USENIX Association

Proceedings of LISA 2002: 16th Systems Administration Conference

Philadelphia, Pennsylvania, USA
November 3–8, 2002

Pp. 189-204 of the *Proceedings of LISA '02: Sixteenth Systems Administration Conference* (Berkeley, CA: USENIX Association, 2002).

Defining and Monitoring Service Level Agreements for Dynamic e-Business

Alexander Keller and Heiko Ludwig – IBM T. J. Watson Research Center

ABSTRACT

Fueled by the growing acceptance of the Web Services Architecture, an emerging trend in application service delivery is to move away from tightly coupled systems towards structures of loosely coupled, dynamically bound systems to support both long and short business relationships. It appears highly likely that the next generation of e-Business systems will consist of an interconnection of services, each provided by a possibly different service provider, that are put together on an ''on demand'' basis to offer an end to end service to a customer.

Such an environment, which we call Dynamic e-Business (DeB), will be administered and managed according to dynamically negotiated Service Level Agreements (SLA) between service providers and customers. Consequently, system administration will increasingly become SLA-driven and needs to address challenges such as dynamically determining whether enough spare capacity is available to accommodate additional SLAs, the negotiation of SLA terms and conditions, the continuous monitoring of a multitude of agreed-upon SLA parameters and the troubleshooting of systems, based on their importance for achieving business objectives.

A key prerequisite for meeting these goals is to understand the relationship between the cost of the systems an administrator is responsible for and the revenue they are able to generate, i.e., a model needs to be in place to express system resources in financial terms. Today, this is usually not the case.

In order to address some of these problems, this paper presents the *Web Service Level Agreement (WSLA)* framework for defining and monitoring SLAs in inter-domain environments. The framework consists of a flexible and extensible language based on the XML schema and a runtime architecture based on several SLA monitoring services, which may be outsourced to third parties to ensure a maximum of accuracy.

WSLA enables service customers and providers to unambiguously define a wide variety of SLAs, specify the SLA parameters and the way how they are measured, and tie them to managed resource instrumentations. A Java-based implementation of this framework, termed *SLA Compliance Monitor*, is publicly available as part of the IBM Web Services Toolkit.

Introduction and Motivation

The pervasiveness of the Internet provides a platform for businesses to offer and buy electronic services, such as financial information, hosted services, or even applications, that can be integrated in a customer's application architecture. Upcoming standards for the description and advertisement of, as well as the interaction with, online services promise that organizations can integrate their systems in a seamless manner. The Web Services framework [16] provides such an integration platform, based on the WSDL service interface description language, the UDDI directory service [31] and, for example, SOAP over HTTP as a communication mechanism. Web Services provide the opportunity to dynamically bind to services at runtime, i.e., to enter (and dismiss) a business relationship with a service provider on a case-by-case basis, thus creating an infrastructure for *dynamic e-Business* [14].

Dynamic e-Business implies dynamics several orders of magnitude higher than found in traditional corporate networks. Moreover, a service relationship also constitutes a business relationship between independent organizations, defined in a contract.

An important aspect of a contract for IT services is the set of Quality of Service (QoS) guarantees a service provider gives. This is commonly referred to as a service level agreement (SLA) [32, 17]. Today, SLAs between organizations are used in all areas of IT services – in many cases for hosting and communication services but also for help desks and problem resolution.

Furthermore, the IT parameters for which Service Level Objectives (SLO) are defined come from a variety of disciplines, such as business process management, service and application management, and traditional systems and network management. In addition, different organizations have different definitions for crucial IT parameters such as Availability, Throughput, Downtime, Bandwidth, Response Time, etc. Today's SLAs are plain natural language documents. Consequently, they must be manually provisioned and monitored, which is very expensive and slow.

The definition, negotiation, deployment, monitoring and enforcement of SLAs must become – in contrast to today's state of the art – an automated process. This poses several challenges for the administration of shared distributed systems, as found in Internet Data Centers, because administrative tasks become increasingly dynamic and SLA-driven.

The objective of this paper is to present the *Web Service Level Agreement (WSLA)* framework as an approach to deal with these problems; it provides a flexible, formal language and a set of elementary services for defining and monitoring SLAs in dynamic e-Business environments.

The paper is structured as follows: In the next section, we describe the underlying principles of our work, analyze the requirements of dynamic e-Business on system administration tasks and on the WSLA framework. We also describe the relationships of our work to the existing state of the art. The WSLA runtime architecture, described later, provides mechanisms for accessing resource metrics of managed systems and for defining, monitoring and evaluating SLA parameters according to an SLA specification. We subsequently introduce the WSLA language by means of several examples. It is based on the XML Schema and allows parties to define QoS guarantees for electronic services and the processes for monitoring them. Finally, the last section concludes the paper and gives an overview of our current work.

Principles of the WSLA Framework

Service level management has been the subject of intense research for several years now and has reached a certain degree of maturity. However, despite initial work in the field (see, e.g., [2]), the problem of establishing a generic framework for service level management in cross-organizational environments remains unsolved yet. In this section, we introduce the terminology and describe the fundamental principles, which will be used throughout this paper. Subsequently, focusing on SLA-driven system administration, we derive the requirements of the WSLA language and its runtime architecture.

Terminology

There are various degrees to which extent a service customer is willing to accept the parameters offered by the service provider. Metric- and SLA-related information appears at various tiers of a distributed system, as depicted in Figure 1.

- **Resource Metrics** are retrieved directly from the managed resources residing in the service provider's tier, such as routers, servers, middleware and instrumented applications. Typical examples are the well-known MIB variables of the IETF Structure of Management Information (SMI) [21], such as counters and gauges.
- **Composite Metrics** 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%, minimum, maximum etc.). This is usually being done within the service providers' domain but can be outsourced to a third-party measurement service as well. Composite metrics are exposed by a service provider by means of a well-defined (usually HTTP or SOAP based) interface for further processing.

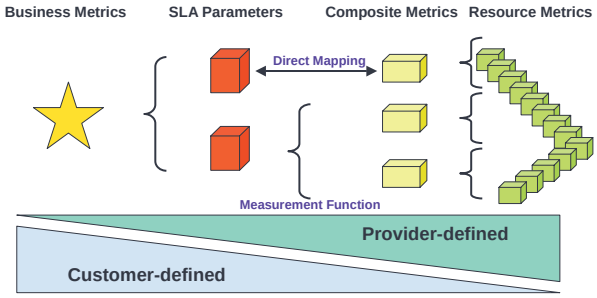

Figure 1: Aggregating business metrics, SLA parameters and metrics across different organizations.

- **SLA Parameters** put the metrics available from a service provider into the context of a specific customer and are therefore the core part of an SLA. In contrast to the previous metrics, every SLA parameter is associated with high/low watermarks, which enables the customer, provider, or a designated third party to evaluate the retrieved metrics whether they meet/exceed/fall below defined service level objectives. Consequently, every SLA parameter and its permitted range are defined in the SLA. It makes sense to delegate the evaluation of SLA parameters against the SLOs to an independent third party; this ensures that the evaluation is objective and accurate.
- **Business Metrics** relate SLA parameters to financial terms specific to a service customer (and thus are usually kept confidential by him). They form the basis of a customer's risk management strategy and exist only within the service customer's domain. It should be noted that a service provider needs to perform a similar mapping to make sure the SLAs he is willing to satisfy are in accordance with his business goals.

The WSLA framework presented in this paper is capable of handling all four different parameter types; apart from the latter, they relate directly to systems management tasks and are our main focus. However, the flexible mechanism for composing SLAs can be easily extended to accommodate business metrics.

Scenarios for SLA Establishment

Often, it is not obvious to draw a line between the aforementioned parameter types, in particular

between Composite Metrics and SLA Parameters. Therefore, we assume that every parameter related to a customer and associated with a guaranteed value range is considered an SLA parameter, which is supposed to be part of an SLA. However, this distinction is also highly dependent on the extent a customer requires the customization of metrics exposed by the service provider (or a third-party measurement service) – and how much he is willing to pay for it. This, in turn, depends on the degree of customization the provider is willing to apply to its metrics. The following scenarios describe various scenarios how SLAs may be defined:

1. **A customer adopts the data exposed by a service provider without further refinement** This is often done when the metrics reflect good common practice, cannot be modified by the customer or are of small(er) importance to him. In this case, the selected metrics become the SLA parameters and thus integral parts of the SLA. Examples are: *length of maintenance intervals* or *backup frequency*.
2. **The customer requests that collected data is put into a meaningful context** A customer is probably not interested in the overall availability of a provider's data center, but needs to know the availability of the specific cluster within the data center on which his applications and data are hosted. A provider's data collection algorithm therefore needs – at least – to take into account for which customer the data is actually collected. A provider may decide to offer such preprocessed data, such as: *Availability of the server cluster hosting customer X's web application*.
3. **The customer requests customized data that is collected according to his specific requirements** While a solution to item 2 can still be reasonably static (changes tend to happen rarely and the nature of the modifiable parameters can be anticipated reasonably well), the degree of choice for the customer can be taken a step further by allowing him to specify arbitrary parameters, e.g., the input parameters of a data collection algorithm.

   This implies that a service provider needs to have a mechanism in place that allows a customer to provide these input parameters – preferably at runtime, e.g., *The average load of a server hosting the customer's website should be sampled every 30 seconds and collected over 24 hours*. Note that a change of these parameters may result in a change of the terms and conditions of an SLA, e.g., when a customer chooses sampling intervals that are likely to impact the performance of the monitored system; eventually, this may entail the violation of SLAs the service provider has with other customers.
4. **The customer specifies how data is collected** This means that he defines – in addition to the metrics and input parameters – the data collection algorithm itself. This is obviously the most extreme case and seems fairly unlikely. However, large customers may insist of getting access to very specific data that is not part of the standard set, e.g., a customer may want to know which employees of a service provider had physical access to the systems hosting his data and would like to receive a daily log of the badge reader.

   This means that – in addition to the aforementioned extension mechanisms – a service provider needs to have a mechanism in place that allows him to introduce new data collection mechanisms without interrupting his management and production systems.

While the last case poses the highest challenge on the programmability of the monitoring system, a service provider benefits greatly from a management system being capable of handling such flexible SLAs because all the former situations are special cases of the latter. It also addresses the extreme variability of today's SLAs. Sample SLAs we analyzed clearly indicate that there is a need for defining a mechanism that allows to unambiguously specify the data collection algorithm. Also, it should be noted that the different possibilities of specifying service level objectives are not mutually exclusive and may all be specified within the same SLA.

SLA-driven System Administration

Now that we have introduced the concepts of SLA management in a dynamic e-Business environment, we are able to derive its implications on systems administration and management. While it is clear that the very high dynamics of the establishment/dismissal of business relationships and the resulting allocation/deallocation of system resources to different users alone is a challenge on its own, we have found several other issues that are likely to impact how system administration is done in such an environment. The way we see the tasks of a system administrator evolve are described in the following subsections.

Express System Resources in Financial Terms

While system administrators usually have an awareness of the costs of the systems they are administering, the need to assign prices to the various resources on a very fine-grained basis will certainly increase. For quite some time, it has been common practice in well-run multi-customer data centers to account for CPU time, memory usage and disk space usage on a per-user basis. What will become increasingly important in SLA-driven system administration is the monitoring, accounting and billing of aggregated QoS parameters such as response time, throughput and bandwidth, which need to be collected across a variety of different systems that are involved in a multi-tiered server environment.

Having such a fine-grained accounting scheme in place is the prerequisite for defining SLOs, together with associated penalties or bonuses. In addition, the business impact of an outage or delay on the customer needs to be assessed. While the latter is mainly relevant to a service customer, a system administrator on the service provider side will need an even better understanding of the cost/benefit model behind the services offered to a customer. As a sidenote, the ability to offer measurement facilities for fine-grained service parameters is likely to become a distinguishing factor among service providers.

Involvement in SLA Negotiation

The technical expertise of a system administrator is likely to play an increasing role in an area that is currently confined to business managers and lawyers: The negotiation of SLAs terms. While current SLAs are dominated by legal terms and conditions, it will become necessary in an environment where resources are shared among different customers (under a variety of SLAs) to evaluate whether enough spare capacity is available to accommodate an additional SLA that asks for a specific amount of resources without running into the risk that the resources become overallocated if a customer's demand increases. While complex resource allocation schemes will probably not be deployed in the near future, an administrator nevertheless needs to have an understanding of the safety margins he must take into account when accepting new customers.

A related problem is to evaluate whether additional load due to SLA measurements is acceptable or not: While it may well be the case that enough capacity is available to accommodate the workload resulting from the service usage, overly aggressive SLA measurement algorithms may have a detrimental impact on the overall workload a system can handle. An extreme example for this is a customer whose application resides on a shared server and who would like to have the availability of the system being probed every few seconds. In this case, an SLA may either need to be rejected due to the additional workload, or the price for carrying out the measurements will need to be adjusted accordingly.

Classify Customers According to Revenue

The previous discussions make it clear that a service provider's approach to SLA-driven management entails the definition of enterprise policies that classify customers, e.g., according to the profit margins or their degree of contribution to a service provider's overall revenue stream. The involvement of system administrators in the process of policy definition and enforcement is a consequence of having both a high degree of technical understanding and insight into the business: First, this expertise is needed to determine which policies are reasonable and enforceable.

Second, once the policies are defined, it is up to the administrator to enforce them, e.g., if the resource capacity becomes insufficient because of increased workload of a high-paying customer, lower-paying customers may be starved out if the penalties associated with their SLAs can be offset by the increased gains from providing additional capacity to a higher-paying customer. Third, it should be noted that such a behavior adds an interesting twist to the problem determination schemes an administrator uses: The non-functioning of a customer's system may not necessarily be due to a technical failure, but may well be the consequence of a business decision.

Fix Outages According to Classification

The establishment of policies and the classification of customers also has implications on how system outages are addressed. Traditionally, system administrators are trained to address the most severe outages first. This may change if a customer classification scheme is in place, because then the system whose downtime or decreased level of service is the most expensive for the service provider will need to be fixed first. Outages are likely going to be classified not according to their technical severity, but rather based on their business impact.

Lessons Learned From Real-Life SLAs

A suitable SLA framework for Web Services must not constrain the parties in the way they formulate their clauses but instead allow for a high degree of flexibility. A management tool that implements only a non-modifiable textbook definition of availability would not be considered helpful by today's service providers and their customers.

Our studies of close to three dozen SLAs currently used throughout the industry in the areas of application service provisioning (ASP) [1], web hosting and information technology (IT) outsourcing have revealed that even if seemingly identical SLA parameters are being defined, their semantics vary greatly.

While some service providers confine their definition of ''application availability'' to the network level of the hosting system (''user(s) being able to establish a TCP connection to the appropriate server''), others refer to the application that implements the service (''Customer's ability to access the software application on the server''). Still others rely on the results obtained from monitoring tools (''the application is accessible if the server is responding to HTTP requests issued by a specific monitoring software''), while another approach uses elaborate formulas consisting of various metrics, which are sampled over fixed time intervals.

These base clauses are then usually annotated with exceptions, such as maintenance intervals, weekend/holiday schedules, or even the business impact of an outage (''An outage has been detected by the ASP but no material, detrimental impact on the customer has occurred as a result''). The latter example, in particular, illustrates the disconnect between the people

involved in the negotiation and establishment of an SLA (usually business managers and lawyers) and the ones who are supposed to enforce it (system administrators). One way of closing this gap is to enable system administrators to become involved in the negotiation of an SLA by providing them with a tool able to create a legal document, namely the SLA.

It is important to keep in mind that, while the nature of the clauses may differ considerably among different SLAs, the general structure of all the different SLAs remains the same: Every analyzed SLA contains

- the involved parties,
- the SLA parameters,
- the metrics used as input to compute the SLA parameters,
- the algorithms for computing the SLA parameters,
- the service level objectives and the appropriate actions to be taken if a violation of these SLOs has been detected.

This implies that there is a way to come up with an SLA language that can be applied to a multitude of bilateral customer/provider relationships.

WSLA Design Goals

In this section, we will derive – based on the above discussions – the requirements the WSLA framework needs to address.

Ability to Accommodate a Wide Variety of SLAs

In the introduction of this paper, we have stressed the point that SLAs, their parameters and the SLOs defined for them are extremely diverse. One approach to deal with this problem (e.g., as it is done today for simple consumer Web hosting services) is to narrow down the ''universe of discourse'' to a few well-understood terms and to limit the possibilities of choosing arbitrary QoS parameters through the use of SLA templates [24]. SLA templates include several automatically processed fields in an otherwise natural language-written SLA. However, the flexibility of this approach is limited and only suitable for a small set of variants of the same type of service using the same QoS parameters and a service offering that is not likely to undergo changes over time. In situations where service providers must address different SLA requirements of their customers, they need a more flexible formal language to express service level agreements and a runtime architecture comprising a set of services being able to interpret this language.

Leverage Work in the B2B Area for SLA Negotiation and Creation

Architectural components and language elements related to SLA negotiation, creation and deployment should leverage existing concepts developed in the electronic commerce and B2B area. In particular, the applicability of automated negotiation mechanisms, e.g., currently being developed within the scope of the OASIS/ebXML [6] Collaboration Profiles and Agreements initiative [7], should be applicable to the negotiation of SLAs as well. A vast amount of work on electronic contracts [25, 22], contract languages [12] and contract negotiation has been carried out in the electronic commerce and B2B arena [4]. We later describe our usage of obligations, a concept widely used in e-commerce, for monitoring SLAs.

Apply the "Need to Know" Principle to SLA Deployment

For each service provider and customer relationship, several instances of a service may exist. The functionality of computing SLA parameters or evaluating contract obligations may be split, e.g., among multiple measurement or SLO evaluation services, each provided by a different organization. It is therefore important that every service instance receives only the part of the contract it needs to know to carry out its task. Since it may be possible that a contractual party delegates the same task (such as measurements) to several different third party services (in order to be able to cross-check their results), different service instances may not be aware of other instances. This implies that every party involved in the SLA monitoring process receives only the part of the SLA that is relevant for him. We present our approach for dealing with this problem later.

Another major issue that underlines the importance of this ''Need to know'' principle are the privacy concerns of the various parties involved in an inter-domain management scenario: A service provider is, in general, neither interested in disclosing which of his business processes have been outsourced to other providers, nor the names of these providers. On the other hand, customers of a dynamic e-Business will not necessarily see a need anymore to know the exact reason of performance degradations as long as a service provider is able to take appropriate remedies (or compensate its customers for the incurred service level violation).

Traditionally, end-to-end performance management has been the goal of traditional enterprise management efforts and is often explicitly listed as a requirement (see, e.g., [26]). However, the aforementioned privacy concerns of service providers and the service customers' need for transparency make that an end-to-end view becomes unachievable (and irrelevant!) in a dynamic e-Business environment spanning multiple organizational domains.

Delegate Monitoring Tasks to Third Parties

Traditionally, an SLA is a bilateral agreement between a service customer and a service provider: The *enhanced Telecom Operations Map (eTOM)* [29], for example, defines various roles services providers can play; however, this work does not provide the delegation of management functionality to further service providers. We refer to the parties that establish and sign the SLA as **signatory parties**. In addition, SLA monitoring may require the involvement of third parties: They come into play when either a function needs

to be carried out that neither service provider nor customer wants to do, or if one signatory party does not trust the other to perform a function correctly. Third parties act then in a **supporting role** and are sponsored by either one or both signatory parties.

The targeted environment of our work is a typical service provider environment (Internet storefronts, B2B marketplaces, web hosting, ASP), which consists of multiple, independent parties that collaborate according to the terms and conditions specified in the SLA. Consequently, the services of our architecture are supposed to be distributed among the various parties and need to interact across organizational domains. Despite the focus on cross-organizational entities, WSLA can be applied to environments in which several (or even all of the) services reside within the boundaries of a single organizational domain, such as in a traditional corporate network.

The work of the IST Project FORM [8] is highly relevant for our work, since it focuses on SLAs in an inter-domain environment. FORM also deals with the important issue of federated accounting [3], which we do not address in this paper. An approach for a generic service model suitable for customer service management is presented in [9].

SLA-driven Resource Configuration

Since the terms and conditions of an SLA may entail setting configuration parameters on a potentially wide range of managed resources, an SLA management framework must accommodate the definition of SLAs that go beyond electronic/web services and relate to the supporting infrastructure. On the one hand, it needs to tie the SLA to the monitoring parameters exposed by the managed resources so that an SLA monitoring infrastructure is able to retrieve important metrics from the resources. [33] defines a MIB for SLA performance monitoring in an SNMP environment, whereas the SLA handbook from TeleManagement Forum [27] proposes guidelines for defining SLAs for telecom service providers.

An approach for the performance instrumentation of EJB-base distributed applications is described in [5]. The capability of mapping resource metrics to SLA parameters is crucial because a service provider must be able to answer the following questions before signing an SLA:

- Is it possible to accept an SLA for a specific service class given the fact that the capacity is limited?
- Can additional workload be accommodated?

On the other hand, it is desirable to derive configuration settings directly from SLAs. However, the heterogeneity and complexity of the management infrastructure makes configuration management a challenge. Successful work in this area often focuses on the network level: [10] describes a network configuration language; the Policy Core Information Model (PCIM) of the IETF [23] provides a generic framework for defining policies to facilitate configuration management.

Existing work in the e-commerce area may be applied here as well since the concept of contract-driven configuration in e-commerce environments [11] and virtual enterprises [20, 13] has similarities to the SLA-driven configuration of managed resources.

WSLA Runtime Architecture

In this section, we describe the WSLA runtime architecture by breaking it down into its atomic building blocks, namely the elementary services needed to enable the management of an SLA throughout the phases of its lifecycle. The first part describes the information flows and interactions between the different services. The next section demonstrates how the SLA management services identified earlier cooperate in an inter-domain environment, where the task of SLA management itself is dynamically delegated to an arbitrary number of management service providers.

WSLA Services and their Interactions

The components described in this section are designed to address the "need to know" principle and constitute the atomic building blocks of the WSLA monitoring framework. The components are intended to interact across multiple domains; however, it is possible that some components may be co-located within a single domain and not necessarily exposed to objects residing within another domain.

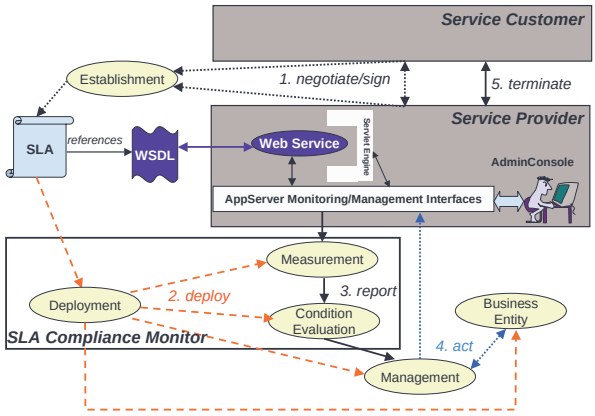

Figure 2: Interactions between the WSLA services.

Figure 2 gives an overview of the SLA management lifecycle, which consists of five distinct phases. We assume that an SLA is defined for a web service, which is running in the servlet engine of a web application server. The web application server exposes a variety of management information either through the graphical user interface of an administration console or at its monitoring and management interfaces, which are accessed by the various services of the WSLA framework.

The interface of the web service is defined by an XML document in the *Web Services Description*

Language (WSDL). The SLA references this WSDL document and extends the service definition with SLA management information. Typically, an SLA defines several SLA parameters, each referring to an operation of the web service. However, an SLA may also reference the service as a whole, or even compositions of multiple web services [30]. The phases and the services that implement the functionality needed during the various phases are as follows.

Phase 1: SLA Negotiation and Establishment

The SLA is being negotiated and signed by both signatory parties. This is done by means of an **SLA Establishment Service**, i.e., an SLA authoring tool that lets both signatory party establish, price and sign an SLA for a given service offering. This tool allows a customer to retrieve the metrics offered by a service provider, aggregate and combine them into various SLA parameters, request approval from both parties, define secondary parties and their tasks, and make the SLA document available for deployment to the involved parties (dotted arrows in Figure 2).

Phase 2: SLA Deployment

Deployment Service: The deployment service is responsible for checking the validity of the SLA and distributing it either in full or in appropriate parts to the involved components (dashed arrows in Figure 2). Since two signatory parties negotiate the SLA, they must inform the supporting parties about their respective roles and duties. Two issues must be addressed:

1. Signatory parties do not want to share the whole SLA with their supporting parties but restrict the information to the relevant information such that they can configure their components. Signatory parties must analyze the SLA and extract relevant information for each party. In the case of a measurement service, this is primarily the definition of SLA parameters and metrics. SLO evaluation services get the SLOs they need to verify. All parties need to know the definitions of the interfaces they must expose, as well as the interfaces of the partners they interact with.
2. Components of different parties cannot be assumed to be configurable in the same way, i.e., they may have heterogeneous configuration interfaces.

Thus, the deployment process contains two steps. In the first step, the SLA deployment system of a signatory party generates and sends configuration information in the *Service Deployment Information (SDI)* format (omitted for the sake of brevity), a subset of the language described later, to its supporting parties. In the second step, deployment systems of supporting parties configure their own implementations in a suitable way.

Phase 3: Measurement and Reporting

This phase deals with configuring the runtime system in order to meet one or a set of SLOs, and with carrying out the computation of SLA parameters by retrieving resource metrics from the managed resources and executing the management functions (solid arrows in Figure 2). The following services implement the functionality needed during this phase:

Measurement Service: The Measurement Service maintains information on the current system configuration, and run-time 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 outside the service provider's domain, e.g., by probing or intercepting client invocations. A Measurement Service may measure all or a subset of the SLA parameters. Multiple measurement services may simultaneously measure the same metrics.

Condition Evaluation Service: This service is responsible for comparing measured SLA parameters against the thresholds defined in the SLA and notifying the management system. It obtains measured values of SLA parameters from the Measurement Service and tests them against the guarantees given in the SLA. This can be done each time a new value is available, or periodically.

Phase 4: Corrective Management Actions

Once the Condition Evaluation Service has determined that an SLO has been violated, corrective management actions need to be carried out. The functionality that needs to be provided in this phase spans two different services:

Management Service: Upon receipt of a notification, the management service (usually implemented as part of a traditional management platform) will retrieve the appropriate actions to correct the problem, as specified in the SLA. Before acting upon the managed system, it consults the business entity (see below) to verify if the proposed actions are allowable. After receiving approval, it applies the action(s) to the managed system.

It should be noted that the management component seeks approval for every proposed action from the business entity. 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. While such corrective actions are limited today to opening a trouble ticket or sending an event to the provider's management system, we envision this component playing a crucial role in the future by acting as an automated mediator between the customer and provider, according to the terms of the SLA. This includes the submission of proposals to the management system of a service provider on how a performance problem could be resolved (e.g., proposing to assign a different traffic category to a customer if several categories have been defined in the SLA).

Our implementation addresses very simple corrective actions; finding a generic, flexible and

automatically executable mechanism for corrective management actions remains an open issue yet.

Business Entity: It embodies the business knowledge, goals and policies of a signatory party (here: service provider), which are usually kept confidential. Such knowledge enables the business entity to verify if the actions specified in the SLA (eventually some time ago) are still compatible with the actual business targets. If this is the case, the business entity will send a positive acknowledgement to the request of the Management Service; in case the proposed actions are in conflict with the actual goals of the service provider, its business entity will decline the request and the management service will refrain from carrying them out. It should be noted that declining prior agreed-upon actions may be regarded by another party as a breach of the SLA entailing, in an severe case, termination of the business relationship. Since it is unlikely that decisions of this importance will be left to the discretion of an automated system, we assume that the decision of the business entity requires human intervention. While we have implemented the aforementioned services, we have postponed an implementation of a business entity component until appropriate mechanisms for specifying and enforcing business policies are available.

Our experience shows that the tasks covered by these two services become extremely complicated as soon as sophisticated management actions need to be specified: First, a service provider would need to expose what management operations he is able to execute, which is very specific to the management platforms (products, architectures, protocols) he uses. Second, these management actions may become very complicated and may require human interaction (such as deploying new servers). Finally, due to the fact that the provider's managed resources are shared among various customers, management actions that satisfy an SLA with one customer are likely to impact the SLAs the provider has with other customers. The decision whether to satisfy the SLA (or deliberately break it) therefore is not a technical decision anymore, but rather a matter of the provider's business policies and, thus, lies beyond the scope of the work discussed in this paper. Consequently, only a few elements of the WSLA language address this phase of the service lifecycle.

Phase 5: SLA Termination

The SLA may specify the conditions under which it may be terminated or the penalties a party will incur by breaking one or more SLA clauses. Negotiations for terminating an SLA may be carried out between the parties in the same way as the SLA establishment is being done. Alternatively, an expiration date for the SLA may be specified in the SLA.

SLA Compliance Monitor Implementation

Figure 2 shows which WSLA services have been implemented. Because of their major importance and their excellent suitability for automated processing, the **Deployment**, **Measurement** and **Condition Evaluation** services have been implemented by us. These services are implemented as Web Services themselves and are jointly referred to as **SLA Compliance Monitor**, which acts as a wrapper for the three services. For information where to download the implementation, the reader is referred to the 'Availability' section. Our ongoing implementation efforts, aimed at completing the WSLA framework, are described in 'Conclusions and Outlook.'

Signatory and Supporting Parties

Figure 3 gives an overview of a configuration where two signatory parties and two supporting parties collaborate in the monitoring of an SLA.

In bilateral SLAs, it is usually straightforward to define for each commitment who is the obliged and who is the beneficiary of the commitment. However, in an SLA containing more than two parties, it is not obvious which party guarantees what to whom. A clear definition of responsibilities is required. The WSLA environment involves multiple parties to enact an SLA instance. As mentioned above, a part of the monitoring and supervision activities can be assigned to parties other than the service provider and customer.

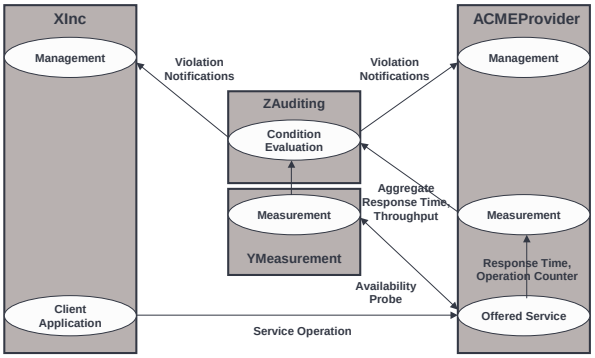

Figure 3: Signatory and supporting parties.

We approach the issue of responsibility by defining two classes of parties: Service provider (ACMEProvider in Figure 3) and service customer (XInc) are the signatory parties to the SLA. They are ultimately responsible for all obligations, mainly in the case of the service provider, and the ultimate beneficiary of obligations. Supporting parties are sponsored either by one or both of the signatory parties to perform one or more of a particular set of roles: A measurement service (YMeasurement) implements a part or all of the measurement and computation activities defined within an SLA. A condition evaluation service (ZAuditing) implements violation detection and other state checking functionality that covers all or a part of the guarantees of an SLA. A management service implements corrective actions.

There can be multiple supporting parties having a similar role, e.g., a measurement service may be

located in the provider's domain while another measurement service probes the service offered by the provider across the Internet from various locations. Keynote Systems, Inc. [15] is an example of such an external measurement service provider.

Despite the fact that a multitude of parties may be involved in providing a service, these interactions may be broken down into chained customer/provider relationships. Every interaction therefore involves only two roles, a sender and a recipient. During our work, we have not encountered a need for multi-party SLAs, i.e., SLAs that are *simultaneously* negotiated and signed by more than two parties. Multi-party contracts do not seem to provide enough value to justify their added complexity.

The WSLA Language

The WSLA language, specified in [19], defines a type system for the various SLA artifacts and is based on the XML Schema [34, 35]. We give an overview of the general structure of an SLA and motivate the various constructs of the WSLA language that will be described by means of examples in the subsequent sections: The information that needs to be processed by a Measurement Service is described later; then, we focus on the parts of the language a Condition Evaluation Service needs to understand for evaluating if a service level objective has been violated.

WSLA in a Nutshell

Figure 4 illustrates the typical elements of a SLA with signatory and supporting parties. Clearly, there are many variations of what types of information and which rules are to be included and, hence, enforced in a specific SLA.

The **Parties** section, consisting of the signatory parties and supporting parties fields identify all the contractual parties. **Signatory Party** descriptions contain the identification and the technical properties of a party, i.e., their interface definition and their addresses. The definitions of the **Supporting Parties** contain, in addition to the information contained in the signatory party descriptions, an attribute indicating the sponsor(s) of the party.

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. Examples of such bindings are SOAP (Simple Object Access Protocol), MIME (Multipurpose Internet Mail Extensions) or HTTP (HyperText Transfer Protocol).
- 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*.
- As mentioned earlier, every SLA parameter refers to one (composite) **Metric**, which, in turn, aggregates one or more other (composite or resource) 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 resource metrics are: *system uptime*, *service outage period*, *number of service invocations*.
  - **Measurement Directives** specify how an individual metric can be accessed. Typical

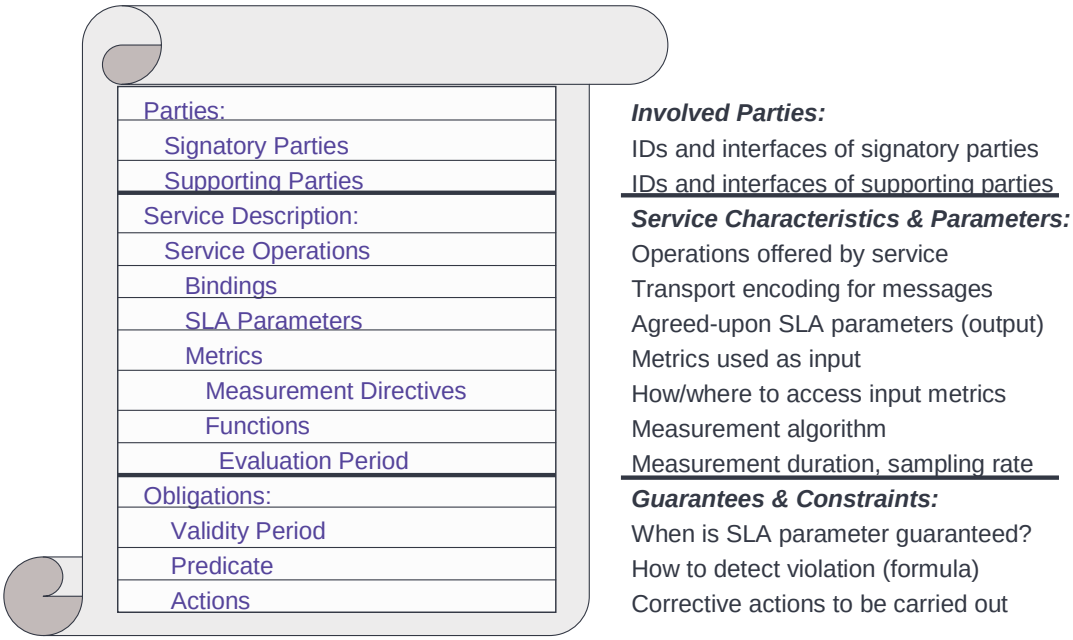

Figure 4: General structure of an SLA.

examples of measurement directives are the uniform resource identifier of a hosted computer program, a protocol message, or the command for invoking scripts or compiled programs.

- **Functions** are the measurement algorithm, or formula, that specifies how a composite metric is computed. Examples of functions are formulas of arbitrary length containing average, sum, minimum, maximum, and various other arithmetic operators, or time series constructors.
- For every function, an **Evaluation Period** is specified. It defines the time intervals during which the functions are executed to 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 various guarantees and constraints that may be imposed on the SLA parameters:

- First, the **Validity Period** is specified; it indicates the time intervals for which a given SLA parameter is valid, i.e., when the SLO may be applied. Examples of validity periods are *business days*, *regular working hours* or *maintenance periods*.
- The **Predicate** specifies the threshold and the comparison operator (greater than, equal, less than, etc.) against which a computed SLA parameter is to be compared. The result of the predicate is either *true* or *false*.
- **Actions**, finally, are triggered whenever a predicate evaluates to *true*, i.e., a violation of an SLO has occurred. Actions are e.g., *sending an event to one or more signatory and supporting parties*, *opening a trouble ticket or problem report*, *payment of penalty*, or *payment of premium*. Note that, as stated in the latter case, a service provider may very well receive additional compensation from a customer for exceeding an obligation, i.e., obligations reflect constraints that may trigger the payment of credits from any signatory party to another signatory or supporting party. Also note that zero or more actions may be specified for every SLA parameter.

Service Description: Associating SLA Parameters with a Service

The purpose of the service description is the clarification of three issues: *To which service do SLA parameters relate? What are the SLA parameters? How are the SLA parameters measured or computed?* This is the information a Measurement Service requires to carry out its tasks. A sample service description is depicted in Figure 5.

Service Objects and Operations

The service object, depicted at the top of Figure 5, provides an abstraction for all conceptual elements for which SLA parameters and the corresponding metrics can be defined. In the context of Web Services, the most detailed concept whose quality aspect can be described separately is the individual operation (in a binding) described in a WSDL specification. In our

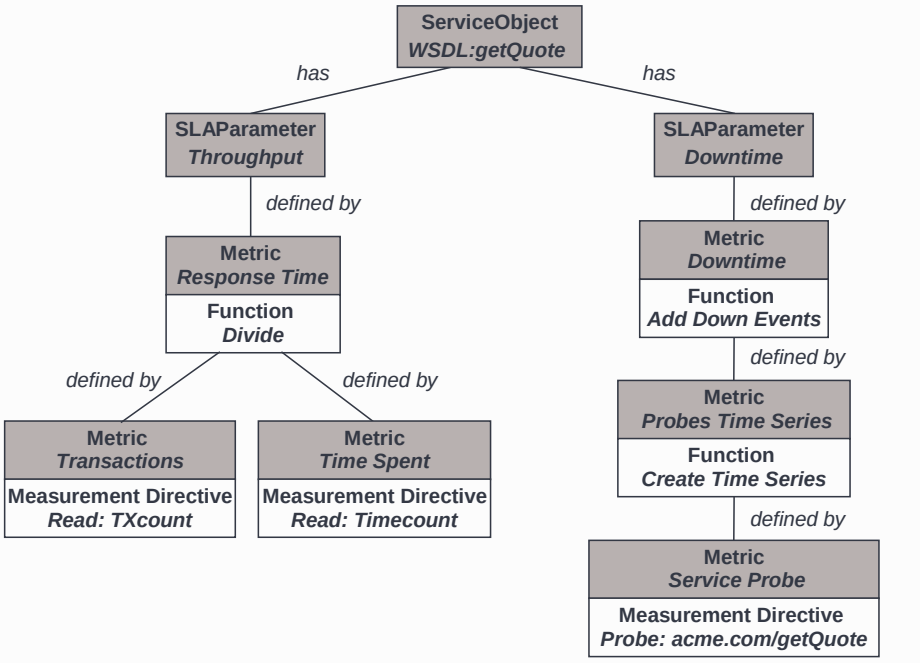

Figure 5: Sample elements of a service description.

example, the operation getQuote is the service object. In addition, quality properties of groups of WSDL operations can be defined – the operation group being the service object in this case. Outside the scope of Web Services, business processes, or parts thereof, can be service objects (e.g., defined in WSFL [18]). Service objects have a set of SLA parameters, a set of metrics that describe how SLA parameters are computed or measured and a reference to the service itself that is the subject of the service object abstraction.

While the format for SLA parameters and metrics is the same for all services (though not their individual content), the reference to the service depends on the particular way in which the service is described. For example, service objects may contain references to operations in a WSDL file.

SLA Parameters and Metrics

SLA parameters are properties of a service object; each SLA parameter has a name, type and unit. SLA parameters are computed from metrics, which either define how a value is to be computed from other metrics or describe how it is measured. For this purpose, a metric either defines a function that can use other metrics as operands or it has a measurement directive that describes how the metric's value should be measured. Since SLA parameters are the entities that are surfaced by a Measurement Service to a Condition Evaluation Service, it is important to define which party is supposed to provide the value (Source) and which parties can receive it, either event-driven (Push) or through polling (Pull). Note that one of our design choices is that SLA parameters are *always* the result of a computation, i.e., no SLA parameters can be defined as input parameters for computing other SLA parameters. In Figure 5, one metric is retrieved by probing an interface (Service Probe) while the other ones (TXcount, Timecount) are directly retrieved from the service provider's management system.

```
<SLAParameter name="UpTimeRatio"
   type="float" unit="downEvents/hour">
  <Metric>UpTimeRatioMetric</Metric>
  <Communication>
    <Source>ACMEProvider</Source>
    <Pull>ZAuditing</Pull>
    <Push>ZAuditing</Push>
  </Communication>
</SLAParameter>
```

Figure 6: Defining an SLA Parameter UpTimeRatio.

Figure 6 depicts how an SLA parameter UpTimeRatio is defined. It is assigned the metric UpTimeRatioMetric, which is defined independently of the SLA parameter for being used potentially multiple times. ACMEProvider promises to send (push) new values to ZAuditing, which is also allowed to retrieve new values on its own initiative (pull). The purpose of a metric is to define how to measure or compute a value. Besides a name, a type and a unit, it contains either a function or a measurement directive and a definition of the party that is in charge of computing this value.

Figure 7 shows an example composite metric containing a function. UpTimeRatioMetric is of type double and has no unit. YMeasurement is in charge of computing this value. The example illustrates the concept of a function: The number of occurrences of "0" in a time series of the metric StatusTimeSeries – assuming this represents a down event in time series of probes once per minute – is divided by 1440 (the number of minutes of a day) to yield the downtime ratio. This value is subtracted from 1 to obtain the UpTimeRatio. Specific functions, such as *Minus*, *Plus* or *ValueOccurs* are extensions of the common function type.

```
<Metric name="UpTimeRatioMetric" type="double" unit="">
    <Source>YMeasurement</Source>
    <Function xsi:type="Minus" resultType="double">
        <Operand>
          <LongScalar>1</LongScalar>
        </Operand>
        <Operand>
          <Function xsi:type="Divide" resultType="long">
            <Operand>
              <Function xsi:type="ValueOccurs" resultType="long">
                <Metric>StatusTimeSeries</Metric>
                <Value>
                  <LongScalar>0</LongScalar>
                </Value>
              </Function>
            </Operand>
            <Operand>
              <LongScalar>1440</LongScalar>
            </Operand>
          </Function>
        </Operand>
    </Function>
</Metric>
```

Figure 7: Defining a Metric UpTimeRatioMetric.

Operands of functions can be metrics, scalars and other functions. It is expected that a measurement service, provided either by a signatory or a supporting party, is able to compute functions. Specific functions can be added to the language as needed.

A **Measurement Directive**, depicted in Figure 8, specifies *how* the metric is retrieved from the source (either by means of a well-defined query interface offered by the Service Provider, or directly from the instrumentation of a managed resource by means of a management protocol operation). A specific type of measurement directive is used in the example above: StatusRequest. It contains a URL that is used for probing whether the getQuote operation is available. Apparently, other ways to measure values require an entirely different set of information items, e.g., an SNMP port, an object identifier (OID) and an instance identifier to retrieve a counter.

Obligations: SLOs and Action Guarantees

Based on the common ontology established in the service definition part of the SLA, the parties can unambiguously define the respective guarantees that they give each other. The WSLA language provides two types of obligations:

- **Service level objectives** represent promises with respect to the state of SLA parameters.
- **Action guarantees** are promises of a signatory party to perform an action. This may include notifications of service level objective violations or invocation of management operations.

Important for both types of obligations is the definition of the obliged party and the definition of when the obligations need to be evaluated. Both have a similar syntactical structure (as previously depicted in Figure 4). However, their semantics are different. The content of an obligation is refined in a service level objective or an action guarantee.

Service Level Objectives

A service level objective expresses a commitment to maintain a particular state of the service in a given period. Any party can take the obliged part of this guarantee; however, this is typically the service provider.

A service level objective has the following elements: Obliged is the name of a party that is in charge of delivering what is promised in this guarantee. One or many ValidityPeriods define when the SLO is applicable.

A logic Expression defines the actual content of the guarantee, i.e., what is asserted by the service provider to the service customer. Expressions follow first order logic and contain the usual operators *and*, *or*, *not*, etc., which connect either predicates or, again, expressions. Predicates can have SLA parameters and scalar values as parameters. By extending an abstract predicate type, new domain-specific predicates can be introduced as needed. Similarly, expressions could be extended, e.g., to contain variables and quantifiers. This provides the expressiveness to define complex states of the service.

A service level objective may also have an EvaluationEvent, which defines when the expression of the service level objective should be evaluated. The most common evaluation event is NewValue, each time a new value for an SLA parameter used in a predicate is available. Alternatively, the expression may be evaluated according to a Schedule. A schedule is a sequence of regularly occurring events. It can be defined within a guarantee or may refer to a commonly used schedule.

The example in Figure 9 illustrates a service level objective given by ACMEProvider and valid for a full month in the year 2001. It guarantees that the SLA parameter ThroughPutRatio must be greater than 1000 if the SLA parameter UpTimeRatio is less than 0.9, i.e., the ThroughPutRatio must be above 1000 transactions per minute even if the overall availability is below 90%. This condition should be evaluated each time a new value for the SLA parameter is available. Note that we deliberately chose that validity periods are always specified with respect to a single SLA parameter, and thus only indirectly applicable to the scope of the overall SLA. Alternatively, validity periods to the overall SLA (possibly in addition to the validity periods for each SLA parameter) could be possible, but we found that this granularity is too coarse.

Action Guarantees

An action guarantee expresses a commitment to perform a particular activity if a given precondition is met. Any party can be the obliged of this kind of guarantee. This particularly includes also the supporting parties of the SLA.

An action guarantee comprises the following elements and attributes: Obliged is the name of a party that must perform an action as defined in this guarantee. A logic Expression defines the precondition of the action. The format of this expression is the same as the format of expression in service level objectives. An important predicate for action guarantees is the Violation predicate that determines whether another guarantee, in particular a service level objective, has been violated. An EvaluationEvent or an evaluation Schedule defines when the precondition is evaluated.

```
<Metric name="ServiceProbe" type="integer" unit="">
    <Source>YMeasurement</Source>
    <MeasurementDirective xsi:type="StatusRequest" resultType="integer">
        <RequestURL>http://ymeasurement.com/StatusRequest/GetQuote</RequestURL>
    </MeasurementDirective>
</Metric>
```

Figure 8: Defining a Measurement Directive StatusRequest.

QualifiedAction contains a definition of the action to be invoked at a particular party. The concept of a qualified action definition is similar to the invocation of an object method in a programming language, replacing the object name with a party name. The party of the qualified action can be the obliged or another party. The action must be defined in the corresponding party specification. In addition, the specification of the action includes the marshalling of its parameters. One or more qualified actions can be part of an action guarantee.

ExecutionModality is an additional means to control the execution of the action. It can be defined whether the action should be executed if a particular evaluation of the expression yields true. The purpose is to reduce, for example, the execution of a notification action to a necessary level if the associated expression is evaluated very frequently. Execution modality can be either: *always*, *on entering a condition* or *on entering and leaving a condition*. The example depicted in Figure 10 illustrates an action guarantee.

In the example, ZAuditing is obliged to invoke the notification action of the service customer XInc if a violation of the service level objective SLO_For_ThroughPut_and_UpTime (cf. Figure 9) occurs. The precondition should be evaluated every time the

```
<ServiceLevelObjective name="SLO_For_ThroughPut_and_UpTime">
    <Obliged>ACMEProvider</Obliged>
    <Validity>
        <Start>2001-11-30T14:00:00.000-05:00</Start>
        <End>2001-12-31T14:00:00.000-05:00</End>
    </Validity>
    <Expression>
      <Implies>
        <Expression>
          <Predicate xsi:type="Less">
            <SLAParameter>UpTimeRatio</SLAParameter>
            <Value>0.9</Value>
          </Predicate>
        </Expression>
        <Expression>
          <Predicate xsi:type="Greater">
            <SLAParameter>ThroughPutRatio</SLAParameter>
            <Value>1000</Value>
          </Predicate>
        </Expression>
      </Implies>
    </Expression>
    <EvaluationEvent>NewValue</EvaluationEvent>
</ServiceLevelObjective>
```

Figure 9: Defining a Service Level Objective SLO_For_ThroughPut_and_UpTime.

```
<ActionGuarantee name="Must_Send_Notification_Guarantee">
  <Obliged>ZAuditing</Obliged>
  <Expression>
    <Predicate xsi:type="Violation">
      <ServiceLevelObjective>
           SLO_For_ThroughPut_and_UpTime
      </ServiceLevelObjective>
    </Predicate>
  </Expression>
  <EvaluationEvent>NewValue</EvaluationEvent>
  <QualifiedAction>
    <Party>XInc</Party>
    <Action actionName="notification" xsi:type="Notification">
      <NotificationType>Violation</NotificationType>
      <CausingGuarantee>
         Must_Send_Notification_Guarantee
      </CausingGuarantee>
      <SLAParameter>ThroughPutRatio UpTimeRatio</SLAParameter>
    </Action>
  </QualifiedAction>
  <ExecutionModality>Always</ExecutionModality>
</ActionGuarantee>
```

Figure 10: Defining an ActionGuarantee Must_Send_Notification_Guarantee.

evaluation of the SLO Must_Send_Notification_Guarantee returns a new value. The action has three parameters: the type of notification, the guarantee that caused it to be sent, and the SLA parameters relevant for understanding the reason of the notification. The notification should always be executed.

Conclusions and Outlook

This paper has introduced the novel WSLA framework for electronic services, in particular Web Services. The WSLA language allows a service provider and its customer to define the quality of service aspects of the service. The concept of supporting parties allows signatory parties to include third parties into the process of measuring the SLA parameters and monitoring the obligations associated with them. In order to avoid the potential ambiguity of high-level SLA parameters, parties can define precisely how resource metrics are measured and how composite metrics are computed from others. The WSLA language is extensible and allows us to derive new domain-specific or technology-specific elements from existing language elements. The explicit representation of service level objectives and action guarantees provides a very flexible mechanism to define obligations on a case-by-case basis. Finally, the detachment from the service description itself makes the WSLA language and its associated services applicable to a wide range of electronic services.

We have developed a prototype that implements a total of three different WSLA services: First, a deployment service to provide the measurement and condition evaluation services with the SLA elements they need to know; second, a measurement service that can interpret measurement directives for the instrumentation of a Web services gateway and can aggregate high-level metrics using a rich set of functions for arithmetic and time series. Third, a general-purpose condition evaluation service has been implemented that supports a wide range of predicates. Currently, we provide extensions to the WSLA language that apply to quality aspects of business processes and a template format for advertising SLAs in service registries such as UDDI. In addition, we are working on an SLA editing environment. The integration with existing resource management systems and architectures is currently underway, with a special focus on the *Common Information Model (CIM)*.

Availability

The SLA Compliance Monitor is included in the current version 3.2 of the IBM Web Services Toolkit and can be downloaded from http://www.alphaworks.ibm.com/tech/webservicestoolkit.

Acknowledgments

The authors would like to express their gratitude to Asit Dan, Richard Franck and Richard P. King for their contribution. The authors are also indebted to Steve Traugott of TerraLuna, LLC. for his constructive suggestions for improving the quality of this paper.

Biography

Alexander Keller is a Research Staff Member at the IBM Thomas J. Watson Research Center in Yorktown Heights, NY, USA. He received his M.Sc. and a Ph.D. in Computer Science from Technische Universität München, Germany, in 1994 and 1998, respectively and has published more than 30 refereed papers in the area of distributed systems management. He does research on service and application management, information modeling for e-business systems, and service level agreements. He is a member of GI, IEEE and the DMTF CIM Applications Working Group.

Heiko Ludwig is a visiting scientist at the IBM Thomas J. Watson Research Center since June 2001. As a member of the Distributed Systems and Services department he works in the field of electronic contracts, both contract representation and architectures for contract-based systems. He holds a Master's degree (1992) and a Ph.D. (1997) in computer science and business administration from Otto-Friedrich University Bamberg, Germany.

References

[1] ASP Industry Consortium, *White Paper on Service Level Agreements*, 2000.

[2] Bhoj, P., S. Singhal, and S. Chutani, ''SLA Management in Federated Environments,'' *Proceedings of the Sixth IFIP/IEEE Symposium on Integrated Network Management (IM'99)*, Boston, MA, IEEE Publishing, pp. 293-308, May 1999.

[3] Bhushan, B., M. Tschichholz, E. Leray, and W. Donnelly, ''Federated Accounting: Service Charging and Billing in a Business-To-Business Environment,'' *Proceedings of the 7th IFIP/IEEE International Symposium on Integrated Network Management*, (N. Anerousis, G. Pavlou, and A. Liotta, editors), Seattle, WA, USA, IEEE Publishing, pp. 107-121, May 2001.

[4] Dan, A., D. Dias, R. Kearney, T. Lau, T. Nguyen, F. Parr, M. Sachs, and H. Shaikh, ''Business-to-Business Integration with tpaML and a B2B Protocol Framework,'' *IBM Systems Journal*, Vol. 40, No. 1, February 2001.

[5] Debusmann, M., M. Schmidt, and R. Kroeger, ''Generic Performance Instrumentation of EJB Applications for Service Level Management,'' Stadler and Ulema [28].

[6] *ebXML – Creating a Single Global Electronic Market*, http://www.ebxml.org .

[7] ebXML Trading-Partners Team, *Collaboration-Protocol Profile and Agreement Specification, Version 1.0*, UN/CEFACT and OASIS, May 2001.

[8] FORM Consortium, *Final Inter-Enterprise Management System Model*, Deliverable 11, IST

Project FORM: Engineering a Co-operative Inter-Enterprise Framework Supporting Dynamic Federated Organisations Management, http://www.ist-form.org, February 2002.

[9] Garschhammer, M., R. Hauck, H.-G. Hegering, B. Kempter, M. Langer, M. Nerb, I. Radisic, H. Roelle, and H. Schmidt, "Towards Generic Service Management Concepts: A Service Model Based Approach," *Proceedings of the 7th IFIP/IEEE International Symposium on Integrated Network Management*, pp. 719-732.

[10] Gopal, R., "Unifying Network Configuration and Service Assurance with a Service Modeling Language," In Stadler and Ulema [28], pp. 711-725.

[11] Griffel, F., M. Boger, H. Weinreich, W. Lamersdorf, and M. Merz, "Electronic contracting with COSMOS – How to establish, Negotiate and Execute Electronic Contracts on the Internet," *Proceedings of the Second International Enterprise Distributed Object Computing Workshop (EDOC '98)*, La Jolla, CA, USA, October 1998.

[12] Grosof, B. and Y. Labrou, "An Approach to using XML and a Rule-based Content Language with an Agent Communication Language," *Proceedings of the IJCAI-99 Workshop on Agent Communication Languages (ACL-99)*, 1999.

[13] Hoffner, Y., S. Field, P. Grefen, and H. Ludwig, "Contract-driven Creation and Operation of Virtual Enterprises," *Computer Networks*, Vol. 37, pp. 111-136, 2001.

[14] Keller, A., G. Kar, H. Ludwig, A. Dan, and J. L. Hellerstein, "Managing Dynamic Services: A Contract based Approach to a Conceptual Architecture," In Stadler and Ulema [28], pp. 513-528.

[15] *Keynote – The Internet Performance Authority*, http:// www.keynote.com .

[16] Kreger, H., *Web Services Conceptual Architecture 1.0*, IBM Software Group, May 2001.

[17] Lewis, L., *Managing Business and Service Networks*, Kluwer Academic Publishers, 2001.

[18] Leymann, F., *Web Services Flow Language (WSFL) 1.0*, IBM Software Group, May 2001.

[19] Ludwig, H., A. Dan, R. Franck, A. Keller, and R. P. King, *Web Service Level Agreement (WSLA) Language Specification*, IBM Corporation, July 2002.

[20] Ludwig, H. and Y. Hoffner, "The Role of Contract and Component Semantics in Dynamic E-Contract Enactment Configuration," *Proceedings of the 9th IFIP Workshop on Data Semantics (DS9)*, pp. 26-40, 2001.

[21] McCloghrie, K., D. Perkins, and J. Schoenwaelder, *Structure of Management Information – Version 2 (SMIv2), RFC 2578*, IETF, April 1999.

[22] Merz, M., F. Griffel, T. Tu, S. Müller-Wilken, H. Weinreich, M. Boger, and W. Lamersdorf, "Supporting Electronic Commerce Transactions with contracting Services," *International Journal of Cooperative Information Systems*, Vol. 7, No. 4, pp. 249-274, 1998.

[23] Moore, B., E. Ellesson, J. Strassner, and A. Westerinen, Policy Core *Information Model – Version 1 Specification, RFC 3060*, IETF, February 2001.

[24] Rodosek, G. Dreo and L. Lewis, "Dynamic Service Provisioning: A User-Centric Approach," *Proceedings of the 12th Annual IFIP/IEEE International Workshop on Distributed Systems: Operations & Management (DSOM 2001)*, O. Festor and A. Pras, editors, Nancy, France, IFIP/IEEE, INRIA Press, pp. 37-48, October 2001.

[25] Schopp, B., A. Runge, and K. Stanoevska-Slabeva, "The Management of Business Transactions through Electronic Contracts," *Proceedings for the Tenth International Workshop on Database and Expert Systems Applications*, A. Camelli, A. Min Tjoa, and R. R. Wagner, editors, Florence, Italy, IEEE Computer Society Press, pp. 824-831, 1999.

[26] SLA and QoS Management Team, *Service Provider to Customer Performance Reporting: Information Agreement,* Member Draft Version 1.5 TMF 602, TeleManagement Forum, June 1999.

[27] SLA Management Team, *SLA Management Handbook,* Public Evaluation Version 1.5 GB 917, TeleManagement Forum, June 2001.

[28] Stadler, R. and M. Ulema, editors, *Proceedings of the IEEE/IFIP Network Operations and Management Symposium*, Florence, Italy, IEEE Press, April 2002.

[29] *Enhanced Telecom Operations Map: The Business Process Framework*, Member Evaluation Version 2.7 GB 921, TeleManagement Forum, April 2002.

[30] Tosic, V., B. Pagurek, B. Esfandiari, and K. Patel, "Management of Compositions of E- and M-Business Web Services with multiple Classes of Service," Stadler and Ulema [28], pp. 935-937.

[31] *UDDI Version 2.0 API Specification*, Universal Description, Discovery and Integration, http://uddi. org, June 2001.

[32] Verma, D., *Supporting Service Level Agreements on IP Networks*, Macmillan Technical Publishing, 1999.

[33] White, K., *Definition of Managed Objects for Service Level Agreements Performance Monitoring, RFC 2758*, IETF, February 2000.

[34] *XML Schema Part 1: Structures*, W3C Recommendation, W3 Consortium, May 2001.

[35] *XML Schema Part 2: Datatypes*, W3C Recommendation, W3 Consortium, May 2001.