Service level agreement management in federated virtual organizations

Tuomas Nurmela[1] and Lea Kutvonen[2]

[1] Tietoenator Processing & Network Oy, Espoo, Finland
Tuomas.Nurmela@tietoenator.com

[2] Department of Computer Science, University of Helsinki, Finland
Lea.Kutvonen@cs.helsinki.fi

Abstract. The present emergence of loosely-coupled, inter-enterprise collaboration, i.e., virtual organizations calls for new kind of middleware: generic, common facilities for managing contract-governed collaborations and the autonomous business services between which those collaborations are formed. While further work is still needed on the functional governance of the collaborations and services, even more work is awaiting on the management of non-functional aspects of the virtual enterprises and their members. In this paper, languages and architectures for service level agreement between Web Services are discussed and the maturity of the service level management solutions is reflected against the needs of federated virtual organizations.

Keywords: virtual organizations, Web Services, service level agreements

1 Introduction

The present emergence of loosely-coupled, inter-enterprise collaboration, i.e., virtual organizations calls for new kind of middleware: generic, common facilities for managing contract-governed collaborations and the autonomous business services between which those collaborations are formed. These facilities are required to manage the collaboration lifecycle and interoperability at technical, semantic and pragmatic levels. We call these facilities B2B middleware [1, 2].

While further work is still needed on the functional governance of the collaborations and services, even more work is awaiting on the management of non-functional aspects of the virtual enterprises and their members. In the category of non-functional aspects three types of phenomenon can be seen: 1) policies and business rules that determine pragmatic decision between alternative business processes or collaborations, 2) private decision-making rules, for example determining trust relationships or quality of service level satisfaction, that have effect on the collaboration memberships (or in breach recovery actions at the collaboration level), and 3) non-functional aspects related to communication between business services, including security, QoS, or other selectable transparencies of the abstract communication channel.

As part of the work on refining the non-functional aspect management in federated virtual organizations and in the Pilarcos architecture [2], we have separately studied sub-architectures for multiparty eContracting [2], binding between peers by federated, open channels, and trust management [1]. To complement this theme, the present paper studies the management of service level agreements, associated either to the communication architecture, or more interestingly to the quality of peer services in the collaboration. The study addresses adaptation to changes either at the organizational, local level, or in the operational environment of the services by different type of runtime agreements on the service level. The present trend on service level management enables the service markets to move from basic cost-competition towards differentiation through variation of service capabilities.

Service level management (SLM) [6] is the business process that contains all the activities relating to *service level agreements* (SLAs, formally negotiated contracts) 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. SLM also contains the notion of reporting the results to the customer. This business-centric approach can be seen as 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 process [2, 8]. However, the difference lies in the scope: in open, dynamic environments, eContracting is required to negotiate and agree 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 SLA commitments.

The practical service level management approach complements the present work on extended service-oriented architectures (SOA) [3, 4], also taking into account adaptation to heterogeneity and autonomy of partners [5]. On implementation level, different research initiatives on Web Services QoS have approached the issue from both performance-perspective and from non-functional aspects (NFA) perspective in general. The approaches focus either on model-driven development (MDD) or policy-expressions or runtime service management.

In this paper, languages and architectures for service level agreement between Web Services are discussed and the maturity of the service level management solutions is reflected against the needs of federated virtual organizations. After presenting a frame of reference in Section 2, the paper surveys Web Service languages that focus on the performance-perspective and in particular include service level agreements (SLAs) and reflect the various architectures behind their development and the SLM phase for which they support in Section 3. The maturity and sufficiency of these approaches, reflected against the Pilarcos architecture design principles, conclude the paper.

2 Service level management

The discussion of service level management is dependent on the type and scope of agreements as well as the agreement management lifecycle. In terms of different *types of SLAs* [6], service providers typically create both internal SLAs and external SLAs. *Internal SLAs* define the requirements between service producers. *Operational Level Agreements (OLAs)* codify what is expected of different units within the service provider company that offers the service to customers. If the service provider utilizes a third party as sub-contractor to provide the service, an *underpinning contract (UC)* is created between the third party and the provider. *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. SLAs contain among other things *SLA parameters* (e.g. availability), with each having a service level objective (SLO), i.e. target value for the given SLA parameter.

