Software Engineering Institute

Service Level Agreements in Service-Oriented Architecture Environments

Philip Bianco
Grace A. Lewis
Paulo Merson

September 2008

TECHNICAL NOTE
CMU/SEI-2008-TN-021

Software Architecture Technology Initiative
Integration of Software-Intensive Systems Initiative

http://www.sei.cmu.edu

Carnegie Mellon

This report was prepared for the

SEI Administrative Agent
ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2100

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about SEI reports, please visit the publications section of our website (http://www.sei.cmu.edu/publications).

Table of Contents

List of Figures

List of Tables

Abstract

Quality attribute requirements play an important role in service selection in a service-oriented architecture environment. It is easy to envision finding several services that fulfill the functional requirements but fail to meet important quality attribute measures. Service level agreements provide the architect with a specification of the verifiable quality characteristics that the service will provide. Such a specification allows the architect to select the service that best supports the system's quality attribute requirements. This report surveys the state of practice in service level agreement specification and offers guidelines on how to assure that services are provided with high availability, security, performance, and other required qualities. In addition, this report discusses the quality properties that have been expressed in service level agreements, and it considers the mechanisms used to assure service quality by contract and those used by service providers to achieve and monitor quality properties. Also, this report examines the support for expressing service quality requirements in existing service technologies and describes areas where more research is needed.

1 Introduction

Quality attribute requirements play an important role in service selection in service-oriented architecture (SOA) environments.

SOA is an approach for designing, developing, deploying, and managing systems that are characterized by coarse-grained services that represent reusable business functionality. These systems serve consumers that compose applications or systems using the functionality provided by these services through standard interfaces. A very common technology for SOA implementation is web services, in which service interfaces are described using the Web Services Description Language (WSDL), and Extensible Markup Language (XML) payload is transmitted using SOAP over Hypertext Transfer Protocol (HTTP). A set of additional web service specifications (e.g., WS-BPEL, WS-Security, WS-ReliableMessaging) can be used in this context to support operations such as service orchestration and management and qualities such as security and availability.

Service-oriented systems have gone through a gradual evolution. Their first generation was based on monolithic components that could be reconfigured at compile time. Currently, their second generation is based on vertically integrated components that can be adapted and reconfigured at installation time and, to some extent, at runtime. In the future, we will see an emerging third generation of service-oriented systems that will be cross-vertically integrated, context-sensitive, and reconfigurable in an autonomic, ad hoc, and on-demand manner [IST 2006].

At design time during service selection, an architect of a first- or second-generation service-oriented system will search for services to fulfill some set of functionality. The architect will make the selection based on desired functional requirements as well as quality attribute requirements that are important to system stakeholders. These qualities will probably be described in a service level agreement (SLA). In the case of third-generation, service-oriented systems, this service selection will likely occur at runtime, and the SLA will need to be in a machine-readable format that can be interpreted at runtime.

An SLA is part of the contract between the service consumer and service provider and formally defines the level of service. The TeleManagement (TM) Forum's *SLA Management Handbook* underscores the importance of SLAs for the telecommunications industry:

> *It is the Service Level Agreement (SLA) that defines the availability, reliability and performance quality of delivered telecommunication services and networks to ensure the right information gets to the right person in the right location at the right time, safely and securely. The rapid evolution of the telecommunications market is leading to the introduction of new services and new networking technologies in ever-shorter time scales. SLAs are tools that help support and encourage Customers to use these new technologies and services as they provide a commitment from SPs (Service Providers) for specified performance levels [TM Forum 2008].*

Organizations seek to develop SLAs for various reasons. From a simple perspective, an SLA is developed between two parties to spell out who is responsible for what, what each party will do, and sometimes more importantly what each party will not do. From a business perspective, an SLA is developed for these reasons:

- New commercial services entering a competitive market can use SLAs to establish themselves as reliable providers and hence attract customers.
- Service interactions across organizations increasingly permeate business processes and become critical to fulfilling business goals. SLAs establish a commitment to quality levels required by service users and providers to interact effectively. Moreover, SLA management helps customers validate and supervise the quality of services through scheduled and on-exception reports.
- In one vision of the future, standardized services will be offered by multiple service providers. For example, realtors provide many different levels of service to home sellers, and all must provide a minimum level of service. That minimum service might offer use of the Multiple Listing Service (MLS) for the home at the cost of 1% of the home's posted sale price. As part of enhanced service, the realtor might actively market and handle the sale for the owner at the cost of 6 or 7% percent of the home's final sale price.

  In the context of web services, standardized services may be offered for common well-defined tasks such as tax return preparation, credit history requests, insurance claims, and banking. A common SLA that applies to a standard service is an important quality management artifact for service users and the regulatory agency responsible for overseeing the service.
- Many organizations provide services that depend on services from other organizations. For example, an online travel-booking system may interact with (1) banks to authorize credit cards, (2) hotel chains to check availability and reserve rooms, (3) airlines for checking and booking flights, and (4) car rental companies. In addition, each organization providing a service may, in its turn, use services from other organizations (commonly called layered or composite services). In some situations, the unavailability or poor performance of one of the services may compromise the whole customer experience. In a multi-party composite service situation, SLAs can be used to identify the party that is responsible in case of a problem.

The goal of this technical note is to provide an overview of the state of the practice in the development and management of SLAs in the context of SOA.[1] We survey this state regarding SLAs in SOA environments—mainly web services—and offer guidelines to organizations that plan to contract external services or move to the third generation of service-oriented systems where SLAs in machine-readable format become enablers.

In this report, we consider important questions such as

- Currently, what mechanisms can be used to ensure quality of service (QoS) by contract?
- What quality attributes can be expressed in SLAs?
- What mechanisms are used by service providers to achieve and monitor those quality attributes?
- What technology support—such as tools, specifications, and standards—is available to express quality requirements, and what support will be considered in the future?

[1] While details of the legal and financial aspects of SLAs are significant, we do not address them in this technical note.

2 Background of SLAs

SLAs have been used in IT organizations and departments for many years. The definition of SLAs in a SOA context is more recent but is becoming extremely important as service-oriented systems are starting to cross organizational boundaries and third-party service providers are starting to emerge. The current state of the practice for SLAs in an SOA context is heavily influenced by the IT management processes and procedures guiding SLAs.

As background for the rest of the report, in this section we look at SLAs in IT organizations and SOA environments and at qualities that can be specified in the SLAs that are associated to services in SOA environments.

2.1 IT SLAs

SLAs have been used for many years in IT organizations and departments to identify the support requirements for internal and external customers of IT services.

In this context, an SLA sets the expectations of both the IT service consumer and provider. It is common for IT service providers to deliver services at different levels of quality based on the cost paid for a service. An SLA is valuable for helping all parties understand the tradeoffs inherent between cost, schedule, and quality because their relationship is stated explicitly. As with any type of contract, the existence of an SLA cannot guarantee that all promises will be kept, but it does define what will happen if those promises are not kept. “An SLA cannot guarantee that you will get the service it describes, any more than a warranty can guarantee that your car will never break down. In particular, an SLA cannot make a good service out of a bad one. At the same time, an SLA can mitigate the risk of choosing a bad service” [Allen 2006]. A “good” service is one that meets the needs of the service customer in terms of both quality and suitability.

