Web Services service level management: overview of service level agreement languages and support infrastructures

Tuomas Nurmela

Abstract—Extended service-oriented architecture –based web services for open environments sets numerous requirements. Issues such as composition and management of services become important, yet neither web services nor the current commercial platforms contain the required capabilities, let alone are mature enough for production use. This paper focuses in particular to service-level management, providing a frame of reference to categorize the dispersed research arena. The paper also contains a comparison of the best known approaches in web services languages and support infrastructures to investigate the differences and commonalities in current initiatives. In order to understand to put the differences in approaches to context, a comprehensive overview of the background relating to the concept of quality of service is provided.

Keywords: Extended Service-Oriented Architecture, Service Level Management, Quality of Service, Web Services

I. INTRODUCTION

Web Services are fast becoming the approach for software interoperability and service oriented architecture implementation. If utilized in manner following service-oriented architecture, the approach provides benefits such as adaptation to local and environmental changes, management of environment heterogeneity and autonomous decisions over service implementation [1].

However, as suggested in extended SOA [2][3], deployment of basic SOA-based architecture is just the first step, with composition and management of services following behind. Both areas introduce new challenges in building advanced environments such as open web services provider markets and open environments such as grid systems and virtual organizations, which contain high local and environmental dynamism. To counter and control these, both development practices (referring to model-driven development efforts) and required middleware support infrastructure need to be reconsidered. This paper discusses the management of services, especially focusing on the middleware support infrastructures. Underlying assumption is that support infrastructures need to be able to express their requirements and utilize generic support infrastructure instead of integrating the logic to each application.

This paper focuses on a particular area of management of services, service level management (SLM), which contains Quality of Service (QoS) –descriptions for web services in the form of service level agreements (SLAs) and support infrastructures to manage, monitor and evaluate the realization of these agreements. Relationship between web service management and web service SLM (e.g. using web service management to manage SLA offering or using web service management as one mechanism in SLA breach management) is outside the scope of the paper. Also, tools for QoS-aware service design and design validation are outside the scope of the paper.

General service selection algorithms or negotiation protocols, which can be utilized with SLAs, are also outside the scope of the paper. The reader is referred to [4] for an overview of different general negotiation protocols.

Web Services QoS has been approached both from performance metrics perspective (narrow context) and from non-functional aspects perspective in general (wide context). The former approach always contains the notion of run-time monitoring. The latter approach is typically more static, supporting design time validation. Examples of wide context approach include WS-policy framework and competitive approaches (e.g. WSPL, WSPC, OWL mappings for WS-Policy). An overview of these is provided in [5]. Extensions to WS-Policy from a more generic QoS perspective include e.g. GlueQoS [6].

This paper focuses on the narrow context and its extensions for Web Services SLM to discuss performance, whether in the context of technical performance metrics or business performance metrics. Likewise, focus is on the support required by different stages of the SLM.

The structure of the paper is as follows: section II discusses the background of Quality of Service from the perspectives of network management and software engineering. Section III discusses the challenges posed by service level management in web services in different layers of extended service oriented architecture. Section IV introduces a frame of reference to support SLM solution evaluation in the sections that follow. Section V introduces and compares service level agreement languages. Section VI introduces and compares support infrastructures. Section VII provides the summary and conclusions.

II. BACKGROUND TO QUALITY OF SERVICE

This section discusses the relevant background from service and network management and software and systems engineering disciplines. These provide the context to the
SLM evaluation framework and approaches discussed in following sections.

A. Service and network management perspectives

Service and network management both contain their own terminology in terms of QoS, with varying differences. One of the varying factors is scope: Stereotypic view is that networking QoS is typically tied to performance metrics (especially in terms of QoS network technologies) whereas in service management QoS relates to both user experience and operational service functionality. Here we focus on service management SLAs. It should be noted that this separation is somewhat artificial given that e.g. ITU separates offered, required, perceived and achieved levels of quality [7].

Service level agreements (SLAs) are formal contracts negotiated with the customers. SLA definition typically includes SLA templates, which are predefined contract templates, SLA parameters defining what can be measured for a particular service (either business or technical perspective), Service Level Objectives (SLOs) are particular values given to QoS parameters in a given SLA. Alternatively to SLOs, Classes of Service bundle different parameters and their thresholds to provide a limited number of choices for the customer to select from. Alternatively a custom SLA can be negotiated with the customer, in which case SLOs and even QoS parameters can be defined on per-customer basis.

In addition to SLOs agreed with the external customer, internal thresholds can be defined in order to provide means of alerting the service management staff prior to actual breach of SLOs. These thresholds can be static or dynamic, depending on operations support software (OSS, e.g. trouble ticket system) capabilities. Likewise, maintenance windows and other factors (such as force majore) are typically excluded with separate clauses although custom SLAs can contain negotiated items on these also.

In terms of types of SLAs [8], service providers typically create both internal SLAs and external SLAs. Internal SLAs codify what is expected of different units within the service provider company to create the service. These are also called Operational Level Agreements (OLA). If the service provider utilizes third parties to provide the service, an underpinning contract (UC) is created with them. External SLAs codify what is being offered to the external customer. A central tenant is that internal SLAs relating to the service (whether OLAs and UCs) are more stringent than external SLAs. An alternative from service provider would be to approach the issue as a risk management scenario and include SLO breach related sanctioning to pricing. However, intuitively this does not lead to long customer relationships given that customer can almost never negotiate the actual financial loss as part of the breach management payoff.

Service level management (SLM) is the business process that contains all the activities relating to SLAs and their management. In business environments, SLM as a process roughly contains the activities of defining SLAs, negotiating SLAs (or buyer selection based on classes of service), monitoring and evaluation of SLAs and managing breaches of SLAs (although in large outsourcing deals, this could bonus management for possible constant over-achievement of SLOs), SLM also contains the notion of reporting the results to the customer. This business-centric approach is the central difference between thinking about management of QoS contracts and management of SLAs.