The different types of SLAs relate especially to organizational form, i.e. whether the virtual organization is a temporary organizational structure like a consortium or a more permanent structure such as a partnership [7]. The virtual organization in practice requires means of either aggregating the SLAs to determine the composite SLA for the whole service (offers-based approach) or using the external SLA in the contractual agreement with the customer to make negotiation demands on the potential members in the virtual organization (reverse-auctioning approach). The latter assumes the service provider either takes the risk that fulfillment of service is not really possible or uses an already existing virtual breeding environment as the basis for negotiation, without having negotiated the details with participating members.

Alternatively service providers could approach the issue as a risk management scenario and include SLA breach-related monetary compensation to service pricing without regard to actual requirements. However, intuitively this does not lead to long customer relationships given that customer probably cannot negotiate the actual financial loss as part of the breach management payoff.

SLA contract scope needs to be considered in addition to considering the different roles that may be related to producing the service. The SLAs can either deal with technical metrics or it can deal with business metrics as part of the eContract. Ideally the technical metrics can be aggregated to business metrics. Yet the business metrics are domain dependent. Therefore, the mapping is problematic.

Figure 1 describes a suggestion for minimal content in regard to different types of SLA and eContracting. Possibility for separation of SLA management from the eContracts provides benefits in terms of reuse and breadth of situations to which the language can be applied. The separation of technical metrics from business metrics supports system modularity. It also supports specification of third party roles in order to manage a specific area of responsibility (e.g. monitoring and evaluation of purely technical SLA parameters). This approach would benefit from indicating dependencies between different metric types.

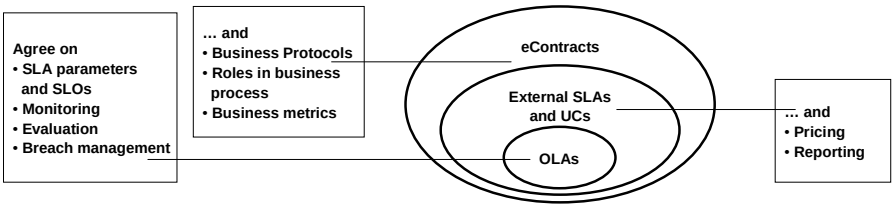

Figure 1: Minimal scope of contract content from SLM perspective.

In addition to the contents of the agreements and the scope of content amongst the involved parties, the service level management lifecycle has to be determined. In the following, the steps of template design, SLA-enhanced process design, negotiation and selection, monitoring, evaluation, breach and bonus management and reporting are identified. The lifecycle is captured in Figure 2. This is loosely based on the ITIL SLM process description [6] and the eContracting process [8].

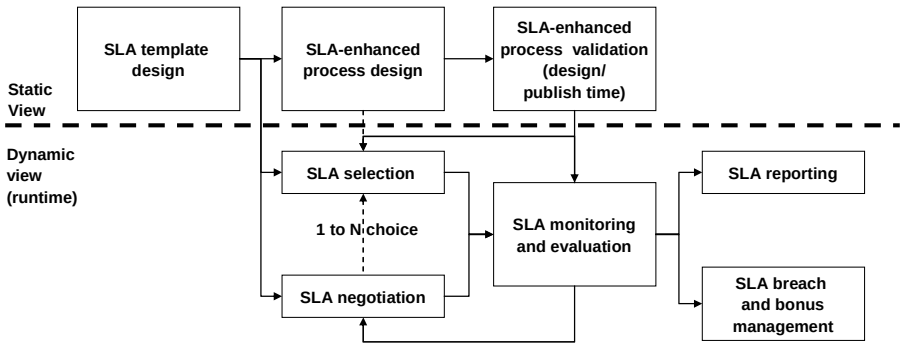

Figure 2: Frame of reference for SLM.

The SLA template design consists of defining the SLA elements, for example in XML. If the SLA is to be negotiated, SLOs are dynamically established. Only SLA parameters and parameter boundaries need to be defined. Alternatively, if a class of service–approach is used, classes need to be defined. This means defining the SLA parameters and the SLOs prior to offer of the service. The class of service approach is beneficial in the sense that possible conflicting technical demands (e.g. minimal latency but assured delivery) can be screened and will not need runtime resolution. However, because customer specific requirements cannot be matched, it fits better to environments focusing only on technical metrics. The template design is particularly impacted by the SLA language design choices.

The 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 supports SLAs.