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 doing 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)

IT SLAs are enforced by service management processes and procedures. An example framework for IT service management (ITSM) is the Information Technology Infrastructure Library (ITIL), which has been widely used since the mid 1990s. The ITIL focuses on the activities involved with the deployment, operation, support, and optimization of applications. Its main purpose is to ensure that deployed applications can meet their defined service levels [Allen 2006]. The ITIL provides several

key practice areas for ITSM. Below, we briefly describe the relationships between a subset of these areas that are relevant to SLAs:

- **Service level management** (SLM) is the main practice area for managing and maintaining QoS. This process area focuses on improving the QoS by continuously reviewing the quality of the services provided by an IT organization. SLM provides input to all service management processes and receives input from service support processes.
- The **change management** process area focuses on managing change in an efficient and consistent manner. It provides input to SLM on how infrastructure changes can affect QoS.
- The **incident management** process area's main purpose is to minimize the disruption that incidents cause to a business. It provides input to SLM on violations to agreements, who should be notified, historical data and trends, and any other actions that may be needed.
- The **configuration management** process area's main purpose is to identify, verify, control, and maintain configuration items. This area is also responsible for managing the relationships between configuration items. It identifies the services affected by problems with configuration items and provides this information to SLM.
- The **capacity management** process area's activities include monitoring, analyzing, tuning, and reporting on service performance to produce SLA recommendations.

SLAs for IT typically fall into three categories:

1. **basic** – a single SLA with well-established metrics that are measured and/or verified. The collection of these metrics is typically done manually.
2. **medium** – the introduction of multi-level quality based on the cost of the service. The objective is to balance the levels of quality and cost. Automatic data collection enables comprehensive reporting to IT managers and stakeholders.
3. **advanced** – dynamic allocation of resources to meet demand as business needs evolve

2.2 SLAs for SOA Environments

The size, complexity, and scope of service-oriented systems continue to grow as these systems are starting to cross organizational boundaries, and a market of third-party services is emerging, as evidenced by companies such as

- SEEC, Inc – insurance and financial service provider (http://www.seec.com)
- Amazon web services – services for businesses, consumers, Amazon associates, sellers, and other developers (http://www.amazon.com)
- IBM SOA Business Catalog – SOA assets in general (http://catalog.lotus.com)
- StrikeIron – data service broker (http://www.strikeiron.com)
- eBay developers program – services that can be used or composed to enhance listings, manage bids, generate commissions, and perform other functions related to online auctions (http://developer.ebay.com/)

Theoretically, architects compose services that span enterprise boundaries into larger services that represent business processes with the intention of exposing these services for use by systems and applications. This new dynamic business vision for system building requires the complex integration of business processes over the internet. Ensuring the quality of services provided over the internet is

an enormous challenge because the internet is dynamic and unpredictable. Some of these challenges are

- poor performance of standard protocols
- security issues
- infrastructure failures such as the undersea cable damage that crippled businesses for several days in parts of the Middle East, Africa, and Asia in January 2008 [BBC 2008]

SLAs can be used when a certain level of verifiable quality is required. A properly negotiated and documented SLA can be quite useful for an architect who needs to test and deploy new services. The key to crafting SLAs is to provide enough information or verifiable metrics for a service consumer to preselect services based on the desired level of quality. This provision can help an architect avoid wasting time evaluating unsuitable candidates. Common examples of metrics defined in an SLA are latency, the number of transactions handled per unit of time, cost per invocation, and the length of time the service will be supported.

Typically, SLAs are defined in plain text, using templates or toolkits. Service providers instrument their systems in such a way that measurements are collected and then compared to the metrics specified in the SLA. Appropriate actions are taken based on the severity of the SLA violations detected. For example, if the violation was a failure to respond to requests at the specified latency, which could reduce revenue, the related action might be to dynamically reconfigure the service or—if the problem is severe enough—to select a new service.

An active area of work is to define machine-readable SLAs. One driver for doing that is to support automated service discovery and the selection of services based on qualities specified in the SLA. Another more ambitious driver is to support the negotiation of the terms specified in an SLA. Creating the automated support for achieving dynamic negotiation of those terms (e.g., paying more for faster response times) and the ability to select different levels of quality at runtime is challenging [Chatterjee 2005] and requires

- a sufficiently expressive language that provides the ability to encode quality attribute specifications and to describe the different levels of qualities that a single-service implementation can provide
- a language for allowing consumers to query providers in order to find services that meet their desired level of quality
- a mechanism that facilitates the selection of quality levels based on runtime conditions such as the number of concurrent users
- the ability to assign priority to requests according to defined rules, such as lowering the priority of requests from a consumer that exceeds a prespecified number of transactions in a given time period
- the ability to support logical expressions for dynamic negotiation of quality attribute tradeoffs

We further discuss the need for machine-readable SLAs in Section 3.2.

2.3 Qualities That Can Be Defined in an SLA

In theory, it is possible to specify any quality in an SLA, provided that all parties understand how to measure or verify its achievement. We've seen two categories of qualities that can be specified in SLAs: measurable and unmeasurable. Measurable qualities can be measured automatically using metrics; for example, the percentage of time a system is available. Unmeasurable qualities are those that cannot be measured automatically from a given viewpoint; for example, determining the cost of changing a service (modifiability) is difficult to automate [Dobson 2005].

Measurable Qualities

- **Accuracy** is concerned with the error rate of the service. It is possible to specify the average number of errors over a given time period.
- **Availability** is concerned with the mean time to failure for services, and the SLAs typically describe the consequences associated with these failures. Availability is typically measured by the probability that the system will be operational when needed. It is possible to specify
  - the system's response when a failure occurs
  - the time it takes to recognize a malfunction
  - how long it takes to recover from a failure
  - whether error handling is used to mask failures
  - the downtime necessary to implement upgrades (may be zero)
  - the percentage of time the system is available outside of planned maintenance time
- **Capacity** is the number of concurrent requests that can be handled by the service in a given time period. It is possible to specify the maximum number of concurrent requests that can be handled by a service in a set block of time.
- **Cost** is concerned with the cost of each service request. It is possible to specify
  - the cost per request
  - the cost based on the size of the data
  - cost differences related to peak usage times
- **Latency** is concerned with the maximum amount of time between the arrival of a request and the completion of that request.
- **Provisioning-related time** (e.g., the time it takes for a new client's account to become operational)
- **Reliable messaging** is concerned with the guarantee of message delivery. It is possible to specify
  - how message delivery is guaranteed (e.g., exactly once, at most once)
  - whether the service supports delivering messages in the proper order
- **Scalability** is concerned with the ability of the service to increase the number of successful operations completed over a given time period. It is possible to specify the maximum number of such operations.

Unmeasurable Qualities

- **Interoperability** is concerned with the ability of a collection of communicating entities to share specific information and operate on it according to an agreed-upon operational semantics [Brownsword 2004]. It is possible to specify the standards supported by the service and to verify

them at runtime. Significant challenges still need to be overcome to achieve semantic interoperability at runtime.