As can be seen, the SLM process activities are nearly the same as for eContracting [25]. However, the difference lies in the scope: in open, dynamic environments, eContracting is required to negotiate the common process between collaborators (e.g. when forming a virtual breeding environment) and between a virtual organization instance and customers when forming an external contract and ensuring that what is agreed will be honored by all parties. Likewise, issues such as capability to utilize support infrastructure in a federation is required. However, in SLM, the focus is only on managing the QoS defined in the SLA (i.e. its selection or negotiation, validation of design, verification of performance, breach and bonus management and reporting).

Internal and external SLAs translate directly to certain kinds of dynamic web services environments, such as the open environments (e.g. multi-agent systems, grid systems and virtual organizations). As an example, in Virtual Organizations, conceptually a virtual breeding environment (VBE) is created on the basis that each member of the VBE is agrees to provide a certain level of service to the service it is willing to provide as part of a virtual organization. Based on these internal SLAs (OLAs or UCs) to the VBE, external SLAs can be offered or negotiated to the external customers.

While SLAs are covered, in regards to web services background, two characteristics of network management are worth noting: first, billing integration is central to quality of service provided. This is very typical in the telco world. Second, network management access control and quality of service management were one of the first areas to widely deploy policy-based management to abstract the management functions from individual vendor implementations. The IETF policy framework [9][10] extends this idea by separating policy decision making from enforcement and management.

B. Software and systems engineering perspectives

Quality of service in software engineering has been approached in software design and development (e.g. design by contract language extensions, meta-object protocol languages, component-based software engineering) as well as in systems engineering (operating systems, middleware support infrastructure).

Contract-support for languages [11] has been suggested for both procedural and object-oriented programming. In the “design by contract” –based language or precompiler extensions, type management of parameters are extended with invariants and pre- and post-conditions. The focus in contracts was to supplement expressiveness on the level of a particular programming language. Contracts enabled means for expressing QoS concerns in terms of e.g. atomicity.

When component-based software engineering (CBSE) arrived, the main issue was taking advantage of reusability offered by the coarser-granularity provided by packaging of CBSE approach. Contracts attached to a method interface
means to additionally express memory, performance bounds and safety descriptions. Breaches of the contract would be reported by the system. Therefore, the 3rd party providing the component could be held accountable. At least this was the theory. The approach never gained widespread commercial popularity [12, s.52].

Meta-object protocol languages [13] provide means for separation of concerns from regarding non-functional aspects (NFAs). This provides means for wide context of QoS support such as reliability and security from the business application perspective. The MOP extensions are e.g. annotations in the application code that are pre-compiled prior to application compilation (or interpretation). To similar vein, aspect-orientation [14] separates the MOP code to separate files and provides means to weave these to the application (or provide hooks that can be at run-time accessed).

Compared to design by contract, meta-object protocols (MOPs) provided a capability for more comprehensive expressions, yet tied the properties to the MOP language. However, this approach provided impetus for the idea of developer specialization.

While functional programming, OOP and CBSE provided means to integrate the QoS view to the language, systems engineering has looked at operating systems and middleware approaches from the infrastructure perspective. Instead of specialized developers in the business application development team, the development capability is required from the systems engineers. Business application developers only focus on the infrastructure configuration. Likewise, with correct systems architecture, reuse point can be moved from individual applications to a centrally managed point.

Distributed object computing (DOC) middleware translated the design-by-contract -approach to adaptive distributed systems: instead of static definitions, QoS-performance bounds on resources and a supporting infrastructure provided means of evaluating the system performance at runtime. Furthermore, with approaches such as open implementation, the meta-interfaces providing management of the NFAs became explicit, providing capability to adapt to runtime environment and application changes. This was especially exploited in multimedia applications -focused middleware. Examples of these approaches include systems such as Quality Objects [15] on CORBA. These systems did not neglect the software engineering lifecycle and hence requirements and design time definition of QoS-parameters typically in separate QoS-languages were included, even though the focus was on moving from integrating the QoS infrastructure at compile-time to integrating it at runtime. However, the scope of QoS was narrowed from generic non-functional aspects to performance-related QoS-parameters. Likewise, interoperability between different middleware and QoS-support infrastructures was not considered. A comprehensive overview of QoS-middleware approaches and languages is provided by Aagedal [16].

Type repositories [17, pp. 120-123] can be seen to extend the notion of contracts and QoS-languages to include also the perspective of interoperability between different languages. The extension provides both breadth and multiple layers to support abstraction and match making. In terms of breadth, service types contain technical, semantic and behavioral descriptions as well as non-functional aspects of service (i.e. QoS in the wide context) [18]. This is understandable, given that the focus on interoperability. In terms of levels of abstraction, service types include abstract service type description, specific service type language descriptions and implementation descriptions. Logical equivalence is required on the abstract service type layer to enable interoperability. The main concern can be said to be in interoperability to support collaboration in DOC middleware [17] or web services based open environments [19][20], where the focus is on the behavioral specification.

This comprehensive, rich background becomes evident when surveying today’s initiatives web services -based systems and serves as the main definition for context for projects surveyed in this paper.

III. EXTENDED SOA, SLM AND WEB SERVICES

In order to utilize quality of service and service level management in the extended SOA -model [2][3] context, the requirements for each level in terms of web services need to be discussed. Likewise, the model contains few implicit assumptions regarding roles without considering their impact on providing quality of service support as shared services in a managed environment. Figure 1 provides an overview of the varying focus areas for SLM.