After creation of the process, the SLA-enhanced process design may be *validated* at design time or the time of publishing a Web Service. 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 the case where services are provided in an open market, it is possible that the Web Service consumer participates in an auction for the best possible Web Service. This would require a negotiation mechanism with support for multiparty negotiation. Alternative approaches include the capability to select an identical service from each service provider and only provide payment for the fastest [9]. In addition, the offered services can be provider resource-constrained. In this case the negotiation may be may revolve around multiple consumers competing in an auction for single provider resources.

As can be seen, the SLA determination can be modeled as a full-blown auction or bargaining scenario. However, this is typically not required in practice, because of SLA having limited scope. Likewise, the negotiation can be separated under a separate negotiation protocol.

The monitoring of SLA parameters contains at least two issues. First, the monitoring can be done either in-band or out-of-band. Second, the link between monitoring and evaluation can be passive, reactive or proactive [8]. *Out-of-band monitoring*, following a typical probe-approach, is suitable for performance metrics. *In-band monitoring* on the other hand can be located on the service host providing host or on a separate tier consisting of e.g. access control, message routing and XML firewall protecting the service. Especially non-performance based metrics utilize in-band monitoring. *Passive monitoring link* merely refers to logging monitoring data at run time. Evaluation is done later as a separate action. *Reactive monitoring link* provides the means for evaluation of SLO breaches for corrective actions. *Proactive monitoring link* would support the use of internal 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.

The evaluation of SLOs can be based on different *modes*, being *event-based* (with e.g. schedules) or *request-based.* Likewise, it can support *complete evaluation* (i.e. utilize all available monitoring data) or *statistical evaluation* (i.e. evaluate only a sample of monitoring data). *Evaluation accuracy* is dependent on the monitoring data sources: for an example, if availability data source consists of trouble tickets, a human element is involved. On the other hand, in case of an end-to-end polling, frequency of polling denotes the accuracy.

The SLA breach management governs SLO or proactive threshold breaches, i.e. it is closely tied to the monitoring link. For example, with passive monitoring link, breach management is typically done a posteriori by people. While little research on automated breach management is available, intuitively this is done by consumer and/or provider. Not all possible mechanisms fit the different monitoring link types (reactive or proactive). Intuitively, a number of mechanisms are possible, including the following:

- Using long-running transactions and their compensation mechanisms as part of the breach management scenarios (provider, reactive monitoring link).
- Reselecting the class of service 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).

Some additional mechanisms may be possible for systems considering only technical metrics such as adapting platform configuration through workload managers or deploying new servers or deploying new servers.

Few issues are worth noting. First, participation of other third party roles depends on the mechanism. Secondly, the mechanisms above assume the relationship between consumer and provider still remains valid. Alternatively the consumer may decide to switch provider. 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.

SLA bonus management could provide additional monetary or reputation bonuses based on over-performance of a member. If no bonus management is utilized, degradation of service is a provider option, though this 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 the next evolutions of workflow systems, which suffered in comparison to ERPs due to lack of reporting facilities [10].

3 SLA languages and SLM architectures

In the following, examples of different types of SLA languages and SLA architectures behind them are discussed. The goal of the survey was to find existing candidates for the SLA templates, negotiation and monitoring, as well as SLA post-processing in federated virtual organizations. As the technical environment, the Pilarcos architecture [1, 2] was used with the following points of interest.

The Pilarcos architecture provides for both the static and dynamic views of SLM (see Figure 2). For the static view, service type definitions include attributes that form part of the SLA template; other parts can be derived from the business network model defining the topology of the collaboration providing the composite service in question. For the dynamic view, each service provider registers its service offer that contains the service interface description (including a process description) and its service level offers and requirements that can be used in the selection and negotiation phases. The negotiation is performed partially by a populator agent, that takes a suggested business network model (defined in terms of service roles, interactions between them, and nonfunctional requirements to be jointly filled by the collaboration) and imports matching service offers to it. Further, the negotiation continues by allowing each potential partner to review the proposed collaboration

structure and conditions gathered to the eContract. In this phase, privately held motivations for decision-making and preferences take effect, for example, trust-based decisions can determine what kind of policy values become accepted, or whether a collaboration is entered at all. For monitoring purposes there are two sources of NFA-related rules. First, from the business network model itself, monitoring rules for business-related aspects can be gathered – these can be expressed either in terms of business concepts, associated to processes and thus multiple services at the same time, or in terms of technical concepts in cases where no translation between business concepts and technical concepts exist. Second, as a result of the negotiations, for each role there is an associated service and functional and non-functional requirements placed on that service alone.