- **Modifiability**, in this context, is concerned with how often a service is likely to change. It is possible to specify how often the service's
  - interface changes
  - implementation changes
- **Security** is concerned with the system's ability to resist unauthorized usage, while providing legitimate users with access to the service. Security is also characterized as a system providing non-repudiation, confidentiality, integrity, assurance, and auditing. It is possible to specify the methods for
  - authenticating services or users
  - authorizing services or users
  - encrypting the data

3 State of the Practice in SLAs for SOA Environments

3.1 Standards and Tools to Support SLAs

Several standards and tools support SLAs for web services—a technology commonly used to implement SOA. The maturity of these standards and tools varies.

3.1.1 Standards

There is no standard for the specification of SLAs. The most referenced and complete specifications that relate to SLAs for SOA environments, and in particular web services, are the Web Service Level Agreement (WSLA) language and framework and the Web Services Agreement Specification (WS-Agreement).

The WSLA is a specification and reference implementation developed by IBM that provides detailed SLA specification capabilities that enable the runtime monitoring of SLA compliance. Section 3.2.1 provides more details about the WSLA.

The WS-Agreement from the Open Grid Forum (OGF) is a

> *Web Services protocol for establishing agreement between two parties, such as between a service provider and consumer, using an extensible XML language for specifying the nature of the agreement, and agreement templates to facilitate discovery of compatible agreement parties. The specification consists of three parts which may be used in a composable manner: a schema for specifying an agreement, a schema for specifying an agreement template, and a set of port types and operations for managing agreement life-cycle, including creation, expiration, and monitoring of agreement states* [Andrieux 2007].

Section 3.2.2 provides more details about the WS-Agreement.

Other ongoing research projects for SLA specification include

- **SLAng**, a language for SLAs under development as part of the Trusted and Quality of Service Aware Provision of Application Services (TAPAS) project at University College London (UCL) [TAPAS 2005]. SLAng, which is currently under development, is defined using a combination of Meta-Object Facility (MOF), Object Constraint Language (OCL), and natural language.
- **Rule-Based Service Level Agreements** (RBSLA) is a project that uses knowledge representation concepts for the specification of SLAs. It provides a set of abstract language constructs (based on RuleML) to define SLA/policy rules. An implementation of RBSLA is available for download via SourceForge [Paschke 2006a].

Also, work is currently being done in the specification of service qualities that are part of SLAs. The most advanced work in this area is the WS-Policy, a recommendation specification developed by the World Wide Web Consortium (W3C). This specification allows web services to advertise their policies and web service consumers to specify their policy requirements [W3C 2007a]. Major contributors to the WS-Policy are IBM, BEA, Microsoft, SAP, Sonic, and VeriSign.

The WS-PolicyAttachment is a specification usually used in conjunction with the WS-Policy for attaching policy assertions to policy subjects [W3C 2006]. Because the WS-Policy is a specification

for associating policies to web services (rather than a specification for expressing the actual policy assertions), it is often used in conjunction with other specifications that conform to the WS-Policy for associating specific policies. These specifications include

- WS Security Policy [OASIS 2007a]
- WS Reliable Messaging Policy [OASIS 2007b]
- WS Atomic Transaction [OASIS 2007c]
- WS Business Activity Framework [OASIS 2007d]
- WS Addressing 1.0 - Metadata [W3C 2007b]

There are several implementations of the WS-Policy, including Apache Neethi [Apache 2008], and middleware called GlueQoS, which is based on extensions to the WS-Policy [Wohlstadter 2004].

Other mechanisms for the specification of nonfunctional properties that are in different stages of development and adoption include the following:

- The Web Services Offering Language (WSOL) enables the "formal specification of functional constraints, some QoS constraints, simple access rights (for the differentiation of service), price, and relationships with other service offerings of the same web service" [Tosic 2002]. While research continues on the applications and implementations of the WSOL, there are no known plans to turn it into a standard or specification.
- The Web Service Management Layer (WSML) goes between an application and web services and allows the specification of nonfunctional characteristics of a service. Version 0.5.1 was released in 2006 [SSEL 2003].
- The Web Service Modeling Ontology (WSMO) is used to specify aspects of semantic web services. It contains a way of describing nonfunctional properties such as cost, accuracy, network-related QoS, performance, reliability, robustness, scalability, and security. The latest draft was published in 2006 [de Bruijn 2006].

Another area of standardization is that of web services management. Web Services Distributed Management (WSDM) is a standard specification developed by the Organization for the Advancement of Structured Information Standards (OASIS). The goal of WSDM is to "define a web services architecture to manage distributed resources" [OASIS 2006]. Basically, the intent of WSDM was to produce a set of standards that specify a common messaging protocol for managed resources and their consumers. Although not specifically related to SLA specification, tools and products that implement this standard would be more easily integrated into applications and infrastructures in order to define, manage, and monitor properties typically specified in SLAs.

3.1.2 Tools and Products

Tools and products with functionality for the creation, management, and runtime monitoring of SLAs in SOA environments—more specifically web services—fall under the product categories of "SLA management," SLA life-cycle management," or "WS management." A simple web search that we conducted located several tools and products that fall into this category. That list, shown in Table 1, is not meant to be comprehensive or an endorsement of these products. They are listed in alphabetical order, and their descriptions were taken from vendors' websites.

Table 1: Example Tools and Products for SLA Management

Name	Vendor	Type	Functionality
Actional	Progress	Add-on to vendor's Sonic ESB product	• Definition of SLAs and SLA metrics on an individual and group level according to service performance and business context requirements • Runtime logging of statistics such as throughput, errors, requests, and response times • Enforcement of SLAs between consumers and providers, logging of SLA violations and customizable alerts, audit logs, and transaction snapshots • Dynamic policy management (e.g., dynamic request routing)
Intersperse	Tidal Software	SOA Management and Monitoring Product	• Runtime monitoring across multiple application servers to create detailed end-to-end transaction monitoring • Performance monitoring at both the component (detailed) and web service (message payload) levels • Performance analysis and notification • Control for automating corrective action to create self-healing solutions • Definition of metrics for root-cause analysis
Service Manager	SOA Software	SOA Management and Security Product	• Runtime monitoring of the performance, throughput, and usage of services and applications • Central policy and metadata definition and management capabilities for runtime policy management • Alert and exception management • Performance, throughput, and reliability monitoring of services provided to both internal and external customers • Dynamic routing to automatically correct performance and reliability issues
SOA Management System	AmberPoint	SOA Management	• Service level definition and management • Segmentation and prioritization of service resources by business criteria • Alerts, impact analysis, and runtime mitigation • Usage analysis • Runtime evaluation of service level objectives
TrueView for SOA and web services	Symphoniq	Service Level Assurance Solution	• Runtime monitoring of end-to-end user transactions • Root-cause analysis of performance problems

The major SOA product suite vendors and some of the traditional network management vendors incorporate service management or SLA management and monitoring as part of their offerings. Examples are BMC Software (BMC Performance Manager), Oracle BEA (BEA Aqualogic SOA Management), CA (CA Service Management), HP (HP SOA Center Suite), IBM (Tivoli SOA Management), Tibco (Service Performance Manager), and Oracle (Oracle Web Services Manager).