![Figure 1: Extended SOA and SLM](image)

The foundation layer of the model, “Description and basic operations”, focuses on the publish-find-bind -approach. At the foundation layer the services are atomic, i.e. focus is on a particular service that does not rely on any other service. Therefore SLA parameters and SLOs (narrow context) or NFAs (wide context) are relatively simple to define, design and implement.

While the extended SOA-model merely discusses the QoS of services, from system QoS perspective this is incomplete. Due to the indirection layer of the directory, QoS requirements for the directory also need to be addressed. In
this sense, in web services implementations the requirements can be either focused on a particular UDDI directory (assumes centralized directory) or the UDDI directory system (assumes distributed directory).

In addition to the QoS requirements of the directory, the directory support for QoS-based service descriptions and finding are relevant from implementation perspective. If the QoS is attached to the endpoints this hampers utilization of the directory. On the other hand, publishing QoS descriptions exclusively in the directory would not support web service deployments today.

The intermediate layer, "Composition", requires that composition is done by a service aggregator, whether this role merely composes the service or participates in producing it. From QoS and SLM perspective in web services, things get a bit complicated. QoS requirements derived from multiple web services depends on whether or not we are viewing QoS in narrow or wide context. Intuitively, in narrow context QoS SLA is simply performance metrics aggregation: availability of service A * availability of service B equals availability of composed service (A,B). Similar approach can be utilized for response time and other metrics. However, this disregards possible semantic differences, i.e. (i) what is meant by a given SLA parameter (e.g. does availability calculation include notified service breaks, is response time the response time of service or operational response time of service desk to start fixing possible incidents concerning the web service platform) and (ii) how is the SLA parameter monitored (e.g. is availability based on polling the service or calculated post-monitoring period from trouble tickets are very different approaches).

The wider context of QoS is even more problematic as QoS now relates to NFAs in general. Even if semantic differences could be solved, matching the NFA requirements of the consumer to all the producers participating to the composite service requires multiparty selection or negotiation.

The top layer, "Management", introduces new roles rather than service attributes. The market maker would in web services context be a web services service provider that would publish of web services on an open market and to some degree manage that market. The service operator on the other hand manages dedicated atomic services and composed services of web service producers and web service aggregators. This management would incorporate web service management, not operational platform management (see e.g. [21]).

In the top layer, SLM perspective changes altogether as the focus shifts from QoS of services to roles. From consumer, producer and aggregator perspective QoS descriptions of the service are not only selected or negotiated as part of the service finding. Rather they are obligations to which the service producer (or aggregator) is (legally) bound to by the SLA.

To support such notions new participants are (roles) producing the management layer must provide means for (legal) certification of SLAs. Likewise, rating the participants through reputation services to provide e.g. means of expressing breaches of SLAs to all participants is required.

The discussion of extended SOA-model falls rather short on these issues. While the new roles for the management layer seem to be defined, a number of issues remain. First, it remains unclear whether market maker could in practice only (i) run a directory with additional reputation services describing providers, monitoring services to measure performance and SLAs services providing signed contracts between different parties or (ii) also provide an execution environment for services. The first approach implies that the SLA parameters are defined in the directory with service descriptions (i.e. they are not managed by endpoints), because otherwise the market maker would not be able to be responsible for them. The second option extends the first by supporting the possibility that they can be attached to endpoints or managed separately in the in-band path (along with other support infrastructure messaging extensions). This approach provides means for the service provider to create shared services by doing in-band message transformations or manipulations with e.g. in web services using SOAP intermediaries. Given the rising popularity of XML firewalls [22][23], this is not as improbable as it would first sound.

Secondly, the same question can be asked from a service operator perspective, i.e. does the role only manage the services or potentially provides their execution environment.

Thirdly, obviously service operators need to play a role in SLAs, as they are responsible management of service. This would suggest also rating the operators and making this participation (legally) binding by being named in the SLA.

Fourth, besides the practical issues discussed, the management layer contains the architectural notion that market maker provides means for rating (with reputation services) service providers, aggregators and potentially operators. However, given that this role also provides other services a clear conflict of interest is shown: market maker cannot rate itself. Therefore, intuitively reputation services should be maintained by a separate trusted third party.

IV. FRAME OF REFERENCE FOR SLM

Given the multiple backgrounds in QoS and the fact that there are even no draft standards in the web services arena, it comes as no surprise that none of the approaches offers a complete solution to the challenges described. In order to provide means for evaluation, first the scope of SLAs is compared to eContracts. After this a frame of reference based on SLM process is described.

SLA contract scope needs to be considered in addition to considering different roles that may be related to producing the service. This impacts to a great extent the functionality that is required to its management. Here the scope is discussed in comparison to eContracts.

When focusing on QoS from the narrow, performance metrics perspective, QoS can either deal with technical metrics or it can deal with business metrics. Ideally the technical metrics can be aggregated to business metrics. However, (i) this is not always possible (if business impact of technical metric is not known, yet technical metric is expressed as a QoS parameter by the web services provider) and (ii)
sometimes the business metric is inexpressible in measurable technical parameters.

Figure 2 describes a suggestion for minimal content with regards to SLM from SLA and eContracting.

![Diagram](Diagram.png)

**Figure 2: Minimal scope of contract content from SLM perspective**

The separation of SLA management from the eContracts provides the benefit in terms of reuse and breadth to which the language can be applied. The figure does not suggest that (i) selection or negotiation would be required to be made at separate levels or that (ii) business metrics would be only business process metrics or that (iii) eContracts need to include business network information or that (iv) only eContracts would be legally binding. These issues are outside the scope of discussion here.

Separation of technical metrics from business metrics would support (i) system modularity and (ii) specification of third party roles in order to manage a specific area of responsibility (e.g. monitoring and evaluation of technical SLA parameters or NFA properties). Suggestion by Haataja [24, pp. 11-12] to separate of virtual organization monitoring from the actual business protocol monitoring are on the same line in terms of modularizing concerns.