Beyond the languages surveyed in this paper a number of others exist, including those in the semantic Web Service arena (e.g. WSML/WSMO QoS extension [18]) and eContracting languages and systems extensions, such as Laura [19] extending ebXML.

3.1 SLAng

SLAng [11, 12] was developed in University College London by deriving SLA requirements from real world SLAs. SLAng approaches SLAs from service management perspective, focusing on performance metrics and automation of system management, a subset of service management. It focuses on utilization of SLAs in support of model-driven development. No implementations using SLAng were found during research for the paper.

SLAng main concepts are SLA metrics, SLA categories and responsibilities. *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, which defines the contract period.

SLAs categories divide to 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 communications 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 applications servers).

Horizontal SLAs are used to establish SLAs between “same layer” elements (i.e. to describe horizontal dependencies). Horizontal SLAs include networking SLAs (between network elements), container SLAs (between application servers) and service SLAs (between Web Services).

Responsibilities enable description of individual and mutual commitments. *Client* and *server responsibilities* describe individual commitments. The approach supports different WSDL message exchange patterns on service SLA level and enables inter-composition of SLAs to take into account requirements on both members. *Mutual*

responsibilities are responsibilities that both members have agreed to. These can be established with a separate negotiation mechanism. Mutual responsibilities can be used to describe the compensation for a given SLO breach. Different types of compensation descriptions are not yet part of SLAng.

SLAng focuses on complementing an abstract description of the behavioural model of the service. Therefore, QoS is modeled as part of the application in Web Services consumer and producer behaviour. The approach is supported by UML Profiles for QoS have been defined by OMG [13]. Use of this for QoS modeling has been discussed also by Paticza, Balogh and Gönczy for both QoS performance and fault tolerance modeling, validation and evaluation [14].

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 and UML Profiles to model SLAng SLA metrics, participants and participant behavior and defining SLAng constraints that define the service level objectives through Object Constraint Language (OCL). Currently available actual formal definitions limits to defining ASP reference model.

Researchers behind SLAng are proponents for MDD-based approach. SLAng approach is for both design time validation support especially intra-service SLA and monitoring and evaluation of runtime behavior between negotiated SLAs. Inter-service SLA composition is also noted. However, much of this seems to be still in the works as future work noted includes service composition and analysis toolkit and incorporating the constraints to applications through code generation for runtime evaluation. Likewise, the lack of negotiation mechanism description would indicate that the issue is not currently addressed. Additional work noted includes transformations from formal descriptions to a human-friendly business contract and SLA document.

SLA metrics, categories and an MDD-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) incorporate the behavioral constraints to applications.

SLAng contains no support for breach and bonus management or service pricing. These, with addition of reuse through SLA templates are also considered part of future work for SLAng. Lack of dependency expression between different types of SLA metrics is not addressed.

In terms of eContracting, SLAng is seen as the main mechanism to complement BPEL with behavioral model 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, the current approach seems insufficient for virtual organization requirements.

3.2 Web Services Level Agreement

Web Services Level Agreement (WSLA) [15, 16] 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 currently utilized in TrustCoM. TrustCoM [20] focuses on enabling dynamic virtual organizations through inclusion of security, trust relationships and contracts. The SLA management subsystem is partitioned among participants. It includes *local SLA management services*, which contain SLA monitoring and management and a *separate third party SLA evaluator service* for actual SLA evaluation. This uses the notification infrastructure to inform of violations, without regard to the actual breach management mechanism. A separate negotiation mechanism is used to establish the SLAs.

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 Service 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 directive*. 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 can be either directly mapped or aggregated to *SLA parameters* which are defined by the Web Services consumer. SLA itself is established through a separate negotiation mechanism outside the scope of WSLA. The optimal end result would be that a single or group of SLA parameters would reflect a business metric for the Web Service consumer. 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 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, although these are not defined in WSLA. An evaluation event or evaluation schedule provides information on evaluation condition.

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 “offer document”, which defines constraints for the template SLA

parameters. For an example, constraints can be used to define a range or list of acceptable values for an SLA parameter to limit negotiation. While WSLA templates are used to describe service offer through the negotiation process, they can be reusable in a sense that a base template is used, which is only refined in the negotiation 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 [16]: *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 contract 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 utilizing WSLA with UDDI directories, or consumer inquiry of WSLA composite metrics without requesting actual service (i.e. metadata exchange). 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 Web Service consumer based on SLA parameters.