3.2 Modeling and Formalization of SLAs

Although managing and monitoring the quality levels of services rely heavily on automated tools, the actual SLA specification is typically a plain-text document, and sometimes it is an informative document published online. It might be signed by all the parties involved and imply legal binding. An example of a document with legal obligations is the *Amazon S3 Service Level Agreement* [Amazon 2008]. In that document, Amazon declares the company's commitment to making the Simple Storage Service (S3) available with a monthly uptime of at least 99.9%. The SLA describes how a service user is entitled to a $1 credit that can be used against the user's Amazon S3 payments if the user experiences low availability. Availability is documented through message logs of the service user system that indicate the service outage.

Although SLAs are easy to create in plain text, it is better to create them in a machine-readable format, such as XML, for these reasons:

- A machine-readable format supports automatic negotiation between service users and providers.
- Sometimes the SLA specifies measures that should be taken by the service user and/or service provider when a deviation from the SLA or a failure to meet the asserted service qualities occurs. In addition, machine-readable SLAs enable measures that can be triggered automatically (e.g., an email notification).
- A billing system can parse the SLA in order to obtain the rules to automatically calculate charges to the service user.
- An automated SLA management system that measures and monitors the quality parameters uses the SLA as input.

Standard formats for SLA documents, such as IBM's WSLA framework and the WS-Agreement specification, have been proposed by vendors, researchers, and industry consortia. The WSLA was a research project that was used in experimental systems within IBM, such as the Web Services Toolkit (WSTK) and Emerging Technologies Toolkit (ETTK), as well as in some IBM internal AIX-based systems.[2] Although the WSLA specification is available for download, few cases studies have been reported outside of IBM. It seems that the specification language has not been widely adopted but rather superseded by the WS-Agreement [Seidel 2007]. Indeed, core concepts of the WSLA were brought into the WS-Agreement, which also contains ideas from the Service Negotiation and Acquisition Protocol (SNAP).[2, 3] Because both the WSLA and WS-Agreement continue to be available, we describe them in the following subsections. Other formats were presented in Section 3.1.1.

3.2.1 The WSLA

The WSLA framework was proposed by IBM in 2001 [Ludwig 2003]. Its language is based on XML and defined as an XML schema. Primarily, the WSLA allows the creation of machine-readable SLAs for services implemented using web services technology that define service interfaces in the WSDL. However, the WSLA language is extensible to deal with other service-based technologies and other

---

[2] The source for this information was personal communication between one of the authors and Heiko Ludwig on June 23, 2008.

[3] The SNAP supports the reliable management of remote SLAs [Czajkowski 2002].

technical fields such as network storage. The WSLA framework comprises several monitoring services that can be replaced with services developed by third parties.

An SLA created using the WSLA language typically contains the following sections:

- **a description of the parties and the interfaces they expose to the other parties**. A party is an organization, not a computational element. However, it may need to implement some web services to handle the actions required by the SLA. For example, a party that has the role of evaluating performance measures collected from services under the scope of the SLA may need to implement a web service that encompasses the notification of other parties when measures are beyond certain thresholds.

  The information provided about each party should include contact information and a description of the organization, as well as the name of a WSDL file that contains the interface description of the web services the party implements.

  The signatory parties are typically the service consumer and the service provider, but either or both of them may commission part of the SLA management responsibility to additional, supporting parties. The following is a snippet of a WSLA SLA declaring a service provider, a service consumer, and an additional (supporting) party:

```
<Parties>
    <ServiceProvider name="ACMEProvider">
        <Contact>
            <Street>PO BOX 218</Street>
            <City>Yorktown, NY 10598, USA</City>
        </Contact>
        <Action xsi:type="WSDLSOAPOperationDescriptionType"
name="notification" partyName="ZAuditing">
            <WSDLFile>Notification.wsdl</WSDLFile>
            <SOAPBindingName>SOAPNotificationBinding</SOAPBindingName>
            <SOAPOperationName>Notify</SOAPOperationName>
        </Action>
    </ServiceProvider>
    <ServiceConsumer name="XInc">
        <Contact>
            <Street>19 Skyline Drive</Street>
            <City>Hawthorne, NY 10532, USA</City>
        </Contact>
        . . .
    </ServiceConsumer>
    <SupportingParty name="ZAuditing" role="ConditionEvaluationService">
        . . .
        <Action xsi:type="WSDLSOAPOperationDescriptionType"
name="parameterUpdate" partyName="ZAuditing">
            <WSDLFile>ParameterUpdate.wsdl</WSDLFile>
            <SOAPBindingName>SOAPNotificationBinding</SOAPBindingName>
            <SOAPOperationName>parameterUpdate</SOAPOperationName>
        </Action>
        <Sponsor>ACMEProvider</Sponsor>
        <Sponsor>XInc</Sponsor>
    </SupportingParty>
</Parties>
```

- **a definition of the services and their operations**. For each operation or group of operations, this section specifies the relevant service level parameters and respective metrics.[4] This section also indicates which party is responsible for measuring each metric and how the measuring should be done. The snippet below shows a service named `DemoService` with an operation called `GetQuote`. A parameter named `Availability_UpTimeRatio` is defined using a metric called `UpTimeRatio`, which in turn uses another metric `StatusTimeSeries`:

```
<ServiceDefinition name="DemoService">
    . . .
    <Operation name="GetQuote" xsi:type="WSDLSOAPOperationDescriptionType">
        <SLAParameter name="Availability_UpTimeRatio" type="float" unit="">
            <Metric>UpTimeRatio</Metric>
            <Communication>
                <Source>YMeasurement</Source>
                <Push>ZAuditing</Push>
            </Communication>
        </SLAParameter>
        . . .
        <Metric name="UpTimeRatio" type="long" unit="">
            <Source>YMeasurement</Source>
            <Function xsi:type="Minus" resultType="double">
                <Operand>
                    . . .
                            <Function xsi:type="ValueOccurs"
resultType="long">
                                <Metric>StatusTimeSeries</Metric>
                                . . .
                            </Function>
                            . . .
                </Operand>
            </Function>
        </Metric>
        <Metric name="StatusTimeSeries" type="TS" unit="">
            . . .
        </Metric>
        . . .
```

- **obligations of the agreement**, which may include service level objectives specified as predicates over service level parameters (e.g., "average response time for operation `getQuote` shall be less than two seconds between 9:00 a.m. and 5:00 p.m."), and **action guarantees**, which are the actions the parties commit to perform in a given situation (typically when a service level objective has been violated).

  In the previous example, service level parameters were declared separately for each operation or group of operations in a service. Therefore, the granularity to specify service level objectives varies from one operation to the entire service (i.e., from one operation to the group of all operations in the service). One cannot specify a single objective that applies to two services or the entire service layer. The following WSLA SLA snippet shows a service level objective named

[4] In this context, a service level metric is different from a service level parameter. A *metric* is a basic measure that can be obtained through instrumentation. A *parameter* is a quantifiable measure that is used in the SLA to specify the service level commitment between service providers and service users. In some cases, a parameter maps one to one to a metric, but in other cases a parameter is calculated based on a set of metrics. In other words, metrics are more network and technology oriented, whereas service parameters focus on user-perceivable effects.