While the scope of SLA does impact SLA management, it does not provide means to determine SLM activities. However, the relationship with eContracting does suggest eContracting can be used to determine SLM activities as was discussed in the background. A frame of reference provided in figure 3. This is somewhat based on the process model [8] and on eContracting process [25]. We next discuss its utilization for evaluating work in web services SLM arena.

![Diagram](Diagram.png)

**Figure 3: Frame of reference for SLM**

**SLA template design** consists of defining the XML elements of the SLA. If the SLA is to be negotiated, SLOs are dynamically established, only parameter boundaries need to be defined. Alternatively, if class of service – approach is used, classes need to be defined. The template design is particularly impacted by the SLA language design choices.

**SLA-enhanced process design** relates to utilization of composite services: SLAs may be involved at design time of the process (composite service), especially if the process is private and therefore only internal SLAs are involved. SLA-enhanced process design requires that process design tool support SLAs.

After creation of process, the SLA-enhanced process design may be validated at design time. This requires extending the type repository to include SLA validation support.

At runtime, after deployment of service, the consumer either negotiates the required SLOs or selects an appropriate class of service. In case web services service providers function in an open market, it is possible that the web service consumer participates to an auction for the best possible service web. This would require a negotiation mechanism support for multiparty negotiation. Alternative approaches include capability to select an identical service from each service provider and only provide payment for the fastest [26]. In addition, the offered services can be demand-constrained. In this case the negotiation may be resolve around multiple consumers auctioning for a single provider. As can be seen, the SLA determination can be modeled as a full-blown auctioning or bargaining scenario. However, this is typically not required in practice, because of SLA limited scope. Likewise, the negotiation can be separated under a separate negotiation protocol.

**Monitoring of SLA parameters** contains at least two issues: first the monitoring can be done either in-band or out-of-band. Out-of-band monitoring, following typical probe-approach, is suitable for narrow, performance metric-based SLM. Wider perspective may require in-band monitoring, which in itself may be located to the service or the access layer of the service responsible for routing and access control. Given the separation of access layer is becoming increasingly popular with “XML firewalls”, the additional functionality fits well to this approach.

Secondly, monitoring link to evaluation can be passive, reactive or proactive [25][24]. Passive monitoring merely refers to logging monitoring data at run time. Evaluation is done later as a separate action. Reactive monitoring provides means for evaluation of SLO breaches for corrective actions. Proactive monitoring would support thresholds prior to SLO breach and actions that would try to ensure breach of SLO would not happen. Evaluation therefore includes threshold evaluation in addition to SLO breach evaluation.

**Evaluation of SLOs** can be based on different modes (event-based (with e.g. schedules) or request-based). Likewise it can support complete evaluation (i.e. utilize all available monitoring data) and/or statistical evaluation (i.e. evaluate only a sample of monitoring data). Evaluation accuracy can be dependent on the monitoring data sources: e.g. in availability evaluation trouble tickets are the source of evaluation, a human element is involved, where as with end-to-end -polling -based monitoring, frequency of polling establishes accuracy. Evaluation is called dynamic verification by Haataja [24].
SLA breach management governs SLO or proactive threshold breaches, i.e. it is closely tied to the monitoring link. E.g. with passive monitoring link, breach management is typically done posteriori by people. While little research on automated breach management is available, intuitively (i) this is done by consumer and/or provider and (ii) not all possible mechanisms fit the different monitoring link (i.e. reactive or proactive) types. Likewise, intuitively a number of mechanisms are possible including:

- Using long-running transactions and their compensation mechanisms as part of the breach management scenarios (provider, reactive monitoring link)
- Adapting the platform configuration through e.g. workload manager interfaces to provide more capacity to avoid future breaches (provider, reactive/proactive monitoring link)
- Deploying a new server and making platform configurations to include (provider, reactive monitoring link)
- Reselecting or renegotiating the SLA (consumer and or provider, reactive monitoring link)
- Automatically or semi-automatically redesigning the process tasks (provider, reactive monitoring link)
- Forcing the virtual organization to undergo an evolution to replace the misbehaving member with another one (provider, reactive monitoring link)
- Making monetary compensation based on the sanctioning clauses of the SLA and continuing business as usual (provider, reactive monitoring link)
- Reducing the reputation of the misbehaving member and continuing business as usual (consumer and/or provider, reactive monitoring link)

Few issues are worth noting. First, participation of other roles depends on the mechanism. Secondly, the mechanisms above assume the relationship between consumer and provider still remains valid. Alternatively the consumer may always decide to switch providers (which puts additional strain to multiparty negotiation member selection). Third, in case of failure due active coordinator node failure (i.e. service aggregator, virtual organization coordinator), many of the approaches are void. In this case possibly reliable messaging and local node self-healing and self-management mechanisms could be utilized for avoidance of unnecessary breach management.

As indicated previously, SLA bonus management could provide additional monetary or reputation bonuses based on over-performance of member. If no bonus management is utilized, degradation of service is a provider option, though is suitable only in completely automated services.

SLA reporting in all likelihood needs to provide both operational reporting and management reporting. This is especially important for next evolutions of workflow systems, which suffered in comparison to ERPs due lack of reporting facilities [27].