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.

3.3 Web Services Offerings Language

Web Services Offerings Language (WSOL) [17] 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 (WSOI). WSOI is basically an XML parser for checking WSOL

definition syntax correctness and a SOAP engine extension, which provides an in-band monitoring and evaluation by using WSOI handlers for interception. Future work includes WSOL code generator to create WSOI handlers from WSOL definitions.

Main concepts of WSOL include the service offerings, constraints and management statements. These are supported by reusability elements and service offering dynamic relationships. *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 can be related to service hours and number of invocations.

While overall the QoS approach seems to fit request-response WSDL message exchange pattern (MEP), use with other WSDL MEPs are 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. The pay-per-use payment supports default price and grouping of operations to limit definition length. 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 implicitly assumes management parties will send notifications [17, 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 provides 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 scoping these in terms of WSDL. Constraints, management statements and reusability elements are formally specified in UML. Extension with ontologies to enable semantic validation is within scope of the ongoing 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 a 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 offering dynamic relations* (SODRs) as means of runtime adaptation by describing replacement of a particular service offering with another particular service offering in case of a particular constraint violation.

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" [17, 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, possibly due to the background and scope of investigation.

3.4 Summary

In the survey, special attention was given on properties related to potential for composing service and their SLA notions, whether the language was designed for the static or dynamic environments, and their relationship to eContract structures. The SLA languages are summarized in Table 1.

We note that at its current state SLAng is designed for development time descriptions and, on service SLA level, is used to complement BPEL by expressing behavioral constraints. On the other hand, WSLA and WSOL focus on runtime support in terms of negotiation or selection and evaluation of offers. However their relationship to eContracting is different. WSLA assumes that Web Services consumer can establish relationship to business metrics based on providers technical metrics, whereas WSOL simply focuses on technical metrics without regard to eContracting.

4 Conclusions

Taking the reviews and the frame of reference into account the presented languages all provide good approaches in specific areas. In particular, the SLAng level of formality and client requirements provide support for design validation and service inter-composition. This is in-line with populator requirements. Second, WSLA provides a comprehensive conceptual frame and does not limit to particular metrics even though it lacks means to express support of runtime dynamism. Third, the use of WSLA in TrustCoM shows that modularity is achievable, potentially supporting separation of evaluation and breach management mechanisms from local

Attribute	SLAng	WSLA	WSOL
Background and approach	Service management, Model-driven development	Service management, Runtime support infrastructure	Network QoS, Runtime support infrastructure
SLM infrastructure or toolset for language	Unknown	TrustCoM	Web Services Offerings Infrastructure (WSOI)
Main concepts	(Domain-specific) SLA metrics, SLA categories, responsibilities	Parties, service definition, obligations	Service offerings (SOs), constraints, management statements
SLA verification	Design-time validation and run-time evaluation	Run-time evaluation	Run-time evaluation
Association mechanism to service descriptions and service offers	Behavioural model	SLA points to WSDL	Service offering points to WSDL
Reusability	None currently	WSLA templates	Reusability elements
Denotations and formal background	UML, UML profiles and OCL	UML	UML
Composition support for aggregated services	Intra-composition and inter-composition based on conformance	Not constrained by language, depends on contracting	Not constrained by language, seen as problematic
Selection or negotiation mechanisms and multiparty aggregations	None currently, separate negotiation protocol intended	Separate negotiation protocol, custom SLAs	Selection, predefined classes of service
Pricing support	None currently	None currently	Yes, in management statements
Breach management support	None currently	Yes, in action guarantees	Yes, in management statements
Dependency expressions between SLAs and SLOs	None	None	SO dynamic relationships
Relationship to eContracting	Used with BPEL	Aggregation of technical metrics to business metrics	Independent of eContracting

Table 1: Comparison of Web-Services –related SLA-language initiatives

monitoring. Finally, the WSOL service offering dynamic relationships provide means of pre-defining runtime support for autonomous service adaptation.

In general, further development is needed on languages that provide better support for NFA-related QoS beyond communications and technical QoS, support composition of service offers, and allow expressions of monitoring rules to complement the associated service level requirements.