UpTimeSLO that specifies that the value of the Availability_UpTimeRatio parameter shall be greater than 97% from 11/30/2008 to 12/31/2008. There is also an action guarantee that requires third-party ZAuditing to send a notification when objective UpTimeSLO is not met.

```
<Obligations>
    <ServiceLevelObjective name="UpTimeSLO">
        <Obliged>ACMEProvider</Obliged>
        <Validity>
            <Start>2008-11-30T14:00:00.000-05:00</Start>
            <End>2008-12-31T14:00:00.000-05:00</End>
        </Validity>
        <Expression>
            <Predicate xsi:type="Greater">
                <SLAParameter>Availability_UpTimeRatio</SLAParameter>
                <Value>0.97</Value>
            </Predicate>
        </Expression>
        <EvaluationEvent>NewValue</EvaluationEvent>
    </ServiceLevelObjective>
    . . .
    <ActionGuarantee name="UpTimeNotificationGuarantee">
        <Obliged>ZAuditing</Obliged>
        <Expression>
            <Predicate xsi:type="Violation">
                <ServiceLevelObjective>UpTimeSLO</ServiceLevelObjective>
            </Predicate>
        </Expression>
        <EvaluationEvent>NewValue</EvaluationEvent>
        <QualifiedAction>
            <Party>XInc</Party>
            <Action actionName="notification" xsi:type="Notification">
                <NotificationType>Violation</NotificationType>
                <CausingGuarantee>UpTimeNotificationGuarantee
3</CausingGuarantee>
                <SLAParameter>Availability_UpTimeRatio</SLAParameter>
            </Action>
        </QualifiedAction>
        <ExecutionModality>Always</ExecutionModality>
    </ActionGuarantee>
    . . .
```

- **Other information**, such as the price charged for using the services and a description of any penalties imposed when the service level objectives are not met, may also be included.

3.2.2 The WS-Agreement

The WS-Agreement specification, developed by the OGF, defines an XML-based language for agreements as well as a protocol for advertising the capabilities of service providers, creating agreements between service consumers and providers, and monitoring agreement compliance [Andrieux 2007]. Like the WSLA, the WS-Agreement provides an XML schema that defines the overall structure of an agreement document. In addition, the WS-Agreement specification defines a protocol for negotiating and establishing agreements dynamically based on web services (a set of WSDL definitions).

A key difference between the WS-Agreement and WSLA is that the structure of a WS-Agreement XML agreement is highly extensible. It contains several sections where intended users are expected to define domain-specific elements and properties.[5] Indeed, the WS-Agreement suggests the creation of agreement templates that embody all the customizable aspects of an agreement. Another difference between the WS-Agreement and WSLA is that the former does not provide a means to specify the metrics associated with parameters used in the agreement. Instead, metrics are defined in any structure required by a domain-specific extension.

Existing or new services that are the subject of a WS-Agreement SLA can be defined in the WSDL or any domain-specific service terms. A WS-Agreement SLA is an XML document that contains the following parts:

- mandatory unique ID for the agreement followed by an optional name
- context for the agreement (which includes the identification of the different parties), various metadata about the agreement (such as an expiration time and template ID in case the agreement was created following a template), and user-defined attributes
- the terms of the agreement, which are divided into service terms and guarantee terms. Extended WS-Agreements might involve other types of terms as well.
- service terms that identify and describe the services that are the subject of the agreement. The actual elements in the service description should be customized based on domain-specific needs. Service terms also comprise the measurable service properties and exposed properties that can be used to express service level objectives. They are equivalent to WSLA service level parameters. For each service property, the agreement may specify the variables that can be used in guarantee terms (see the next bullet). Examples of variables include configurable attributes, such as the number of CPUs available for service execution, and metrics, such as response time. The following is a snippet showing two service description terms from the same agreement—one specifying that 32 CPUs will be used to execute a job and the other specifying that 8 CPUS will be used:

```
. . .
<wsag:ServiceDescriptionTerm wsag:Name="numberOfCPUsHigh"
wsag:ServiceName="ComputeJob1">
    <job:numberOfCPUs>32</job:numberOfCPUs>
</wsag:ServiceDescriptionTerm>
<wsag:ServiceDescriptionTerm wsag:Name="numberOfCPUsLow"
wsag:ServiceName="ComputeJob1">
   <job:numberOfCPUs>8</job:numberOfCPUs>
</wsag:ServiceDescriptionTerm>
. . .
```

- guarantee terms that specify the levels of service quality that the parties are agreeing to. Examples of guarantee terms are the minimum availability of a service, the number and type of CPUs that the service provider should make available, and the average response time for requests. For each of these service level objectives, qualifying conditions may indicate the context for the objective to be enforced (e.g., only during weekdays). The syntax of a qualifying condition is not specified by the standard. Guarantees may be required from a service consumer in cases where the

[5] In many places, the WS-Agreement specification talks about domain-specific aspects of the agreement, but it could be customized to cover technology-specific aspects.

consumer has to provide a resource or input that a service provider depends on. Service level objectives are expressed in terms of service properties, or more specifically, as the variables defined for each service property. The scope of a guarantee term can be as narrow as a single operation of a web service or as broad as all operations of multiple services. The following snippet shows two options regarding the number of CPUs allocated for service `ComputeJob1`. These options are based on two service description terms (SDTs) previously defined in the SLA (see the previous code snippet). The business value,[6] indicated as 0.8 utility for 32 CPUs and 0.5 utility for 8 CPUs, can be used in the dynamic negotiation and renegotiation of service levels.

```
. . .
<wsag:GuaranteeTerm wsag:Name="ConfigurationPreference"
wsag:Obligated="ServiceProvider">
    <wsag:ServiceScope>
        <wsag:ServiceName>ComputeJob1</wsag:ServiceName>
    </wsag:ServiceScope>
    <wsag:ServiceLevelObjective xsi:type="sdtc:OpType">
        <Or>
            <SDT>numberOfCPUsHigh</SDT>
            <SDT>numberOfCPUsLow</SDT>
        </Or>
    </wsag:ServiceLevelObjective>
    <wsag:BusinessValueList>
        <wsag:Preference>

<wsag:ServiceTermReference>numberOfCPUsHigh</wsag:ServiceTermReference>
            <wsag:Utility>0.8</wsag:Utility>

<wsag:ServiceTermReference>numberOfCPUsLow</wsag:ServiceTermReference>
            <wsag:Utility>0.5</wsag:Utility>
        </wsag:Preference>
    </wsag:BusinessValueList>
</wsag:GuaranteeTerm>
```

3.3 Monitoring and Managing SLAs

Figure 1 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 1 also shows measurement subsystem components on the service user and service provider sides, but there are 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 1 illustrates, the values of the SLA parameters are input for the evaluation procedure, which can run on

- either the service user or service provider

[6] The definition of utility associated with the number of CPUs as well as the syntax for the corresponding service-level-objective section of the agreement are outside the scope of the WS-Agreement specification. The service provider and consumer have to agree on the understanding of this objective and define the syntax for expressing it in XML.

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

The evaluation procedure checks the values against the guaranteed conditions (i.e., the service level objectives) 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.

