction of the Response Time

The command line results are more detailed, as they also provide composed metrics, such as min response time or average response time. Specifically, the extracted metrics values are:

- Average response time
- Min response time
- Max response time
- Response time
- Throughput
- Total number of authentication failures
- Total number of authentication successes
- Total number for requests that caused SOAP faults
- Total number of successful invocations of the method

4.2.5 Computing of further metrics using the metrics provided by GlassFish

There are some other metrics that can be calculated using the above metrics, provided by GlassFish. In our work [8] we also compute **availability** and **reliability**:

Availability is the probability that a service is up and running. It can be calculated in the following way:

$$availability = \frac{successfulInvocations}{TotalInvocations}$$

Reliability is the Ratio of the number of error messages to total messages and can be calculated in the following way:

$$reliability = \frac{totalFaults}{totalMessages} x100$$

4.2.6 Comparing the agreed metrics values of SLA with the metrics extracted by the monitoring system

Now that we have the metrics values of the web service and the agreed values of the SLA, we can do the condition evaluation. Thus, we check if the web service running on the application server has a corresponding SLA with the agreed values stored in the database. If there is such a matching, we can compare the metric values to see and if there is any violation. Once the Condition Evaluation Service has determined that an SLO has been violated, corrective management actions need to be carried out. The Management Service will retrieve the appropriate actions to correct the problem, as specified in the SLA.

4.3 UML diagrams

The UML class diagram of our implementation is shown in Figure 4.4. The ServiceLevelObjectiveClass implements objects of the corresponding type. It defines all the fields of the ServiceLevelObjective as string type and contains all the appropriate set and get methods to handle them.

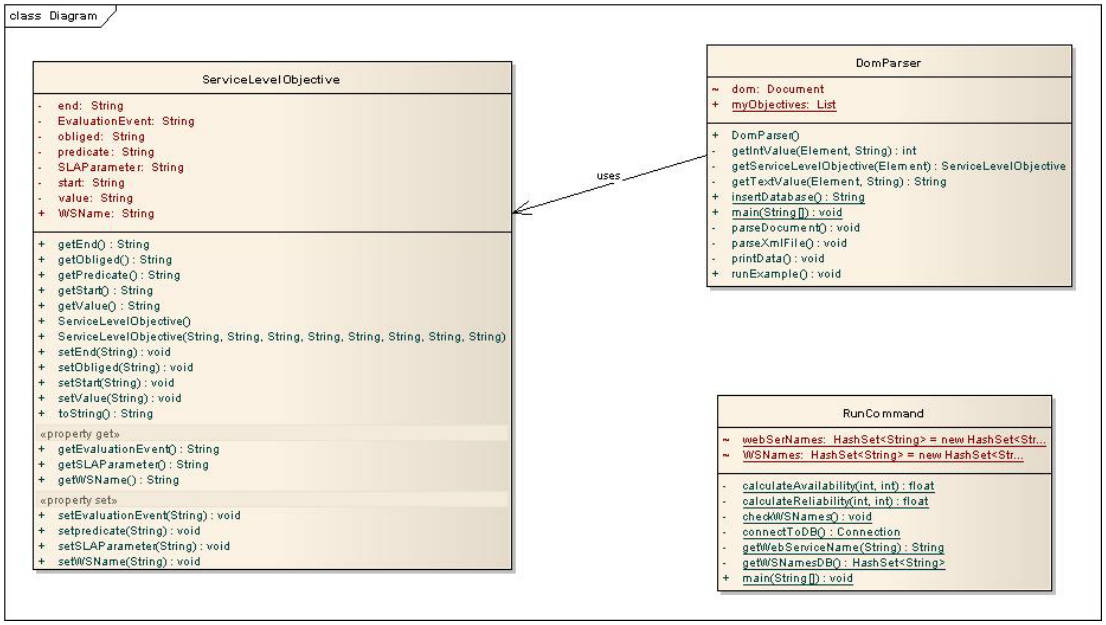

Figure 4.4: The UML class diagram of our implementation

The role of the DomParser class (step 2) is to parse the WSLA document. As we have already mentioned, WSLA is an XML-based language, so a simple XML parser is needed. Methods parseDocument() and parseXmlFile() are responsible for the parsing, getIntValue(), getServiceLevelObjective() and getTextValue() methods are used to get the values of each of the corresponding types and the printData()

method prints the results (name and values of metrics), after we have inserted the parsed values in the PostGreSQL database with the insertDatabase() method.

The RunCommand class (steps 4-6) collects the metrics from the GlassFish ApplicationServer and computes additional metrics, such as availability and reliability. It checks the web service names to see if there is a corresponding WSLA document for the deployed Web services at the database and if there is such a correlation it checks for possible violations in all available metric values. An alert message is shown for every violation.

The UML sequence diagram is shown in Figure 4.5 and corresponds to steps 4-6 of the previous section. In the beginning, the user asks the *condition evaluator* if there are any violations in the metrics values. The *condition evaluator* follows four steps to complete the procedure.

1. It asks for and gets the metrics from the *Glassfish Application Server*.
2. The *metric computer*, after getting the retrieved metrics from *Glassfish*, computes some further metrics and returns them to the condition evaluator.
3. The *condition evaluator* gets the agreed metric values and the predicate from the WSLA database and
4. The *condition evaluator* computes possible violations and returns them to the user.