V. SLA LANGUAGES

A. Attributes of SLA languages

Based on previous sections following attributes are discussed with each of the SLA languages surveyed:

- Background: Relationship to service and network management as well as software and systems engineering. Provides the context for the particular language.
- SLM infrastructure or tool support: Relationship to particular SLM infrastructure (discussed in section VI).
- Main concepts: The main concepts which are utilized in the language without going into details (i.e. XML element level descriptions).
- Attachment mechanism: Description of how the SLA description is attached to the particular WSDL description, how UDDI is supported.
- Reusability: Describes means of reusability of SLA descriptions, whether based on structure or attachment mechanism.
- Selection and/or negotiation mechanism: Description of whether the language design implies selecting a particular class of service or enables negotiation of a customized service level agreement (or possibly both). Also notes if multiparty agreements are explicitly supported in addition to bilateral agreements.
- Formality: Description of level of formality. Also describes the possible language heritage of the descriptions and support for validation of the SLAs.
- Composability: Description of how composition of web services is supported. If multiple services have an SLA mechanism and a composed service is created based on these, the composed service needs to take into account the existing SLAs when defining its own.
- Breach management and billing: Description of how breach management and service billing is integrated to the language design, if defined.
- Independence of eContracting: Description of how the language ties to particular eContracting languages, if defined.
- Dependency expressions: Description of a dependency mechanism or priority mechanism, if defined. More likely in the case of languages approaching QoS in the wide context (i.e. NFA descriptions).
- Extensibility: Description of how the language is extended to areas that were not in the initial scope of the SLA content.

B. SLAng

SLAng [28] [29] was developed in University College London by deriving SLA requirements from real-world SLAs. For now, SLAng approaches SLAs from service management perspective, focusing on performance metrics and automation of system management, a subset of service management. No implementations using SLAng were found during research for the paper.

SLAng main concepts are SLA metrics and SLA categories. SLA metrics are part of the SLAng definition. The exact metrics depend on the domain of SLA. For application service provider (ASP) domain, metrics are categorized to four QoS characteristic groups: service backup, service monitoring, client performance and operational QoS characteristics. SLA metrics are valid during a schedule defining a contract period.

SLAs are categorized as vertical and horizontal SLAs.
Vertical SLAs identify different parts of a web service platform in order to establish internal SLAs between them. This is intended to enforce behavior with network elements, databases, middleware and application servers. Vertical SLAs include communication SLA (between network element and host OS), hosting SLAs (between host OS and application server), persistence SLAs (between host OS and database) and application SLAs (between web services and application servers). To establish SLAs between “same layer” elements (i.e. to describe horizontal dependencies), horizontal SLAs are used. Horizontal SLAs include networking SLAs (between network elements), container SLAs (between application servers) and service SLAs (between web services).

SLAng does not focus on a runtime selection or negotiation behavior, rather setting design time validation as the main goal. This also excludes it from having an attachment mechanism to WSDL: QoS is modeled as part of the application in web services consumer and producer application logic. The approach is not altogether new given that UML Profiles for QoS have been defined by OMG [30]. Use of this for QoS modeling has been discussed by e.g. Pataricza, Balogh and Gönçzy for both QoS performance and fault tolerance modeling, validation and evaluation [31].

However, SLAng designers correctly note that in order to support validation from type systems perspective, a number of extensions are required beyond application QoS modeling. They advocate using UML Profiles to model SLAng SLA metrics, participants, participant behavior and SLA metrics. SLAng constraints that define the service level objectives (SLOs) are formally defined using Object Constraint Language. Currently available actual formal definitions limits to defining ASP reference model and behavioral model.

Although the focus is on design time validation, researchers behind SLAng are proponents for MDA-based approach. Therefore, future work may include incorporating the constraints to applications through code generation for runtime evaluation. Also, in theory, the added benefit is that, with XML, all the UML models could be defined with XML and, coupled with SLAng SLAs, XMlt can be used to define transformations from formal descriptions to a human-friendly business contract and SLA document.

SLA metrics, categories and MDA-approach provides a view to the design principles behind SLAng usage in ASP domain: first the system management environment is spliced to elements. After this, each of their QoS characteristic groups and SLA metrics defined. This is followed by relationship definition. The assumption is that after this, one can (i) validate that there are no mismatches and (ii) potentially incorporate the QoS to applications.

SLAng contains no support for breach and bonus management and billing. Likewise reuse of SLAs is not within SLAng scope. Also, there is no dependency expression between different types of SLA metrics, only between different types of participants.

In terms of eContracting, SLAng is seen as the main mechanism to extend BPEL all the way to eContracting requirements. However, given that the language has to be extended to other domains beyond ASP and lacks breach and bonus management support, this seems problematic.

### C. Web Service Level Agreement (WSLA)

Web Services Level Agreement (WSLA) [32] [33] has been developed and prototyped by IBM during 2000-2003. WSLA perceives SLAs for web services from a service management perspective with narrow scope, implicitly focusing on providing a customized SLA containing such as response time, availability and throughput. WSLA is utilized in e.g. TrustCoM and Web Services Management Network.

Main concepts of WSLA SLAs are parties, service definition and obligations. These are utilized in WSLA templates and contracts, although neither of the terms is part of the WSLA definition. Parties define the signing parties (web services consumer and provider) and supporting parties (third parties). Third parties include measurement (i.e. monitoring) providers, condition evaluators and management providers (i.e. breach management handlers). The different participating parties enable different contract types, related to composition of services. Likewise, although the contract is for two parties, composition of contracts enables multi-party fulfillment of SLA. This also means a contract can be split into multiple sub-contracts.

`Service definition` defines the service (or group of services) and the SLA parameters that relate to it. The SLA parameters support hierarchies. The foundation is based on resource metrics (e.g. SNMP MIB counters), which is collected based on a measurement profile. Multiple resource metrics can be aggregated to a composite metrics according to some function, which is computed based on an interval defined by a schedule. Composite metrics in turn can be either directly mapped or aggregated to SLA parameters which are defined by the web services consumer. The optimal end result would be that a single or group of SLA parameters would reflect a business metric (i.e. monetary valuation of SLA breach). WSLA itself does not define any QoS metrics but provides the XML elements to make the resource-based definitions. It should be noted that while dependencies through aggregation of metrics can be expressed, dependencies between e.g. SLA parameters cannot be expressed.

Obligations provide means to express service level objectives, which define the party responsible, validity period and target values of SLA parameters. Obligations also define action guarantees, which define service management actions (i.e. breach management mechanism) to be done in case SLO is not achieved. Definitions for workload manager resource management and service deployment are examples of management actions. An evaluation event or evaluation schedule provides information on condition evaluation.

WSLA template consists of two parts: first part provides a partially filled contract that defines basic characteristics (e.g. who the parties are). Second part extends the first with an “offering document”, which defines constraints for the template. Constraints for SLA parameters can e.g. define a range or list of acceptable values to limit negotiation. While WSLA templates are used to describe service offering through the negotiation process, they can be reusable in a sense that a base template is used, which is refined in the offering process.

WSLA contracts emulate the technical part of business
contracts. In order to make them legal, a contracting framework utilizing WSLA must provide a separate eContracting mechanism. WSLA contracts contain the SLA parameters and SLOs formed based on the WSLA template offered to the consumer. Contract types depend on parties involved and the contracting framework. This also defines service composition support, which is not limited by the language itself, but can be difficult to implement.

As an example, the following contract types are used in one implementation of WSLA [33]: *offers* are WSLA templates that provider provides to consumer (i.e. they are external SLAs). *Usage contracts* are realized contracts for a particular service by a particular consumer. *Provider contracts* are aggregated SLAs by multiple providers to enable one provider to represent others in a composite service or group of independent services. *Basic contracts* provide the business part outside the scope of WSLA.

WSLA contracts attach to web services by pointing to the WSDL description that defines the services for WSLA contract is created for. No discussion is provided on (i) utilizing WSLA with UDDI directories or (ii) consumer inquiry with a separate management protocol. Presumably latter is to be done with a separate management protocol.

WSLA is not tied to a particular eContracting language or mechanism and can be used to supplement basic contract definitions. However, the underlying assumption is that the business metrics can be defined by the consumer based on SLA parameters.

Overall WSLA provides means for expressing what is measured, by whom and how. It also defines means to express actions based on breaches. Yet it does not provide information on meaning of any of the third party functions regarding monitoring, evaluation and breach management. These have to be separately defined. These definitions impact the formality of the language: validation of WSLA-enhanced process designs seems problematic even based on the basic language specification. Likewise, clearly a comprehensive support infrastructure is required to provide a suitable support for applications that wish to utilize WSLA.

D. Web Services Offering Language

Web Services Offering Language (WSOL) [34] has been developed and prototyped in Ottawa-Carlton Institute of Electrical and Computer Engineering during 2001-2005. WSOL perceives QoS for web services from a networking perspective, extending this with design by contract – concepts. However, implicitly the focus is on describing performance metrics. WSOL is utilized in Web Services Offerings Infrastructure.

Main concepts of WSOL include the service offering, constraints, management statements and reusability elements. *Service offerings* utilize a class of service –approach, i.e. offerings (SLAs) describe different levels of service for the web services consumer to select from. No negotiation mechanism is possible for either customization of SLAs or bidding in case multiple parties provide the same service offer on an open market. The service offerings reusability is done through service offering items, i.e. constraints, management statements and reusability elements.

*Constraints* express evaluated conditions, which can be behavioral, QoS and access related. *Behavioral constraints* enable pre- and post-condition and invariant expressions. Also “future-conditions” are expressible, i.e. conditions that surface after some specific amount of time has passed from the service request. *QoS constraints* describe QoS metrics and the monitoring entity. QoS metrics themselves are defined by an external ontology. QoS metrics are evaluated with each service request. Alternatively, “periodic QoS” can be expressed, whereby evaluation is done to random requests. Only the average of evaluation is expressed. *Access rights* are related to e.g. service hours and number of invocations.

While overall the QoS approach seems to fit request-response SOAP message exchange pattern (MEP), use with other SOAP MEPs is not discussed. *Management statements* contain management information for different classes of service. This includes price statements, monetary penalty statements and management responsibility statements. *Price statements* divide to pay-per-use and subscription payments. *Pay-per-use payments* supports default price and grouping of operations to limit service use and subscription payments. *Subscription payments* are intended to support time-based billing. The payment statements are separate XML schemas, alternative models, such as volume pricing could be defined as an alternative XML schema. *Monetary penalty statements* are the only supported breach management mechanism currently in WSOL. WSOL implicit assumes management parties will send notifications [, pp. 91]. Monetary units are defined in an external ontology. *Management responsibility statements* specify role responsibilities for particular constraints, supporting third trusted parties. No link to reputation services is provided to evaluate the third parties.

*Reusability elements* are a central enabler in reusing the service offering items. Basically it offers means to reuse service offering items by defining templates and specializing these with parameter definitions. The approach supports specifying different levels (e.g. groups of expressions, individual expressions) of reuse. Likewise “applicability domains” enable scope these in terms of WSDL (e.g. operations, ports, port types and services). Constraints and management statements and reusability elements are all formally specified. Extension with ontologies to enable semantic validation is within scope of the research work.

WSOL descriptions point to the WSDL file describing the operations. WSDL extensions were considered but discarded. No discussion is provided on utilizing WSOL with UDDI directories. WSOL information (i.e. metadata) can be requested with an integrated management protocol.

WSOL provides excellent means for dependency expressions by supporting both static and dynamic relationships. Static relationships are expressed in service offerings themselves. Service offerings can be created, updated or deleted after deployment of service. However, given the performance focus of the design, these are insufficient to accommodate runtime changes to a service that is utilized by a consumer. WSOL uses service offerings dynamic relations (SODRs) as means of runtime adaptation by describing replacement of a particular service offering with another par-
Composition of WSOL service offerings is not currently addressed. This is a problematic area given that the QoS metrics are defined by an external ontology. Some preliminary work has been done in this area, but it has been noted that “implementation of these mechanism to the management infrastructure would not be trivial” [34, pp. 63].