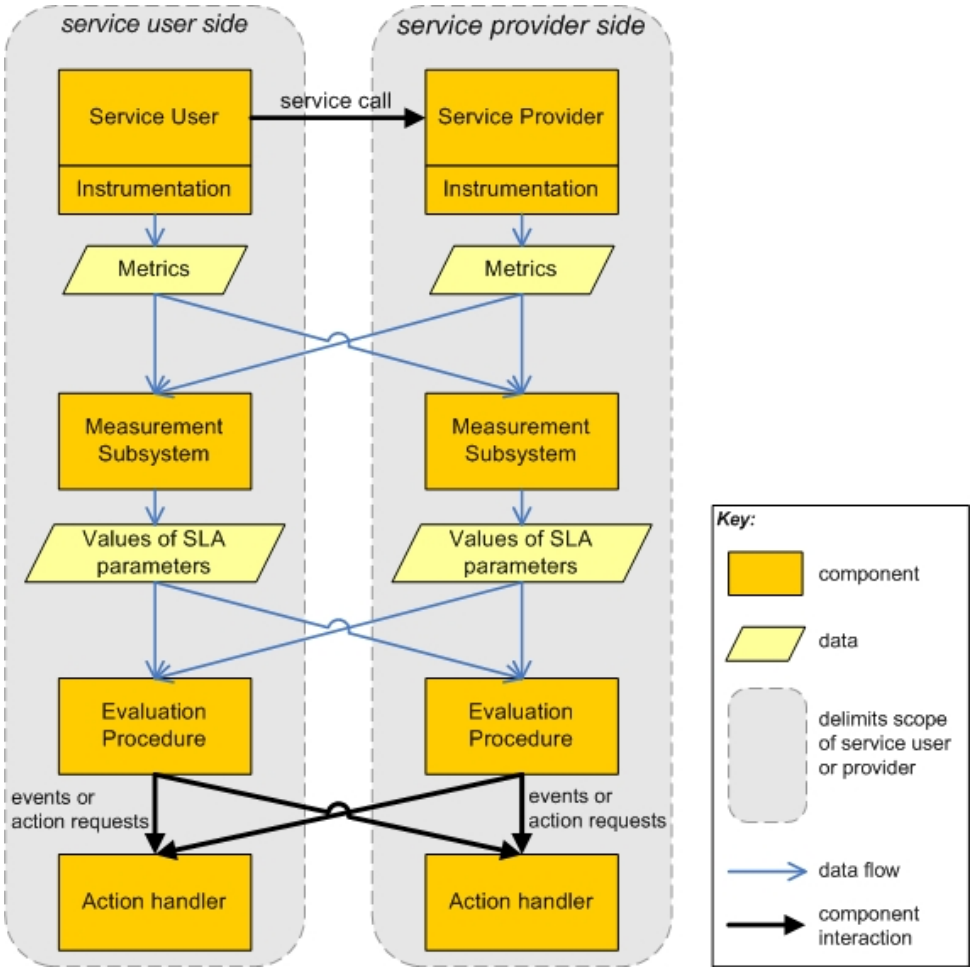

Figure 1: Conceptual Architecture for SLA Monitoring and Management (adapted from the work of Ludwig and colleagues [Ludwig 2003])

An important output of SLA management not shown in Figure 1 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.

3.4 SLA Life Cycle

A good example of an SLA life cycle is defined by the TM Forum in its *SLA Handbook Solution Suite V2.0* [TM Forum 2008]. Even though the audience for that handbook is the telecommunications industry, the identified six phases apply to services in general. An adaptation of this life cycle that applies to services in a SOA context is described below and presented in Figure 2.

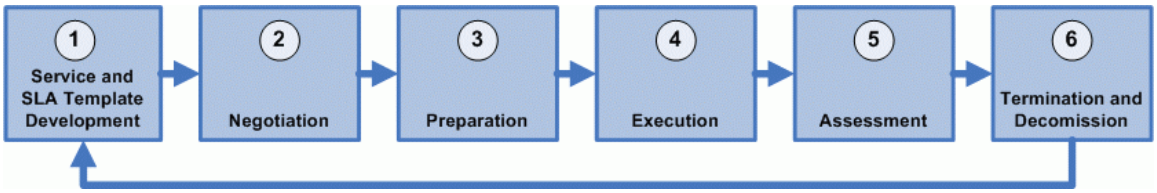

Figure 2: SLA Life Cycle

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. Hasselmeyer and colleagues discuss whether this phase is part of the life cycle or a predecessor to the life cycle [Hasselmeyer 2007]. They see it as it as a predecessor and not as part of the life cycle.
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:
   - 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.
   - 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 the decommission of discontinued services.

An active area of research is the automation of the SLA life cycle to enable the dynamic provisioning of services between organizations [Dan 2004, Paschke 2006b, Trzec 2003, Yuanming 2008]. In this scenario, all the phases outlined in Figure 2 occur at runtime. The next section will present the state of the practice of SLAs in SOA environments, including standards, tools, and modeling, management and monitoring practices, and all the key factors for automating the SLA life cycle.

4 Guidelines for SLAs in SOA Environments

Historically, service providers have created SLAs that specify key performance indicators (KPIs) associated with the communication infrastructure. More recently, the management of services, especially in the context of web services and grid services, has led to new indicators that focus on service quality rather than network performance [TM Forum 2008]. Because of the potential for automating the monitoring and management of SLAs as part of the SOA infrastructure, as indicated in Section 3.3, an active area of work is defining machine-readable SLAs. Whether SLAs are defined in plain-text or machine-readable form, what follows are some guidelines for developing and managing SLAs in SOA environments.

4.1 SLA Parameters and Metrics

Metrics are used in process control, software process improvement, business strategy implementation, and basically any field where data has to be collected in order to verify whether goals are being met. Literature in these fields indicate that creating metrics is a difficult task, which is why it commonly contains guidelines for creating good metrics.

SLA parameters are specified by a set of metrics. These metrics determine the measures that need to be gathered in order to verify whether the SLA parameters are being met. The following guidelines are proposed to define SLA parameters in a way that facilitates SLA management. Given that SLAs are mutually defined and negotiated, these guidelines apply to both providers and consumers. The parameters should be

- **reasonable.** They should motivate involved parties to act in a manner that is mutually beneficial. For example, an SLA parameter that specifies payment based on the number of services invoked may lead to poor service design in the interest of maximizing profit.
- **attainable.** Metrics that are beyond the control of either party should not be included. For example, because of the unpredictable nature of internet communication, it doesn't make sense to include metrics for total latency of a distributed transaction that runs across the internet.
- **enforceable.** This quality is especially important for the service consumer. SLAs should describe and contain evidence of how SLA parameters are monitored and enforced by the service provider. A comprehensive SLA monitoring solution should comprehend instrumentation and measurement both on the service consumer and service provider side. For instance, service consumers cannot expect the service provider to notify them when an SLA violation has occurred.
- **measurable.** Service metrics should be quantifiable and allow for repeatable measurements. Measurements should not be too difficult or too costly to collect. For example, if collecting the metrics uses significant computing resources, it may not be worth the effort.
- **objective.** Any SLA parameter that is open to more than one interpretation probably needs to be revised. For example, an SLA parameter might say that the service provider will notify service consumers when transactions are posted but say nothing about the timing of those notifications. Such a parameter could lead to different interpretations of acceptable latency.

Also, it may be useful to define higher level metrics that are related to SLA parameters. For example, collecting performance data from a cluster of servers can provide useful data for load balancing and resource management.