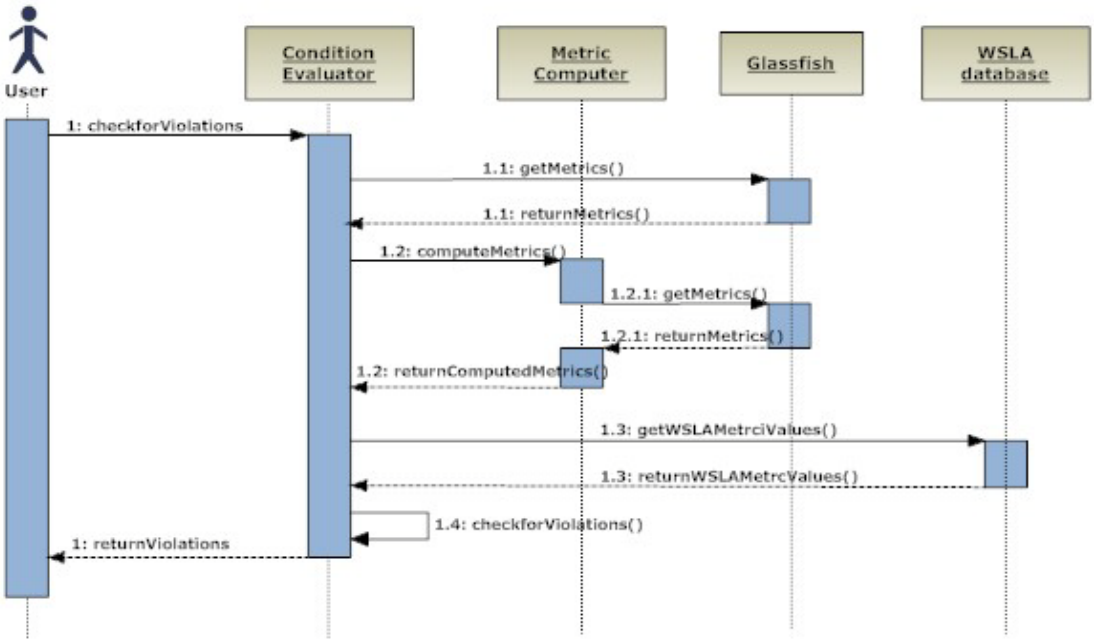

Figure 4.5: The UML sequence diagram

5 Conclusions and Future Work

In this technical report, which derived from work in [8], we propose a system that combines the monitoring system of Sun Application Server GlassFish with SLA documents, written in WSLA, an XML-based language. Our system requires the availability of both the Web services to be deployed on the application server and the corresponding SLA of the Web service. If these requirements are fulfilled, then the system compares the pre-agreed metric conditions with the monitored metric values and if there is a violation, it alerts the interested parties with the detected differences. The results are satisfying, compared to other similar systems, as they seem to be fast and precise. Furthermore, these results are very helpful for the service provider, who can take corrective actions in case metric values don't comply with SLAs. In addition, statistical analysis of the results can extract many useful conclusions to assist the provider in developing new and more effective Web services.

As future work, it would be ideal to make our monitoring system work at runtime with the Web service execution engine. In this way, the provider could take statistical results and correct any violations very quickly and without any mediators. Another interesting direction is to focus on the corrective actions after a violation is detected. This can be performed by another management service that would take as input the condition evaluation results and then, if appropriate, would try to make the service compliant with the SLA document. All this procedure would be ideal to be distributed.

References

[1] **L. Baresi, C. Ghezzi and S. Guinea.** "Smart Monitors for Composed Services". In *Procs of ICSOC'04*, pages 308 - 315, New York, USA, November 2004.

[2] **P. Bianco Philip, G. Lewis and P. Merson.** "Service Level Agreements in Service-Oriented Architecture Environments". *Technical Note of Software Engineering Institute*, September 2008.

[3] **A. Dan, D. Davis, R. Kearney, A. Keller, R. King, D. Kuebler, H. Ludwig, M. Polan, M. Spreitzer and A. Youssef.** "Web services on demand: WSLA-driven automated management". In *IBM Systems journal*, Vol. 43, No 1, 2004.

[4] **M. Debusmann and A. Keller.** "SLA-driven management of distributed systems using the Common Information Model". In *Procs of IM 2003*, pages 563-576, Colorado, USA, March 2003.

[5] **R. Jurca, W. Binder and B. Faltings.** "Reliable QoS Monitoring Based on Client Feedback". In *Procs of WWW'07*, pages 1003-1012, Banff, Alberta, Canada, May 2007.

[6] **R. Kassab and Aad van Moorsel.** "Mapping WSLA on Reward Constructs in Mobius", In *Procs of UKPEW 2008, London, England, July 2008.*

[7] **M. Papazoglou.** "Web Services: principles and tenchnology", Pearson Publications, 2008.
[8] **C.Zeginis,** “Monitoring the QoS of Web services using SLAs – Computing metrics for composed services”, Master Thesis, Greece, Heraklion, March 2009
[9] **E. Wustenhoff.** "Service Level Agreement in the Data Center". *Sun BluePrints*, April 2002.
[10] **K. Mahbub and G. Spanoudakis.** "Run-time Monitoring of Requirements for Systems Composed of Web-Services: Initial Implementaion and Evaluation Experience". In *Procs of ICWS 2005*, pages 257-265, Orlando, Florida, USA, June 2005.
[11] **L. Zeng, H. Lei and H. Chang.** "Monitoring the QoS for Web Services", In *Procs of ICSOC 2007*, Springer Publications, pages 132-144, Vienna, Austria, September 2007.
[12] **A. Keller and H. Ludwig.** "The WSLA Framework: Specifying and Monitoring of Service Level Agreements for Web Services". *IBM research report RC22456*, May 2002.