Overall the language design leaves relationship to eContracting open: means for legal binding of SLAs and using WSOL with business protocols remains an open topic, most likely due to the background and scope of investigation.

VI. SUPPORT INFRASTRUCTURES

A. Attributes of support infrastructures

Based on previous sections, the following attributes are discussed with each of the support infrastructure surveyed:

- **Background**: Relationship to service and network management as well as software and systems engineering. Provides the context for the particular support infrastructure and tool.
- **Solution completeness**: Description of what the scope of the solution is in comparison to the static and dynamic phases in the process model.
- **Support infrastructure and tool overview**: Overview of main support infrastructure elements that relate to the frame of reference.
- **Monitoring capabilities**: In case the solution includes monitoring, description of whether this is in- or out-of-band and type (passive, reactive, proactive) of monitoring, in this is defined.
- **Evaluation capabilities**: Description of how monitoring data is evaluated.
- **Reporting capabilities**: Description of to what extent operational and management reporting of the monitoring results is provided, if included.
- **Extensibility and interoperability**: Description of possible means of extending the solution and whether it is intended to be utilized with another solution.

B. Web Services Offering Infrastructure (WSOI)

Web Services Offering Infrastructure (WSOI) [34] [35] has been developed as part of the WSOL investigation discussed previously. Current implementation has not been deployed as part of a larger prototype project.

WSOI is basically a XML parser and SOAP engine extension, which provides an in-band processing on WSOI artifacts.

The **XML parser** provides means to validate WSOL files as well as transform the WSOL service offerings and related files (ontologies, WSDL files) to XML document object model (DOM) trees and separate symbol tables that contain the data required for semantic validation of the WSOL extracted from the ontologies.

While currently not available, the research group is looking to extend this functionality with a **code generator** that provides interceptors (WSOI specific handlers) for QoS measurement, evaluation and accounting. Separate Web Service Offering Descriptors (WSOD) are used to define the chaining order of the interceptors. The parser and code generator are depicted in figure 4.

Regardless of whether the WSOI handlers are manually or automatically created, they are currently in practice Apache Axis handlers. These provide the means for SOAP request, response and error message interception and manipulation in Axis. Through this the WSOI is able to both provide the service offers as well as monitor the service requests and replies in an in-band manner. Figure 5 provides the overview of the architecture.

C. TrustCoM SLA management

TrustCoM is a research project ([www.eu-trustcom.com](http://www.eu-trustcom.com)) carried out in the EU 6th Framework program (Networked Business and Government). The project started in 2004 and is to conclude in the first half of 2007. The project consists of both industry and academic researchers, with focus on holistic evaluation of virtual organization (VO) concept and a
reference implementation of middleware to support VO concepts. TrustCoM WSLA negotiation is done by a separate negotiation protocol (outside the scope of this paper). TrustCoM is also a partial system in the sense that breach management is not discussed although e.g. reputation services and semi-automated process re-design support exists.

The TrustCoM middleware [35] SLA management subsystem can be partitioned among participants to local SLA management services, which contain SLA monitoring and management. These can be provided also by separate trusted third party (TTP, i.e. WSLA supporting participant). Additionally TTP provides SLA evaluator service.

![Figure 6: TrustCoM system architecture [36]](image)

TrustCoM utilizes extended WSLA to enable negotiation, monitoring and SLA breach detection through notification service under TTP notification broker and VO infrastructure services subsystem. Breach management, notification event handling is provided by separate subsystems (e.g. reputation services under Trust and Security Services subsystem and virtual organization reconfiguration under business process (BP) enactment and orchestration subsystem).

SLA is instantiated through a negotiation process initiated by the TTP SLA management service. Currently TrustCoM does not contain negotiation processing logic, rather just exchanging predefined WSLA descriptions.

Local SLA managers utilize the SLA provided to configure local monitoring. Currently monitoring of resources is based on (i) using Windows Management Interface (WMI) for reading Windows performance counters, (ii) Java Management Extensions used in the application services themselves and (iii) GANGLIA monitoring tool available for both Linux and Windows for unified systems monitoring. These are collective called the State Data Provider.

The monitoring data from State Data Provider is handled by the local SLA monitoring service, which calculates the composite metrics, aggregating these to SLA parameters. These are provided to the evaluator based on a schedule (push) or as reply to request (pull).

The TTP SLA evaluator compares the SLA parameters to the WSLA service level objectives. The SLA Evaluator utilizes TrustCoM notification service to publish notifications for both breaches and fulfillments of service, without regard to what is being done with these. Assumption is that the related parties (e.g. reputation service, VO manager) subscribe to SLA Evaluator notifications and handle the events.

These are connected to the VO reconfiguration policies that define breach management actions, enabling limited self-management for the platform.

### D. Web Services Management Network

Web Services Management Network (WSMN) [37] [38] was produced as a pilot project by HP. While based on WSLA as TrustCoM SLA management subsystem, WSMN contains a number of differing notions that provides an interesting comparison in terms of how the solution can differ irrespective of utilization of same language. This is in all likelihood due the loose nature of the WSLA specification and the number of supporting parties (monitoring, evaluation and management providers) that are suggested by the language specification.

WSMN members each have the same capabilities that are used to create an overlay network that communicates with out-of-band messages based on a number of WSMN specific protocols. Each WSMN member represents a different organization, WSMN only focuses on the public processes (i.e. process organizational interfaces). In order to identify particular business protocol process flows, a (separate or business protocol provided) global identifier (GUID) is used by the members.

The WSMN architecture contains an SLA engine, measurement engine, business correlation engine and a message handler for WSMN specific protocols. These are (with the exception of the message handler) depicted in figure 7 along with potential related applications.

![Figure 7: WSMN intermediary [36]](image)

The SLA engine comprises multiple subsystems to support functionality of WSLA. A process controller manages the initiation and flow control. During initiation it provides the SLA customizer to configure an event manager and input the SLOs to an SLA repository. SLA repository also contains SLO validity period information. The validity period of the SLOs defines their evaluation period, which is not necessarily right after the initiation. Event manager is responsible for both managing timed measurements and event-based measurements during SLO validity period. Measurements are pushed to the SLO evaluator, which compares the meas-
measurements to the SLA repository SLO information. SLO breaches are pushed to a separate SLA violation engine that has the required logic for breach management.

Note that compared to WSLA notions of signing and supporting party, all functionality of these parties is provided by a single WSMN member functionality. There are no separate supporting party and signing party implementations (like in TrustCoM).

The business process correlation engine (BPC engine) communicates with a workflow engine API or logs to establish runtime measurements of the public process. BPC engine can also be used manage events to modify to the process execution if an API is available. The HP prototype utilizes HP Process Manager as the workflow engine and its Java API to manipulate execution (e.g. replace misbehaving members).

The measurement engine provides environment measurements providing an overlay for different types of external probes (e.g. SNMP agents, WMI data providers etc).

Beyond measurement and BPC engines, WSMN external information regarding the private process can be gathered from a separate business activity monitor. These establish the different WSLA metrics. All WSMN members have a model of the business process in a separate model repository (not in figure 7) which defines the measurement targets. These are utilized to store the measurement data in an operational database (mySQL database) which is utilized by the SLA engine.

WSMN message handler is responsible for three classes of out-of-band protocols to manage the federation (i.e. partners) and SLAs. Lifecycle protocols are a set of basic partner connectivity and cooperation protocols. Initiation protocols provide means to initialize the management network. The initiation protocol has a hard-coded bootstrap mechanism; secure negotiation protocol problematics are not discussed. Keepalive protocols provide means to monitor WSMN members. Clock synchronization protocol provides time synchronization among WSMN members. Teardown protocol provides means for explicit signoff. Possible use of existing protocols or the details of the protocols are not described in the articles.

Obviously lifecycle management is not an integrated part of the SLA management. E.g. in TrustCoM it is handled by the Business Process and Enactment and Orchestration subsystem in conjunction with other subsystems.

Measurement (exchange) protocols (MEPs) provide means of exchanging measurement information amongst WSMN members. Authors note that although WSLA supports bilateral agreements (i.e. 2 signing parties) it is possible that the web service is related to a third party that is more than a supporting party (e.g. in the case of service provider providing composed services without using choreography to generate a shared view of the public web services interfaces). Their view is that in this case, instead of making internal SLAs and a separate external SLAs or decomposing the SLA to subsets with indicated responsible parties, the SLA measurements should be exchanged with a separate measurement exchange protocol in order to evaluate the SLA.

The MEPs approach seems to be riddled with problems, yet these are not discussed by the authors. These seem to stem from the decoupling of SLA signing parties from the members participating to the execution. This has legal and trust management implications. From legal perspective, if legal binding of contract is established between SLA signing members what is the contractual obligation of WSMN members participating but outside the SLA? From trust perspective, if negotiation protocol would be used to established between parties in SLA, what would consumers use to establish trust to external members outside SLA?

Assurance protocols provide means for management network optimization and integration to interface systems related to management. Negotiation protocols provide WSLA negotiation. Trouble ticket exchange protocol provides means to integrate the WSMN node to trouble ticket systems. Authors note the assurance protocols are outside the focus of their work, details of the protocols are not provided. Utilization of service directories, reputation services and TTP negotiation services to support functionality of this layer is merely mentioned.

WSMN members are explicitly indicated to be SOAP intermediaries. Therefore, additional functionality such as the business correlation engine

WSMN includes a visualization tool which can utilize the WSMN intermediary databases (both in memory and RDBMS) to provide a view to all metrics gathered as part of SLA monitoring.

Overall, whereas TrustCoM focuses to a great extent on virtual organization related specific topics (dynamic reconfiguration, negotiation of partners and impact to SLA), WSMN is focused on integrating workflow systems and providing an exchange point between service management and workflow engine performance.

VII. SUMMARY AND CONCLUSIONS

Web services service level management (SLM) and quality of service (QoS) approaches are a vast area. This is to be expected given the extensive background behind QoS. While the overviews provided in this article touched the most visible projects a number of others remain. However, these seem to be narrower partial solutions than the ones presented. The frame of reference in this article provides means of categorizing the dispersed research projects.

Many of the surveyed languages were comprehensive, implicitly suggesting need for very comprehensive support infrastructures and tools in order to enable full-blown deployment within reasonable effort. The problem of semantic differences of terminology and variance in accuracy of monitoring data was often discarded. It is to be expected that the model-drive development or semantic web services research teams will have a strong foothold on this area (compared to the systems engineering-focused research teams).

The support infrastructures on the other hand often utilized the languages in a limited manner and avoided the challenging areas: aggregation of SLAs was typically sub-sidied by use of independent bilateral agreements and use of UDDI directories was not even discussed. Likewise the background and particular problem scope clearly show in
each: the approaches provide partial solutions to their intended problem scope. The intended use of support infrastructure, i.e. whether it is for service provider internal use or external use, makes a clear separation in terms of required (and thus implemented) functionality, impacting especially breach management mechanisms and reporting features.

VIII. REFERENCES

Main references

[8] OCG, IT Infrastructure Library Service Delivery, OCG 2000
[12] Clemens Szyperski, Component Software: Beyond Object-oriented programming, November 2002
[34] Vladimir Tosis, Service Offerings for XML Web Services and Their Management Applications, PhD Thesis, Carleton University, Department of Systems and Computer Engineering, August 2004