4.2 Machine-Readable SLAs

Unless an organization is interested in automated SLA management, has service consumers interested in the automated discovery of services based on attributes such as QoS, or has advanced registry query capabilities based on service quality, there is no need to create a machine-readable version of an SLA in addition to a plain-text version. However, having a machine-readable version of the SLA available might make a service more attractive for service consumers and also help the organization prepare for the time when tools and products can make use of it. If this is the case, a good practice is to follow a standard format, such as the WSLA or WS-Agreement (see Section 3.2), to benefit from the experience of others who have used it.

4.3 Negotiation and Flexibility in SLAs

As indicated in Section 3.4, negotiation is a major step in the SLA life cycle. To facilitate the negotiation process, you should

- involve a diverse set of stakeholders so that all business attributes from product implementation to marketing and finance are adequately addressed [TM Forum 2008]
- allow modifications to the SLA because the quality requirements of the service consumer may change over time
- if necessary, allow for multiple service levels so that the same service can be offered with different guaranteed quality properties to different service consumers
- create a decomposable SLA format so that SLAs can be put together according to service consumer quality concerns. For example, a service consumer concerned with the nondisclosure of personally identifiable information might not be concerned with high throughput as long as privacy is maintained. Another advantage of decomposability is that portions of an SLA that contain, for example, configuration parameters, can be given to a third party that is responsible for SLA monitoring without having to disclose the entire SLA.

4.4 SLAs and Governance

It is important to not confuse SOA governance with SLAs. SOA governance defines the set of policies, rules, and enforcement mechanisms for developing, using, and evolving service-oriented systems and for analyzing their business value. SOA governance typically includes policies, procedures, roles, and responsibilities for design-time governance and runtime governance. It is common for vendors to define SOA governance according to what their tools and products can measure and monitor. Essentially, these tools and product capabilities enable the definition and monitoring of qualities that could be part of SLAs, but they do not cover other aspects of SOA governance, such as which services should be created, how they should be created, who should have access to them, and how will they be provisioned.

5 Areas That Need More Research

The standardization and development of best practices for defining and managing SLAs in SOA environments is an active area of research. However, given the interest in dynamic discovery and selection of services, the emergence of third-party services, and the desire for machine-readable SLAs with few or no humans in the loop, there are still open areas in need of research:

- A recognized standard for SLAs is needed for specification and management. SLAs need to be machine readable to make the third generation of service-oriented systems possible—dynamic discovery, adaptation, and composition of services. An important contribution would be the creation of a generic and standardized framework that could provide the appropriate automation and support for (1) mapping contractual SLAs to standard and actionable implementations and (2) the monitoring and management of SLAs at runtime.
- A big issue in SLAs and service-oriented systems in general is that of trust. How does a service user trust a data source not in its control [Dobson 2005]? For example, if an SLA parameter deals with response time and there is a delay, how does the service consumer know whether the problem lies with the service or the network? In this case, the measures related to the service can only be collected by the service provider (controller), which means that the service consumer has to trust the data given by the service provider.
- More research is needed to understand and determine the QoS of composite services. For example, if the performance measure for a set of services is known, how can the performance for a composite service that used this set of services be determined?

6 Conclusions

SLAs have been used for many years in the telecommunications industry and in service organizations in general, including IT organizations, to formalize the level of service between a service provider and service consumer. While SLAs are well understood in these domains, they are less understood for services in the SOA context.

SOA enables the integration of automated services from multiple organizations. External providers may offer services that were not initially implemented to meet the quality attribute requirements of the service consumer organization. Defining an SLA and establishing SLA management mechanisms are important factors when clarifying the quality requirements for achieving the business and mission goals of SOA systems.

Machine-readable, standardized SLAs are going to be an important element for organizations moving to third-generation, service-oriented systems characterized by the dynamic discovery, composition, and invocation of services based on QoS and other contextual information. There are ongoing efforts to codify and standardize SLAs for web services, in an effort to make SLAs machine readable. However, there is no established standard for SLA specification. Current tools and products for SLA management and specification mostly use their own proprietary formats. While this is indeed a step toward the automated negotiation and management of SLAs between service providers and consumers, additional work in this area is needed to apply those tools in production environments.

7 Acronym List

CPU
central processing unit

ETTK
Emerging Technologies Toolkit

HTTP
Hypertext Transfer Protocol

IT
information technology

ITSM
Information Technology Service Management

ITIL
Information Technology Infrastructure Library

KPI
key performance indicator

MOF
Meta-Object Facility

MLS
Multiple Listing Service

OASIS
Organization for the Advancement of Structure Information Standards

OCL
Object Constraint Language

OGF
Open Grid Forum

QoS
quality of service

RBSLA
rule-based service level agreement

S3
Simple Storage Service

SDT
service description term

SLA
service level agreement

SLM
service level management

SNAP
Service Negotiation and Acquisition Protocol

SOA
service-oriented architecture

SP
service provider

TAPAS
Trusted and Quality of Service Aware Provision of Application Services

TM
TeleManagement

UCL
University College London

W3C
World Wide Web Consortium

WSDL
Web Services Description Language

WSDM
Web Services Distributed Management

WSLA
Web Service Level Agreement

WSML
Web Service Management Layer

WSMO
Web Service Modeling Ontology

WSOL
Web Services Offering Language

WSTK
Web Services Toolkit

XML
Extensible Markup Language

References

[Allen 2006]
Allen, Paul. *Service Level Agreements.* http://www.cbdiforum.com/report_summary.php3?page=/secure/interact/2006-12/service_level_agreements.php&area=silver (2006).

[Amazon 2008]
Amazon Web Services. *Amazon S3 Service Level Agreement.* http://aws.amazon.com/s3-sla/ (2008).

[Andrieux 2007]
Andrieux, Alain, et al. *Web Services Agreement Specification (WS-Agreement).* http://www.ogf.org/documents/GFD.107.pdf (2007).

[Apache 2008]
Apache Software Foundation. *Welcome to Apache Neethi.* http://ws.apache.org/commons/neethi/index.html (2008).

[BBC 2008]
BBC News. *Severed Cables Disrupt Internet*. http://news.bbc.co.uk/1/hi/technology/7218008.stm (2008).

[Brownsword 2004]
Brownsword, Lisa L.; Carney, David J.; Fisher, David; Lewis, Grace; Meyers, Craig; Morris, Edwin J.; Place, Patrick R. H.; Smith, James; & Wrage, Lutz. *Current Perspectives on Interoperability* (CMU/SEI-2004-TR-009, ADA 443493). Software Engineering Institute, Carnegie Mellon University, 2004. http://www.sei.cmu.edu/pub/documents/04.reports/pdf/04tr009.pdf

[Chatterjee 2005]
Chatterjee, Abhishek Malay; Chaudhari, Anshuk Pal; Das, Akash Saurav; Dias, Terance; & Erradi, Abdelkarim. *Differential QoS Support in Web Services Management: One Service Implementation - Many Levels of Service.* http://webservices.sys-con.com/read/121946_3.htm (2005).

[Czajkowski 2002]
Czajkowski, K.; Foster, I.; Kesselman, C.; Sander, V.; & Tuecke, S. "SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems," 153-183. *Job Scheduling Strategies for Parallel Processing, 8th International Workshop, JSSPP 2002. Revised Papers.* (Lecture Notes in Computer Science Volume 2537), 2002.