As a conclusion, there is need for further developing a family of aspect languages for NFAs with a number of requirements: Each language should have a sufficient set of joint basic concepts so that aggregations can be negotiated over them in a sensible way. Consequently, each broad category of business services has a separate set of concepts and related metrics, so that these are understandable to the business process designers in business terms. At the more technical level, it is required that each concept and metrics has a supported transformation to technical terms in a transparent way. Also, it is necessary that the technical level concepts and metrics are provided for communication service business.

References

1. Lea Kutvonen, Toni Ruokolainen, and Janne Metso, ”Interoperability middleware for federated business services in web-Pilarcos”, International Journal of Enterprise Information Systems, 3(1):1-21, January 2007
2. Lea Kutvonen, Janne Metso, and Sini Ruohomaa. From trading to eCommunity population: Responding to social and contractual challenges. In *Proceedings of the 10th IEEE International EDOC Conference (EDOC 2006)*, Hong Kong, October 2006.
3. Mike P. Papazoglou, “Service oriented computing: concepts, characteristics and directions”, In *4th International Conference on Web Information Systems Engineering (WISE'03)*, 2003.
4. M. P. Papazoglou, D. Georgakopoulos, “Service oriented computing”, Communications of the ACM, Vol 46, Issue 10, 2003, pp. 25-27.
5. Mundimar P. Singh, Michael N. Huhns, *Service-Oriented Computing: Sematincs, Processes, Agents*, John Wiley & Sons, 2005.
6. OCG, *ITIL Service Delivery*, The Stationary Office, 2001.
7. Luis M. Camarinha-Matos and Hamideh Afsarmanesh, ”Virtual Enterprise Modeling and Support Infrastructures:Applying Multi-agent System Approaches” in M. Luck et al (eds), *Multi-Agent Systems and Applications*, ACAI 2001, LNAI 2086, 2001, pp. 335-364.
8. Z. Milosevic, A. Berry, A. Bond, K. Raymond, "Supporting business contracts in open distributed systems," In *2nd International Workshop on Services in Distributed and Networked Environments*, 1995.
9. Heiko Ludwig, “Web Services QoS: External SLAs and Internal Policies, Or: How do we deliver what we promise?” IBM research center report, 2003.
10. Jorge Cardosa, Robert M. Bostrom, Amith Sheth, “Workflow Management Systems and ERP Systems: Differences, Commonalities, and Applications”, Kluwer, Information Technology and Management 5, 2004, pp.319-338.
11. James Skene, D. Davide Lamanna, Wolfgang Emmerich, “Precise Service Level Agreements”, In *26th International Conference on Software Engineering (ICSE'04)*, 2004.
12. D. Lamanna, J. Skene, W. Emmerich, “SLAng: A Language for Defining Service Level Agreements”, In *Proc. of the 9th IEEE Workshop on Future Trends in Distributed Computing Systems, FTDCS 2003 (Puerto Rico, May 2003)*, 2003.
13. Object Management Group (OMG), *UML profile for quality of service and fault tolerance characteristics and metrics*, 2004.
14. András Patarizca, András Balogh, Lázló Göczy, “Verification and validation of Nonfunctional aspects in Enterprise modeling”, in Peter Rittgen (ed), *Enterprise Modeling and Computing with UML*, Idea Group, November 2006, pp. 261-303.
15. Heiko Ludwig et al., ”Web Service Level Agreement (WSLA) Language Specification”, Version 1.0, revision wsla-2003/01/28. available from: www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf
16. A. Dan et al. ”Web Services on demand: WSLA-driven automated management”, IBM systems journal, Vol. 43, issue 1, 2004, pp. 136-158.
17. Vladimir Tosic, *Service Offerings for XML Web Services and Their Management Applications*, PhD Thesis, Carleton University, Department of Systems and Computer Engineering, August 2004.
18. Ioan Toma, Douglas Foxvog, Michael C. Jaeger, “Modelling QoS characteristics in WSMO”, In *Proceedings of the 1st workshop on Middleware for Service Oriented Computing (MW4SOC 2006)*, Australia Nov. 27 – Dec. 01, 2006.
19. Adomas Svirskas, Bob Roberts, “Towards business QoS in Virtual Organizations through SLA and ebXML”. In *10th ISPE International Conference on concurrent engineering: Research and Applications*, 2003.
20. TrustCoM, “TrustCoM Reference Architecture”, Version 1, Deliverable D09, Work package 27, 14.8.2005.