[Dan 2004]
Dan, A.; Davis, D.; Kearney, R.; Keller, A.; King, R.; Kuebler, D.; Ludwig, H.; Polan, M.; Spreitzer, M.; & Youssef, A. *Web Services on Demand: WSLA-Driven Automated Management*. http://www.research.ibm.com/journal/sj/431/dan.html (2004).

[de Bruijn 2006]
de Bruijn, Jos; et. al. *D2v1.3. Web Service Modeling Ontology (WSMO).* http://www.wsmo.org/TR/d2/v1.3/ (2006).

[Dobson 2005]
Dobson, Glen; Lock, Russell; & Sommerville, Ian. *QoSOnt: A QoS Ontology for Service-Centric Systems.* http://ieeexplore.ieee.org/iel5/10177/32500/01517730.pdf?arnumber=1517730 (2005).

[Hasselmeyer 2007]
Hasselmeyer, Peer; Koller, Bastian; Kotsiopoulos, Ioannis; Kuo, Dean; & Parkin, Michael. *Negotiating SLAs with Dynamic Pricing Policies.* http://www.hasselmeyer.com/pdf/socinside07.pdf (2007).

[IST 2006]
Information Society Technologies (IST). Fitzgerald, B., group chairman. Olsson C. M., ed. *The Software and Services Challenge*. ftp://ftp.cordis.europa.eu/pub/ist/docs/directorate_d/st-ds/fp7-report_en.pdf (2006).

[Ludwig 2003]
Ludwig, Heiko; Keller, Alexander; Dan, Asit; King, Richard; & Franck, Richard. *Web Service Level Agreement (WSLA) Language Specification, Version 1.0*. http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf (2003).

[Ludwig 2006]
Ludwig, André; Braun, Peter; Kowalczyk, Ryszard; & Franczyk, Bogdan. "A Framework for Automated Negotiation of Service Level Agreements in Services Grids,"89-101. *Business Process Management Workshops* (Lecture Notes in Computer Science 3812). Nancy, France, September 2006. Springer, 2006.

[OASIS 2006]
Organization for the Advancement of Structured Information Standards (OASIS). *OASIS Web Services Distributed Management (WSDM) TC*. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsdm (2006).

[OASIS 2007a]
Organization for the Advancement of Structured Information Standards (OASIS). *WS-SecurityPolicy 1.2*. http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702 (2007).

[OASIS 2007b]
Organization for the Advancement of Structured Information Standards (OASIS). *Web Services Reliable Messaging Policy 2 Assertion (WS-RM Policy) Version 1.1.* http://docs.oasis-open.org/ws-rx/wsrmp/200702/wsrmp-1.1-spec-os-01.pdf (2007).

[OASIS 2007c]
Organization for the Advancement of Structured Information Standards (OASIS). *Web Services Atomic Transaction (WS-AtomicTransaction) Version 1.1.* http://docs.oasis-open.org/ws-tx/wstx-wsat-1.1-spec-os/wstx-wsat-1.1-spec-os.html (2007).

[OASIS 2007d]
Organization for the Advancement of Structured Information Standards (OASIS). *Web Services Business Activity (WS-BusinessActivity) Version 1.1*. http://docs.oasis-open.org/ws-tx/wstx-wsba-1.1-spec-os/wstx-wsba-1.1-spec-os.html (2007).

[Paschke 2006a]
Paschke, Adrian. *RBSLA: Rule-Based Service Level Agreements*. http://ibis.in.tum.de/projects/rbsla/index.php (2006).

[Paschke 2006b]
Paschke, A. & Schnappinger-Gerull, E. *A Categorization Scheme for SLA Metrics*. http://ibis.in.tum.de/staff/paschke/docs/MKWI2006_SLA_Paschke.pdf (2006).

[Seidel 2007]
Seidel, Jan; Wäldrich, Oliver; Ziegler, Wolfgang; Wieder, Philipp; & Yahyapour, Ramin. *Using SLA for Resource Management and Scheduling: A Survey*. http://www.coregrid.net/mambo/images/stories/TechnicalReports/tr-0096.pdf (2007).

[SSEL 2003]
System and Software Engineering Lab (SSEL). *Web Services Management Layer Homepage*. http://ssel.vub.ac.be/wsml/ (2006).

[TAPAS 2005]
Trusted and Quality of Service Aware Provision of Application Services (TAPAS). *Overview.* http://www.cs.ucl.ac.uk/staff/J.Skene/slang/ (2005).

[TM Forum 2008]
Telemanagement (TM) Forum. *SLA Handbook Solution Suite V2.0*. http://www.tmforum.org/DocumentLibrary/SLAHandbookSolution/2035/Home.html (2008).

[Tosic 2002]
Tosic, Vladimir; Patel, Kruti; & Pagurek, Bernard. *WSOL — Web Service Offerings Language*. http://www.springerlink.com/content/pt03ur5fx9w5weda/fulltext.pdf (2002).

[Trzec 2003]
Trzec, Krunoslav & Huljenic, Darko. *Modeling Agent-Based Framework for the Automation of SLA Management Lifecycle*. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=01214627 (2003).

[W3C 2006]
World Wide Web Consortium (W3C). *Web Services Policy 1.2 - Attachment (WS-PolicyAttachment)*. http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/ (2006).

[W3C 2007a]
World Wide Web Consortium (W3C). *Web Services Policy 1.5 – Framework.* http://www.w3.org/TR/ws-policy/ (2007).

[W3C 2007b]
World Wide Web Consortium (W3C). *Web Services Addressing 1.0 – Metadata.* http://www.w3.org/TR/2007/PR-ws-addr-metadata-20070731/ (2007).

[Wohlstadter 2004]
Wohlstadter, Eric; Tai, Stefan; Mikalsen, Thomas; Rouvellou, Isabelle; & Devanbu, Premkumar. *GlueQoS: Middleware to Sweeten Quality-of-Service Policy Interactions*. http://ieeexplore.ieee.org/iel5/9201/29176/01317441.pdf?arnumber=1317441 (2004).

[Yuanming 2008]
Yuanming, Cao; Wendong, Wang; Xiangyang, Gong; & Xirong, Que. *Initiator-Domain-Based SLA Negotiation for Inter-Domain QoS-Service Provisioning*. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=04476553 (2008).

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	September 2008	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Service Level Agreements in Service-Oriented Architecture Environments	FA8721-05-C-0003

6. AUTHOR(S)

Philip Bianco, Grace A. Lewis, Paulo Merson

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2008-TN-021

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

Quality attribute requirements play an important role in service selection in a service-oriented architecture environment. It is easy to envision finding several services that fulfill the functional requirements but fail to meet important quality attribute measures. Service level agreements provide the architect with a specification of the verifiable quality characteristics that the service will provide. Such a specification allows the architect to select the service that best supports the system's quality attribute requirements. This report surveys the state of practice in service-level-agreement specification and offers guidelines on how to assure that services are provided with high availability, security, performance, and other required qualities. In addition, this report discusses the quality properties that have been expressed in service level agreements, and it considers the mechanisms used to assure service quality by contract and those used by service providers to achieve and monitor quality properties. Also, this report examines the support for expressing service quality requirements in existing service technologies and describes areas where more research is needed.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Service-Level Agreements, SLAs, Service-Oriented Architecture, SOA, web service, quality of service, QoS	40

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL