Decision Support Systems 53 (2012) 234–244

Contents lists available at SciVerse ScienceDirect

Decision Support Systems

journal homepage: www.elsevier.com/locate/dss

Workflow composition of service level agreements for web services ☆

M. Brian Blake [a,*], David J. Cummings [b], Ajay Bansal [c], Srividya Kona Bansal [c]

[a] Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN, United States
[b] Department of Computer Science, Stanford University, United States
[c] Department of Engineering, Arizona State University, United States

ARTICLE INFO

Article history:
Received 19 July 2010
Received in revised form 22 December 2011
Accepted 30 January 2012
Available online 8 February 2012

Keywords:
Service level agreements
Quality of service
Web services
Service-oriented computing

ABSTRACT

Service-oriented architecture enables an environment where businesses can expose services for use by their collaborators and their peer organizations. In this dynamic environment, organizations require the use of service level agreements (SLAs) to assure the quality of service (QoS) standards of services provided by their collaborators. In an ad-hoc workflow scenario, a business may need to perform real-time composition of existing services in response to consumer requests. In this work, we suggest that, in parallel to traditional web service composition, the business must also compose the existing SLAs in order to ensure the service levels that must be guaranteed to new consumers. Ultimately, this approach to SLA composition must align with the overarching principles of the provider and the priorities of the consumer. In this paper, we introduce a model and representations of service level agreement attributes appropriate for managing a service provider's expectations when adding new partners. Our evaluations suggest that the SLA composition can efficiently run concurrently with traditional service composition.

1. Introduction

A service level agreement, *SLA*, is a technical contract between two types of businesses, producers and consumers. A SLA captures the agreed-upon terms between organizations with respect to quality of service (QoS) and other related concerns. In simple cases, one consumer forms a SLA with a producer. In more complex cases, a consumer may form a SLA that defines a set of producer businesses. Considering a service-oriented computing environment, capabilities are shared via the implementation of web services exposed by a producer organization. The ultimate goal of service-oriented computing is for consumers to access these shared capabilities on-demand. As such, in cases where businesses have longstanding relationships, such as workflow and supply chain environments, peer companies that share services must be able to assure a level of service to their underlying customers [4,8].

New specifications, such as the Web Service Level Agreement (WSLA) and Web Service Agreement (WS-Agreement) [2] enable SLAs to be associated with an individual web service or even groups of web services. These specifications define an eXtensible Markup Language (XML)-based data model that can be used along with the Web Service Description Language (WSDL) documents that traditionally describe the web services. These specifications provide a significant opportunity. Organizations can specify QoS-related concerns in concert with the functionality concerns already captured in the WSDL files. As a result, when a new organization searches for a pertinent web service, the SLA-enhanced WSDL file can be used to determine the appropriateness of the service to meet the required business need. Furthermore organizations can use the SLA-enhanced WSDL file to negotiate the QoS terms.

Although these SLA technologies and specifications present new opportunities for service-oriented business processes, there are a number of significant barriers. When a consumer organization must create a new business capability that requires the workflow composition of multiple web services, then that organization will also need to understand the composite impact of the underlying SLAs. Consequently, in addition to composing web services that are functionally compatible, the organization will need to ensure that the web services are compatible with regard to their service levels. Also, the product of all the SLAs for a composition of web services must be within the required threshold of *feasibility* as defined by the end users. As the service-oriented computing paradigm increases in popularity, the consumer will have the option of many similar services that may meet a particular requirement. As such, the composition of web services that is most efficient for a particular business purpose will rest on the organization's ability to understand and optimize the corresponding composition of SLAs.

To deal with the aforementioned issues, we introduce the phrase, *workflow composition of SLAs*. Our approach suggests the multi-dimensional evaluation of existing agreed-upon QoS standards in order to predict the standards possible for the introduction of new agreements. While the notions of multi-dimensional analysis, optimization, dynamic programming are not new [9,10,12,17,35] in this

☆ This paper is a substantial extension of earlier work presented in [7] and [5].

* Corresponding author. Tel.: +1 574 631 1625; fax: +1 574 631 8007.
E-mail addresses: m.brian.blake@nd.edu (M.B. Blake), david.cummings@stanford.edu (D.J. Cummings), srividya.bansal@asu.edu, ajay.bansal@asu.edu (A. Bansal).

0167-9236/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.dss.2012.01.017

work, we identify the specific SLA-based attributes that allows for the introduction of new partners. Furthermore we develop a set of principles and the associated process that utilize the SLA information to estimate service levels. This approach favors services with clean request/response (RPC-type) communication, generally known as WSDL-based web services. Further investigation would be required to assess this approach as it relates to REST-based services [37].

In this work, we investigate several research issues relevant to the integration of web services-based workflow:

1. *What SLA measures and principles are appropriate to support QoS-based assessment of existing service level guarantees?*
2. *Given a group of SLAs and knowledge about current consumer service level needs, can an on-demand request be analyzed against existing SLAs to guarantee a certain service level for a new consumer?*

The paper proceeds in the following section with a discussion of related work. In Section 3, we discuss how the SLA-based QoS assessment values are derived from higher-level organizational principles. The formal details of the attributes are defined in Section 4, and how the attributes are physically captured in markup languages in shown in Section 5. Finally, in Section 6 we evaluate the performance SLA composition as it runs parallel with traditional service composition routines.

2. Related work

There are many related projects that investigate the general use of SLAs for web services [18]. Some projects characterize SLA approaches to specific domains, such as military, database management, or information systems [13,20,26,29]. There is also a large body of work that attempts to automate the management and negotiation of SLAs [11,16,23,27,33]. Other work attempts to use semantics to automate the negotiation of SLAs [15,25].

Our work leverages markup language (i.e. WS-Agreements) for providing SLA measures as in other studies [1,28,30]. All related work describes the importance of composing SLAs. In [28], their emphasis is on compatibility between user requirements and provider constraints. Their approach suggests a promising model-based approach to assuring the compatibility.

Our work is closely related to the comprehensive work performed by [9,10,35]. Each of these approaches investigates the QoS-based and constrained composition of web services. Although Canfora et al. [9] has an elegant approach that allows for the insertion and aggregation of any user-defined QoS attribute, our approach identifies the specific SLA measures that support the user-driven assessment of an environment where their SLAs dictate current system state. Table 1 shows a survey of SLA attributes and how they are exploited in related projects specifically in the service-oriented computing domain.

Our work can be loosely classified in the body of work that looks to automate the aggregation of QoS attributes [14,21,24,32]. The uniqueness of our approach is that we consider the impacts when new web service workflows must be added as they affect the existing operational SLAs. More specifically, if a composite capability overlaps multiple SLAs, then the characteristics of an early SLA can impact a later SLA in the composition routine. Canfora et al., Cardoso et al., Zeng et al., [9,10,35] concentrate on deriving a specific composition routine as constrained by QoS values. Canfora et al., Zhang et al., and Yu et al. focus on iterative multiattribute utility approaches [12] where to focus is on the overall optimization function and less on the details of each of the attribute. Our work attempts to consider both consumer and producer concerns when assessing the entry of a new workflow. As such, our work contains low-level details for each service level objective such that subsequent optimization approaches use them as a model for optimization that targets each attribute at a low-level. Although [22] has a similar approach where SLAs are aggregated formally, they do not consider consumer and producer services independently as in our work.

In summary, we define specific SLA measures and formally integrate measures across multiple SLAs. We also define a principled process for the composition of SLAs. This work extends related work [6] by concentrating on SLA measures in markup language files as opposed to Unified Modeling Language (UML) models. Unlike other work in QoS-based web service composition, we attempt to classify QoS attributes by those associated with the provider and those associated with the consumer. Another variation here is the introduction of several high-level criteria that can be used to characterize organizations. We believe that by aggregating all lower-level attributes into a smaller set of higher-level criteria then organizations can be quantitatively evaluated or *scored*. Further evaluation in this paper justifies that such on-demand assessment of SLAs performs feasibly in an operational environment where very large numbers of web service workflows exist.

3. Assessing an enterprise based on its SLAs

The typical SLA has a large number of measures and criteria. However, in this work, we attempt to choose the measures that are most closely aligned to aggregation of a group of SLAs and ultimately their assessment. In the operational notion of web service composition, a basic web services workflow system must ensure that the input information supplied by the consumer ultimately leads to the required actions and outputs required by that consumer. In parallel, the workflow management system must ensure that the predicates and requisites match (either by syntactical or semantic techniques) in each step of the workflow. It is the operational composition routines that motivate the set of SLA attributes relevant to our work.

In order to designate which attributes that are most relevant to our proposed innovation, we developed a set of principles important to managing the quality of an enterprise with many business processes. The three relevant principles are *Compliance (Suitability)*,

Table 1
Survey of research projects that consider SLA attributes for web services.

Author names	Run time	Reputation	Uptime (Avail)	Resp time ***	Negotiation (rebinding)	Cost (price)	Success rate/ reliability	Problem resolution	Maintenance
(Blake et al.) [7] and this paper	✓	✓	✓	✓	✓	✓	✓	✓	✓
(Canfora et al.) [9]	✓	✓	✓	✓	✓	✓	✓	✓	✓
(Yu et al.) [34] Zhang et al.) [36] **									
(Zeng et al.) [35]		✓	✓	✓		✓	✓		
(Cardoso et al.) [10]			✓	✓		✓	✓		
(Jin et al.)[18] *	✓	✓	✓	✓		✓	✓		
(Mohabey et al.) [22]		✓	✓	✓		✓	✓		

* [18] list attributes, but do not develop formal approaches for composition.
** Although Canfora et al., Yu et al., and Zhang et al. do not formally define each attribute, their work focuses on an approach that allows *any* attribute to be aggregated within the composition routine.
*** Response time and run time (and in other places Service Rate) have the same or different definitions. This table places those attributes into separate columns to show the different meanings across the survey of related literature.

Sustainability, and *Resiliency*. By correlating the QoS attributes to these principles, we have developed a list that is suitable for assessing an organization based on existing SLAs. The three principles (illustrated in Fig. 1 as a Unified Modeling Language class diagram) are defined below in addition to their underlying SLA measures.

3.1. Compliance (suitability)

Compliance is the principle that ensures that the consumer receives the requested composite capability at the service level that is required. The functional notion of web service composition (as illustrated in Fig. 1) fits within this principle, since the consumer specifies their required outputs of the composition. Considering SLA terms, the composition process must assure that the aggregate *cost*, *uptime*, and *run time* are compliant with the user requirements. Cost is the sum total price of all services participating in the solution process. Uptime is a guarantee by the service providers that their services will be available a specified percentage of the time per day or month. Finally, run time is the time it takes to complete the process by adding the response times of each service in the composition.

3.2. Sustainability

Sustainability is the ability to maintain the underlying services in a timely fashion. *Success rate*, *Negotiation/renegotiation*, and *problem resolution* are related to ensuring the process continues to execute effectively. Success rate is defined by the historical rate at which a provider successfully completes a request. A consumer will require assurance that a particular business is capable of agreeing on contract terms (i.e. negotiation/renegotiation) in a timely manner. Moreover, the result of a negotiation scheme might be the development of new agreement. In addition, the service providers must be capable of resolving high-impact problems (perhaps identified by the consumer) in a timely manner. Success rate, negotiation, and problem resolution times ensure that a consumer can meet the demands of their end users.

3.3. Resiliency

Resiliency is the principle of a service to perform at high service levels over an extended period of time. This principle was derived from our work with data-centric government transactional systems. The resiliency principle specializes many of the principles that underly the general notion of *mean time between failures (MTBF)* [3,31]. Low resiliency is represented by a service that is frequently taken off-line for maintenance. Additional, the frequency of updates may impede the predictability of its operation. Consumers will need adequate notice prior to maintenance downtimes. In addition, resiliency dictates a low frequency of maintenance downtime. Peer consumers may also add comments about a particular provider and thus create a quantified *reputation*. The reputation rating also influences the resiliency of a service.

The model elements of our work were derived from our literature review and from work with collaborating stakeholders acknowledge in this paper. As such, we assume the validity of this model as the applicability of the work to other domains, in some sense, relies on these underlying elements. The paper proceeds with a formal definition of all attributes relevant to aggregating and assessing a group of SLAs, and subsequently a description of how the information can be stored physically. The final sections discuss the approach and performance for aggregating SLAs of many business processes.

4. Aggregating and assessing service level agreements

When assessing a group of web service workflows, the resulting assessment must be a result of the aggregated SLA terms of the underlying services. In some cases, aggregating the SLA terms is as straightforward as adding the measures of each of the dependent services, but in other cases the aggregate measure must be created based on consumer requirements. Since there may be subjectivity with respect to how the aforementioned SLA measures can be aggregated and calculated, we introduce the following formal concepts for composing the SLA measures in the context of web service composition.

4.1. Defining the physical entities for assessing SLAs

Definition 1. Server

Composition in a service-oriented architecture involves a consumer collaborating with a producer. The mapping from service to resources (servers and computational units) must consider system boundaries (i.e. which servers are dedicated, which are not), the size of the messages and transactions, and initialization/set up costs. Here, we illustrate an over-simplified correlation of the services to the server. The capabilities of the producers and consumers are hosted on servers. Each server can be characterized by its performance, uptime percentage, throughput, and the pre-notification time, i.e., the server informing its constituents prior to any downtime associated with server enhancements and repairs. A server, v, can be formally defined as a tuple of these SLA measures as shown below:

$$v = \left(Perf_v,\ Up_v,\ Ti_v, To_v,\ PreNT_v\right) \tag{1}$$

where $Perf_v$ is the performance of the server measured loosely in computations/second, Up_v is the uptime percentage of the server, Ti_v is the throughput of input messages measured in bytes/second, To_v is the throughput of output messages measured in bytes/second, and $PreNT_v$ is the pre-notification time measured in seconds.

Definition 2. Set of servers

Let Ω be the set of servers of all producer and consumers involved in the collaboration.

$$\Omega = \{v_1,\ v_2,\ v_3,\ldots,\ v_n\}.$$

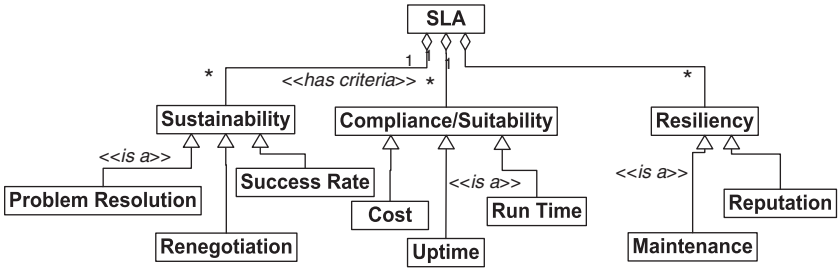

Fig. 1. A taxonomy of SLA measures for web services workflow composition.

where v_1 , v_2 , v_3 , ..., v_n are the producer and consumer servers. Each $v_i \in \Omega$ (for all i = 1 to n), is a tuple of SLA measures as described above.

Definition 3. Service

A service can be defined by some function (in some cases the average) of computations (i.e. processor cycles or relevant measure for the specific domain) it requires based on the nature of the processing domain. This computation must incorporate length of the message it receives when realizing its capabilities. A service, s, can be defined as the following pair:

$$s = (Comp_s,\ ML_s) \tag{2}$$

where $Comp_s$ is the number of required computations of the service and ML_s is the message length measured in bytes. The intention of this formalization is to illuminate conceptually the connection between the services and the servers that host them.

Definition 4. Set of services

Let Γ be the set of all services involved in a composition.

$$\Gamma = \{s_1,\ s_2,\ s_3,\ldots, s_m\}$$

where s_1 , s_2 , s_3 , ..., s_m are the services involved in the composition. Each $s_i \in \Gamma$ (for all i = 1 to m), is a pair of the number of required computations and the message length (in bytes) as described above.

Definition 5. SLA of a service

SLA of a service consists of specific web service-oriented measures, as defined in the lower elements in the taxonomy in Fig. 2. Let A_{si} be the SLA of a service s_i. A_{si} can be represented as a tuple shown below:

$$A_{si} = \left(Up_{si},\ RTime_{si},\ SResp_{si},\ Cost_{si},\ PreNT_{si},\ RenegotT_{si,}Rep_{si,}Rel_{si}\right) \tag{3}$$

where Up_{si} is the uptime of the service specified by the agreement, $RTime_{si}$ is the allowable run time, $\mathrm{SResp_{si}}$ is the allowable service response time, $Cost_{si}$ is the subscription cost or price of the service, $PreNT_{si}$ is the maintenance pre-notification time, $RenegotT_{si}$ is the renegotiation (expiration) time of the agreement, Rep_{si} is the reputation of the service and Rel_{si} is the reliability rating of the service.

4.2. Generating aggregate SLAs across workflows

In this context, we define a web service workflow as a pre-established process (as defined by an SLA) between a consumer and provider (consisting of a group of web services). At composition time, this web service workflow is not yet an active process or instance, but the specification of all pre-established partnerships or agreements of organizations to share web services. The SLA for a body of many web service workflows is obtained by composing the set of SLAs of all the services participating in each composition. This process is similar to the traditional QoS-based service composition process although QoS measures must be separated by provider and consumer concerns. The authors work closely with a federal government organization that leverages data-centric web services to develop the list of attributes that are most closely related to services concerned with sharing data from distributed information stores. Hence, the SLAs here consists of uptime of the composite service, the allowable run time, the service response time, the subscription cost or price, the maintenance pre-notification time, and the renegotiation time of the agreement.

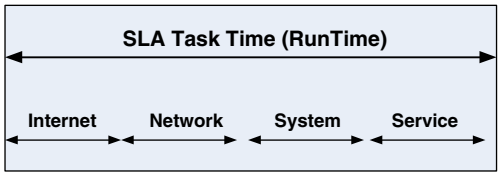

Fig. 2. Task time and run time.

Let A_{Wf} represent the composite SLA that is calculated by composing the set of all agreements of all relevant services, i.e., the services on the producer servers and any other services on the consumer's server that might impact the overall system-wide assessment.

Let A_{PS} represent the set of agreements of the services on the producer's servers that are involved in the composition.

$$A_{PS} = \{A_{PS1},\ A_{PS2},\ A_{PS3},\ldots, A_{PSm}\}$$

Let A_{CS} represent the set of agreements of the services on the consumer's servers that may impact the composite agreement A_{Wf}.

$$A_{CS} = \{A_{CS1},\ A_{CS2},\ A_{CS3},\ldots,\ A_{CSn}\}$$

4.2.1. Composite service uptime (Up_{Wf})

The uptime for the composite SLA must be less than the minimum of:

- The uptime Up_{vi} for consumer's server $v_i \in \Omega$ and
- The agreed uptimes of the producer's services in $A_{PS} = \{A_{PS1}, A_{PS2}, \ldots, A_{PSm}\}$

The relationship can be shown as:

$$Up_{Wf} < \min(Up_{vi}, \min_{i=1tom}(Up_{PSi})) \tag{5}$$

4.2.2. Composite run time ($RTime_{Wf}$)

The time required for an individual service to complete its execution can be considered the *task time*. The repeatability of the task time while the service is in commission translates to the *run time*. For a web service, the run time is a function of the internet connection, the internal network connection, the hosting hardware, and the software service. Run time is defined by the provider or consumer which is captured within the SLA. The combined run time is illustrated in Fig. 2.

The run time for the composite SLA must be less than the minimum of:

- The producer server throughput divided by the message lengths of service input and output
- The consumer server throughput divided by the message lengths of service input and output

Let $RTime_C$ represent the run time of the consumer. Including the sum of run times already guaranteed to others is important to avoid the impact of a voluminous load from external operations. The run time of the consumer server is obtained by dividing the server throughput T_C by the message length of the consumer services ML_{CS}.

$$RTime_C = T_C / ML_{CS} \tag{6}$$

Let $RTime_P$ represent the run time of all the stakeholders, i.e., the producers, p. It is the difference of the run times of the sum, m, of all services on the provider-side and the sum, n, of all client-side services.

$$RTime_P = \sum_{i=1tom} RTime_{PSi} - \sum_{i=1ton} RTime_{CSi} \tag{7}$$

The run time for the composite SLA is represented as:

$$RTime_{Wf} < \min\ (RTime_P, RTime_C) \tag{8}$$

4.2.3. Composite service cost ($Cost_{Wf}$)

The cost of using a web service can be aggregated to understand the price of an entire process. The cost of an individual service is the sum of:

- The bandwidth used (i.e. the product of the usage frequency and the message length) multiplied by the cost per byte/sec of bandwidth, which is a predefined constant *BC*. *BC* could be zero if bandwidth costs are negligible to the service provider.
- The computations used multiplied by the cost of computation/sec, which is some predefined constant *CC*. As above, *CC* could be zero if computation costs are negligible to the provider.
- The sum of the costs of all dependent services.

This results in the following equation:

$$Cost_{Wf} \geq \left(RTime_{Wf} \times ML_{CS} \times BC\right) + \left(RTime_{Wf} \times C_S \times CC\right) + \sum_{i=1tom} Cost_{PSi} \quad (9)$$

4.2.4. Composite service pre-notification time ($PreNT_{Wf}$)

When a service must be disabled for maintenance, organizations must inform their collaborators. The minimum time for notification before maintenance begins is calculated as the minimum of:

- The time of notification for the consumer server v_i
- The minimum of all notification times agreed upon for all dependent services

This relationship can be illustrated as:

$$PreNT_{Wf} \leq \min(PreNT_{vi},\ \min_{i=1tom}(PreNT_{PSi})) \quad (10)$$

4.2.5. Composite service renegotiation time ($RenegotT_{Wf}$)

Negotiation/renegotional time is the guarantee giving by the provider and consumer that defines how quickly a request for new QoS attributes will be acknowledged the other party. The renegotiation date for this agreement must be after the predefined constant waiting time, WT, with respect to each other's agreement to which this server is a party. This notion of renegotiation time is illustrated in Fig. 3. The equation below must hold for each agreement A_{CSi} A_{CS} to which the service provider is a party. This test can be shown as:

$$for\ all\ i = 1\ to\ n,\ Renegot\ T_{Wf} = max(RenegotT_{CSi} + WT) \quad (11)$$

Problem resolution is not formally defined because the calculation is a similar formula as renegotiation. As a definition (shown in Fig. 4), problem resolution can be defined as the sum of the time for recognizing the error, the time that it takes for a provider to acknowledge the error, and the actual time for resolving the problem.

4.2.6. Composite service reputation (Rep_{Wf})

In a service composition scenario, reputation is defined as a numeric score on a relative scale for a web service as captured by consumers and providers in a web service repository. The Reputation for the composite service is calculated as the average of:

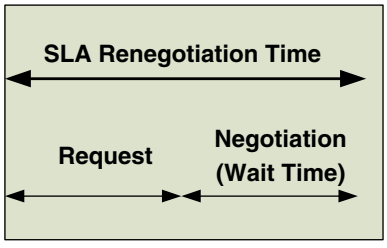

Fig. 3. Aspects of renegotiation time.

- The reputation for the consumer server v_i
- The average reputation for all the dependent services

This relationship can be illustrated as:

$$Rep_{Wf} = average(Rep_{vi} + (\sum_{i=1tom} Rep_{Psi})/m) \quad (12)$$

4.2.7. Composite service reliability (Rel_{Wf})

Reliability in traditional QoS-based web service composition has been calculated using several approaches in related work [19]. Unlike other approaches, this approach deals with SLA specifications which tend to be the worst case agreement between consumer and provider. As such, we believe closest estimate is to aggregate the values by calculating the minimum of:

- The reliability for the consumer server v_i
- The minimum of reliabilities for all the dependent services

This relationship can be illustrated as:

$$Rel_{Wf} \leq \min(Rel_{vi},\ \min_{i=1tom}(Rel_{PSi})) \quad (13)$$

Definition 6. Composite SLA

The composite SLA, A_{Wf} is a tuple of all the aggregated SLA measures as shown:

$$A_{Wf} = \left(Up_{Wf},\ RTime_{Wf},\ SResp_{Wf},\ Cost_{Wf},\ PreNT_{Wf},\ Renegot\ T_{Wf},Rep_{Wf},Rel_{Wf}\right)$$

where the aggregated SLA measures Up_{Wf}, $RTime_{Wf}$, $SResp_{Wf}$, $Cost_{Wf}$, $PreNT_{Wf}$, $RenegT_{Wf}$, Rep_{Wf} , Rel_{Wf} are obtained from the relations shown in Eqs. (5), (8), (9), (10), (11), (12), and (13) respectively.

5. Representing SLA attributes as WS-Agreements (WSAG)

In order to store and manage SLAs, the attributes must be represented in a format conducive for distributed data management. XML is a language that allows complex information to be represented with embedded metadata. An XML-based approach to representing SLA information facilitates quick interpretation of data and using translation techniques, such as the eXtensible Stylesheet Language, XSL, allows the comparison and aggregation of the underlying information. WSAG is an XML-based language that is defined with SLA attributes. A brief background is discussed here, but more information can be found at [21]. In a WSAG document, the data are represented hierarchically underneath the notion of an agreement. An agreement can be further specified with name or identifying string. An agreement can also be described by its context. Context information includes the name of the consumer and producer, the timeframe by which the agreement is valid, and other related template information. Each agreement encapsulates a list of terms. Terms describe the information of the services that are included.

Of most importance to this work are the guarantee terms. Guarantee terms consist of a service scope which contains the service names of the specific service relevant to the guarantee. The service level objective contains a predicate for the metrics that quantitatively define

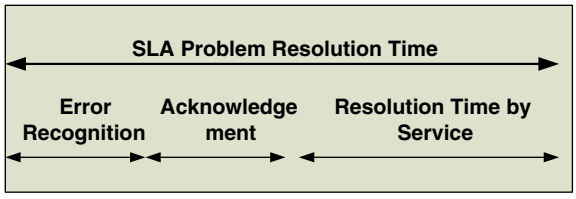

Fig. 4. Aspects of problem resolution.

the guarantee. The service level objective contains the parameter name, value, and unit of measure. As an example, the SLA attributes of uptime and maintenance are shown in Table 2. Uptime is a common attribute for SLAs. As described earlier, an uptime SLA guarantees the availability of a service by percentage over a designated period of time. The other SLA metric, maintenance notification time, is not as universally used as uptime. The example in Table 2 shows that maintenance notification time is also straightforward with regard to representation in the WS-Agreement notations.

Capturing service level agreements in XML-based notations allows the SLA attributes to be represented in a format similar to the WSDL files that represent the operational specifications of the service. WSAG can be transported and negotiated along with WSDL files. This represents a benefit if a service stakeholder wants to evaluate multiple services, side-by-side. Another benefit of capturing SLA attributes in XML-based files is the ability of enhancing attributes with semantics. It is possible that organizations will name attributes with their own specialized naming schemes. Semantics would allow disparate organizations to mediate SLA attributes that are the same but may be named differently. This approach leverages the numerous projects that use semantics to highlight web services for mediation.

However, a detriment of capturing these attributes in XML-based notations occurs when organizations have pre-established agreements. It is likely that, in a service-oriented computing environment, coalitions of businesses will form similar to partnerships that occur in traditional businesses. In such cases, the overhead of descriptive tags may be unnecessary. In addition, SLA constraints may become more comprehensive as the partnerships enhance their coordination and negotiation. These files may become too cumbersome for semi-automated manipulation where human inspection may be a required step in overall process.

6. Creating aggregated SLAs on-demand

The formalization in previous sections shows the necessary measures for creating SLAs for new business workflows that reflect the effects of other workflows existing at the same organization. The authors collaborated with The MITRE Corporation for the development of a framework for defense information systems organizations shown in Fig. 5. These organizations host large numbers of web services that provide battlefield information such as weather information, force location and tracking information, and satellite telemetry. Each of the various defense forces (Army, Navy, Air Force, etc.) provides access their web services via this shared portal. The sources vary in their guaranteed service level objectives. The defense information systems organizations are devising portals that manage SLAs in governance database while recording historical service level information. The approaches devise in this paper are incorporated within the portal logic. As such, when producers provide new services and when clients gain access to existing services, SLAs are verified for validity.

Table 2
Sample SLA attributes shown in WS-Agreement representations.

WS-Agreement for uptime

```
<wsag:GuaranteeTerm wsag:Name="uptimePref" wsag:Obligated="ACME">
  <wsag:ServiceScope>
    <wsag:ServiceName>AcmeService1</wsag:ServiceName>
  </wsag:ServiceScope>
  <wsag:ServiceLevelObjective>
    <wsag:predicate type="greater">
      <wsag:parameter>job:uptimePercentage</wsag:parameter>
      <wsag:value>5</wsag:value>
      <wsag:unit>time:seconds</wsag:unit>
    </wsag:predicate>
  </wsag:ServiceLevelObjective>
</wsag:GuaranteeTerm>
```

WS-Agreement for Maintenance

```
<wsag:GuaranteeTerm wsag:Name="maintenanceNotificationPref"wsag:
  Obligated="ACME">
  <wsag:ServiceScope>
    <wsag:ServiceName>AcmeService1</wsag:ServiceName>
  </wsag:ServiceScope>
  <wsag:ServiceLevelObjective>
    <wsag:predicate type="greater">
      <wsag:parameter>job:maintPreNotification</wsag:parameter>
      <wsag:value>7</wsag:value>
      <wsag:unit>time:days</wsag:unit>
    </wsag:predicate>
  </wsag:ServiceLevelObjective>
</wsag:GuaranteeTerm>
```

In such an ad-hoc environment, it is important to understand if the SLA composition process can be performed efficiently enough to support the real-time insertion of new partners looking to exploit existing services. In this work, we define an integrated process for workflow-based SLA composition when suggesting new SLAs. This process can be decomposed into two steps, SLA composition and evaluation. These two steps are illustrated in Fig. 6.

The SLA composition step is similar to traditional QoS-based web service composition approaches. In the evaluation step, user preferences are used to prioritize the list of candidate service chains. Numerous dynamic programming techniques can be used to achieve this step. We present a unique approach where instead of acquiring specific QoS value expectation from the user, instead user's priorities are collected. A quality score is generated based on the user's preferences. Our evaluation shows that, in real-time operations, the composition and evaluation steps are feasible considering a large number of existing business processes.

6.1. Composition: building candidate workflows

In order for SLA measures to be useful for both real-time operations and decision support, the web services workflow generation must integrate traditional web service composition with the SLA composition procedures. The SLA composition procedure must be integrated at each step and also applied to the workflow as a whole once the full process is generated. We defined the information provided by the user to be the user.predicate and the desired outcome to be the *user.reqresults*. The step.predicate is the set of information used to select subsequent services in the composition. At the initiation of a composition routine, the user.predicate is equivalent to the step.predicate. These relationships are illustrated in Fig. 7.

We introduce an integrated procedure that combines standard web service composition with SLA composition. Table 3 shows the pseudocode of the integrated process. The composition process has a main integrated process, *IntegratedComp()*. The *StepCompose()* process occurs at each step and *WorkflowChk()* process occurs once the required information and actions are realized with the execution of the sequence of web services. The *ComposeSLA()* process is the computation mechanisms that implement the SLA aggregation procedures.

6.2. Evaluation: prioritizing SLA compositions

In the prior section, candidate service chains are generated that meet the functional and SLA requirements of the user. Nevertheless, at this point, there are still multiple chains that can fulfill a capability. As such, there remains an open requirement to sort the chains based on quality and choose the best chain in the group.

In order to prioritize service chains, users are asked to provide a priority, Pr, for each attribute with respect to their environment, where $Rr = \{Pr_1, Pr_2, Pr_3, \ldots, Pr_n\}$. The priorities represent the rank ordered SLA measures from greatest to least importance. The priorities relate to the corresponding set of SLA measures, where $SLA = \{SLA_1, SLA_2, SLA_3, \ldots, SLA_n\}$. A SLA attribute has the best possible value, B. Our approach will strongly consider chains that perform favorably with respect to the user's preferences. After evaluation, G_S and G_W represent the quality score for the service and workflow, respectively.

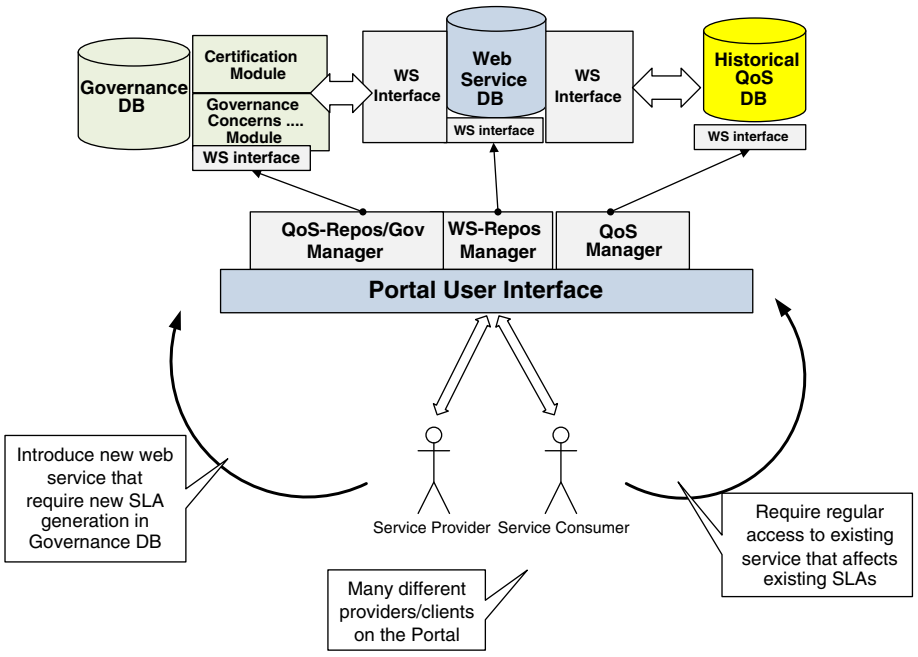

Fig. 5. A practical operational scenario.

As such, the quality score for an individual service can be defined by weighing the user's priority with respect to the ratio of the SLA measure to the best possible measure for that attribute. The calculation is defined as:

$$G_S = \sum_{n=0}^{n} \left(\Pr_n \bullet \frac{SLA_n - B(SLA_n)}{B(SLA_n)} \right) \tag{14}$$

The corresponding quality score for the service chain is:

$$G_W = \sum_{m=0} \left((G_S)_m \right) \tag{15}$$

This approach relies on user-supplied priority weights and perhaps lacks the precision of standard multiple criteria decision analysis [12] which focuses on decision optimization. However, in the next section, we demonstrate that this approach performs favorably as a real-time assessment during the dynamic composition process.

7. Performance evaluation of the integrated composition process

In real-time operations, SLA composition will be required to occur within a reasonable response time. In this work, we evaluate expected response time for the composition and prioritization of SLA measures. Our experiments are performed on a Mobile Intel Pentium 4, 2.4 GHz, 1 GB RAM, running Windows XP and the Java Runtime Environment (JRE) 6 Update 1. An initial experiment was performed that evaluates the response when prioritizing SLA-

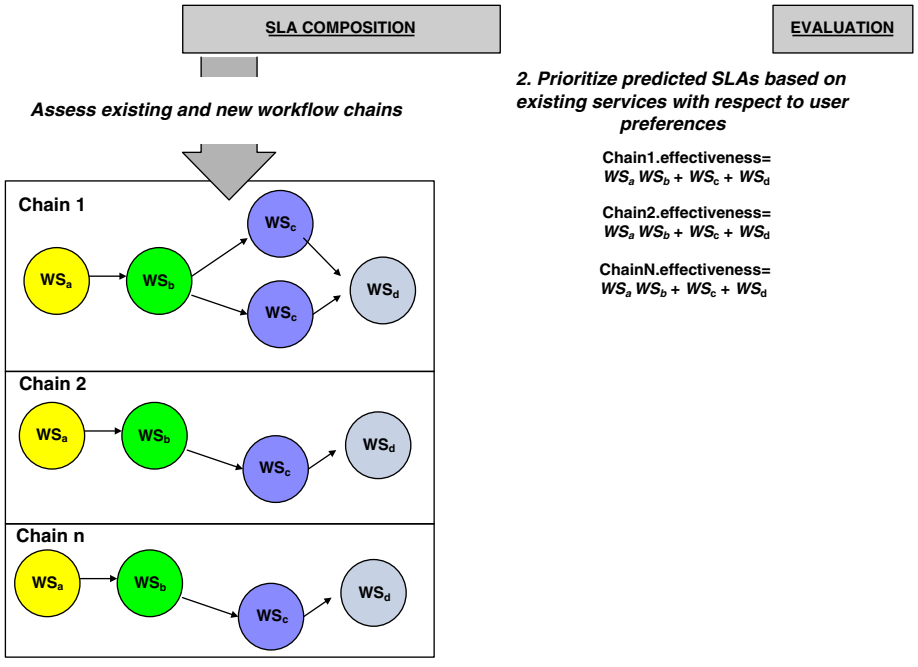

Fig. 6. Two-step process for integrated SLA workflow composition.

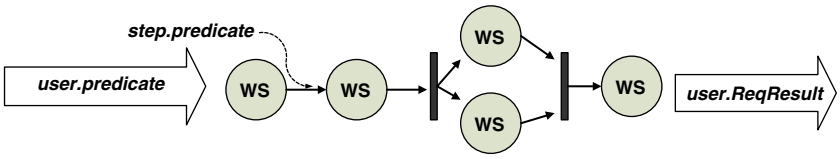

Fig. 7. Data flow in a composition routine.

annotated workflows of web services. Based on a set of random WSAG files (with uniformly distributed randomly generated SLA terms and their values), we created software that generates web service objects. In this context, using a uniformly distributed, randomly generated set of attributes for values is appropriate. This experimentation is a proof of concept that the SLA composition can execute fast enough to appropriately be included into the real-time composition scenarios. This is mostly an information management procedure. The authors understand that a non-uniform list of attributes may incur additional overhead. Here, the argument is made that data-centric organizations must keep their list of guarantees (as represented by attributes) uniform, such that their operations are consistent across each of their consumers. It is also not implausible that the provider use such policy to *ensure* a predictable processing time.

Each service objects has a unique identification codes that associates it with the corresponding WSAG files. We duplicated identification codes (i.e. ids) such that increasing numbers of services have the same ids. The process of aggregating ids was done in an attempt to simulate composition as a prerequisite step to the actual SLA composition. Each web service object was populated with six SLA measures. Although there are more attributes, we decided to experiment with 6 with the expectation that more attributes would scale consistently. Our experimentation searches the repository of web service objects, composes services of the same id, and then prioritizes the resulting chain. This experiment was executed on a repository with varying sizes from 100 to 100,000 services. The workflow size (number of services in a chain) was also varied. The number of services per chain is varied from 10 to 100,000 services within a repository of 1,000,000 services. Fig. 8 shows the results of this first experiment. Response time represents the average time required for the proposed algorithm to correlate the SLA attributes to generate a composite measure for a particular web service composition routine. Considering a reasonable workflow of web services of 100 interconnected services or less, a response time of 1.6 ms per service chain is promising. As a variation of the response time experimentation, we also investigated how the size of the repository affects the performance of our algorithm. Fig. 9 shows the response time of our algorithm as the repository increases. The workflow chain of the composition request is held constant at 10 services but the repository increases from 100 to 100,000. In the results of this experiment, search time is also considered. Given the largest repository of 100,000 services, the SLA composition time (including the processing time for discovering the relevant services (i.e. 10 services) is 30.2 s. Although the results are favorable in a simulated environment, the reader should understand that many other conditions with regard to performance variations and real-time factors on open systems reduce the confidence of these results.

In a second experiment, we evaluated the performance by varying the number of SLA attributes that are used for calculation. In this paper, we experiment with up to six attributes, but we expect that other attributes will be required to extend this approach in the future. As such, it is important to understand the overhead associated with adding a new SLA attribute for real-time operations. Although it is understood that the performance increases as the computing hardware is improved, we are generally interested in how the approach scales as the number of attributes increases in a fixed-size repository.

Fig. 10 shows the search and calculation time when varying the numbers of attributes (i.e. the number of attributes tested for each). Although this graph has a high concentration of search time (~90%), it is clear that the performance degrades favorably (linearly) at less than 12 ms per attribute (i.e. the calculation time increases less than 1 millisecond for the addition of each new attribute). Another variation of this experiment, also shown in Fig. 10, considers the impact of SLAs that process fewer attributes. In some cases, service agreements may only consider a subset of the total possible guarantee conditions. As such, less attributes may be stored per service (i.e. Attributes Stored). The Attributes Stored measure in Fig. 10 shows that there is only a slight advantage for service-oriented providers to be less stringent. Another variation considers increasing performance of SLA composition by limiting the number of attributes that a service composition considers. The major question is "Can runtime performance increase if the business management system only considers a subset of possible SLA attributes?". Experimentation shows that even at the most extreme case, if a service provider only allows services to be constrained by one attribute, the performance is only improved by approximately 30%. Predictably, as shown in

Table 3
Integrated composition pseudo-code.

```
IntegratedComp: Main integrated composition function
  StepCompose: Function that occurs at each step
  WorkflowChk : Process-Level Functional/SLA check
  composeSLA: Function that aggregates SLA measures
  PR,RenegotT,Cost : Problem Resolution, Renegotiation, Cost
  RTime,Up,PreNT: Run time, uptime, maintenance
  Rel,Rep: Reliability, reputation
  step.predicate: Message information required to execute service
  ws WSDL Object with SLA measures
  user.reqresults: Message information required as output of service
  service_chain: Candidate workflow of web services
  IntegratedComp
    {
      step.predicate = user.predicate
      WHILE (step.predicate != user.reqresults)
        { StepCompose() }
      WorkflowChk()
    }
  StepCompose
    {
    FOR EACH candidate ws where
        ws.message  step.predicate
        THEN ADD (ws) to List<ws>
    FOR EACH candidate ws IN List<ws>
      IF ((ws.PR<= step.PR) || (ws.RenegotT<= step.RenegotT) ||
        (ws.Cost>step.Cost) || (ws.RTime<= step.RTime) ||
        (ws.Up<= step.Up) || (ws.PreNT<= step.PreNT) ||
        (ws.Rel<= step.Rel) || (ws.Rep<= step.Rep))
      THEN REMOVE (ws) from List<ws>
      ELSE {
        ADD ws to List<service_chain>
        ADD ws.outputs to List<step.predicate>
          }
    }
  WorkflowChk
    {
      FOR EACH service_chain
        IF (chain.outputs  user.reqresults ) &&
        (ComposeSLA (ws.PR, ws.RenegotT ,ws.Cost, ws.Up,
          ws.PreNT, ws.RTime, ws.Rep, ws.Rel )<user.SLA))
        THEN ADD (service_chain) to Candidate List
    }
```

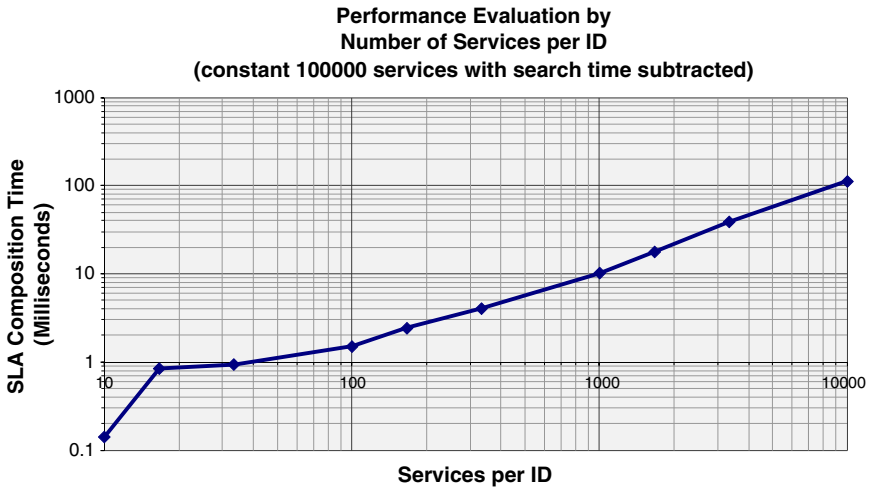

Fig. 8. The performance of the SLA prioritization function as the number of services per workflow increase (*service discovery time excluded*).

Fig. 10, the improvement in performance decreases as more attributes are considered.

8. Conclusion

When provider organizations expose their services for consumption by their peers, it is important for them to understand what they are guaranteeing. Moreover, consumers that receive such commitments must abide by their own commitments such that the providers can meet their guarantees to *all* consumers. In this work, we introduce a method of assessing an organization based on both pending and existing SLAs. Since the estimation of QoS measures for web services has been investigated in great detail, in this paper, we make a varied contribution. Our work builds on the existing studies by considering the QoS guarantees, as captured in SLAs, such that provider and consumer concerns can be modeled independently. This work also considers that SLA assessment occurs in a separate process than standard QoS estimation at service composition time. Here, we introduce high-level criteria that can be created from the aggregation of a comprehensive list of lower-level QoS-based attributes. This variation to related work facilitates *automated* cross-enterprise SLA negotiation. In this paper, we define a two-step process for composing SLAs and evaluating their efficiency. Consistent with results in related work that estimate QoS on fully operational web services, we have found through simulated experimentation that the management of third-party SLA information can be composed efficiently in parallel with the actual operational service composition. In future work, we plan to implement our approach within a real operational setting as opposed to the simulation setting that is represented in this paper. In the real operational setting, data will be produce using a variety of distributions. As such, WS-Agreement files will be appended to WSDL files and evaluated in a network environment for performance and feasibility. In addition, we plan to extend the SLA composition paradigm to protocols that mandate the discovery process within web service registries. In this way, it may be possible for adaptive software (or intelligent agents) to negotiate SLAs, in real time, while they perform on-demand, service discovery.

Acknowledgment

This work was benefited by the participation of Dr. M. Brian Blake in the Service Level Agreement Technical Exchange Meeting held at The MITRE Corporation on July 2006 in McLean, Virginia. In addition, the service discovery approach/software used in this work was partially funded by the National Science Foundation under award number 0548514.

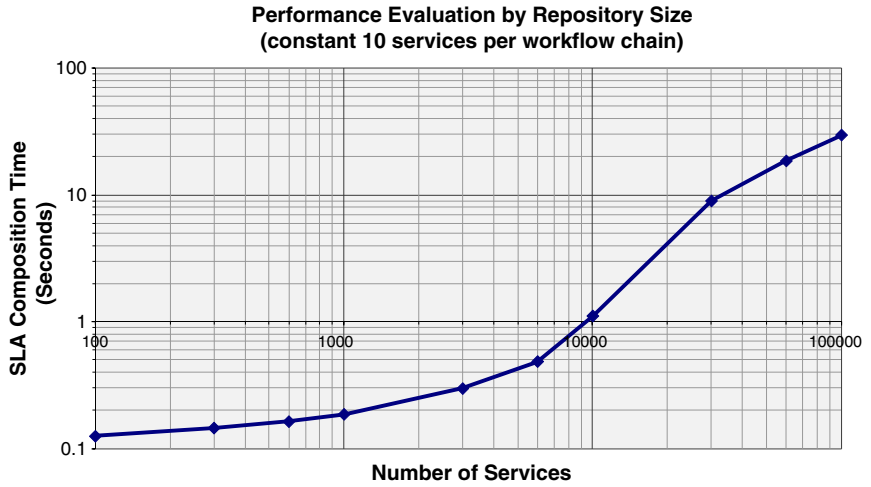

Fig. 9. The performance of the SLA prioritization function as the repository increases (*service discovery time included*).

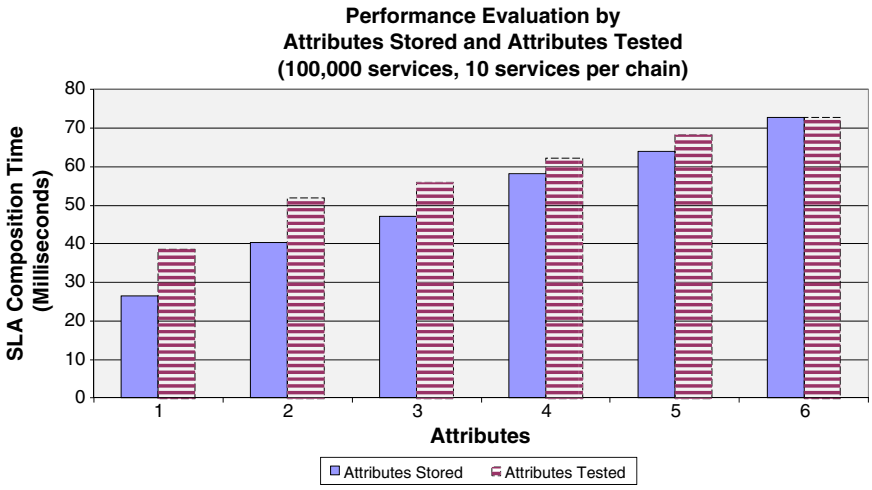

Fig. 10. The performance overhead associated with the addition of new attributes.

References

[1] P. Alipio, S. Lima, P. Carvalho, XML service level specification and validation, Proc. of the 10th IEEE Symposium on Computers and Communications (ISCC), June 2005, pp. 975–980.

[2] A. Andrieux, K. Czajkowski, A. Dan, K. Keahe, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, M. Xu, "Web Service Agreements Specification (WS-Agreement)," Proposed Recommendation, Open Grid Forum (OGF) Document Number GFD-R-P.107, Mar 2007 OGF Grid Resource Allocation Agreement Protocol Working Group (GRAAP-WG)Accessible at, http://www.gridforum.org/Public_Comment_Docs/Documents/Oct-2005/WS-AgreementSpecificationDraft050920.pdf.

[3] J.E. Angus, On computing MTBF for a k-out-of-n: G repairable system, IEEE Transactions on Reliability 37 (3) (1988) 312–313.

[4] M.B. Blake, B2B electronic commerce: where do agents fit in? Proceedings at the AAAI-2002 Workshop on Agent Technologies for B2B E-Commerce/AAAI Press, Edmonton, Alberta, Canada, July 2002.

[5] M.B. Blake, Decomposing composition: service-oriented software engineers, IEEE Software 24 (6) (Nov/Dec 2007) 68–77.

[6] M.B. Blake, A lightweight software design process for web services workflows, Proc. of the 4th IEEE International Conference on Web Services (ICWS), Sept. 2006, pp. 411–418.

[7] M.B. Blake, D.J. Cummings, Workflow composition of service level agreements, Proc. of the IEEE International Conference on Services Computing (SCC 2007), July 2007, pp. 138–145.

[8] M.B. Blake, M. Gini, Guest editorial: agent-based approaches to B2B electronic commerce, International Journal of Electronic Commerce 7 (1) (2002) 113–114.

[9] G. Canfora, G. M Di Penta, R. Esposito, F. Perfetto, M.L. Villani, Service composition (re)binding driven by application-specific QoS, Proc of International Conference on Service-Oriented Computing (ICSOC), 2006, pp. 141–152.

[10] A.J. Cardoso, "Quality of Service and Semantic Composition of Workflows," *Ph.D. Dissertation*, University of Georgia, 2002.

[11] G. Di Modica, V. Regalbuto, O. Tomarchio, L. Vita, Dynamic re-negotiations of SLA in service composition scenarios, Proc. of the 33rd EUROMICRO Conference on Software Engineeringand Advanced Applications, August 2007, pp. 359–366.

[12] J. Dyer, Maut — multiattribute utility theory in multiple criteria decision analysis: state of the art surveys, International Series in Operations Research, Management Science, vol. 78, Springer, 2005, pp. 265–292.

[13] T. Falkowski, S. Vob, Application service providing as part of intelligent decision support for supply-chain management, Proc. of the 36th Annual Hawaii International Conference on System Sciences (HICSS), vol. 3, 2003, p. 80.

[14] M. Gillman, G. Weikum, W. Wonner, Workflow management with service quality guarantees, Proc. of ACM SIGMOD International Conference on Management of Data, 2002, pp. 223–239.

[15] L. Green, Service level agreements: an ontological approach, Proc. of the 8th ACM International Conference on Electronic Commerce (ICEC), August 2006, pp. 185–194, Fredericton, Canada.

[16] D. Greenwood, G. Vitaglione, L. Keller, M. Calisti, Service level agreement management with adaptive coordination, Proc. of the International Conference on Networking and Services (ICNS), July 2006, p. 45–50, Silicon Valley, USA.

[17] C.-W. Hang, M.P. Singh, Trustworthy service selection and composition, ACM Transactions on Autonomous and Adaptive Systems 6 (1) (Feb 2011).

[18] L. Jin, V. Machiraju, A. Sahai, Analysis on service level agreement of web services, HP Technical Report, HPL-2002-180, June 2002, accessible at (2008), http://www.hpl.hp.com/techreports/2002/HPL-2002-180.pdf.

[19] J. Ko, C.O. Kim, I. Kwon, Quality-of-service oriented web service composition algorithm and planning architecture, Journal of Systems and Software 81 (11) (November 2008) 2079–2090.

[20] H. Ludwig, A. Keller, A. Dan, R.P. King, R. Franck, Web Service Level Agreement (WSLA) Language SpecificationAccessible at, http://www.research.ibm.com/wsla2007.

[21] M. Mecella, M. Scannapieco, A. Virgillito, R. Baldoni, T. Catarci, C. Batini, Managing data quality in cooperative information systems, Lecture Notes in Comptuer Science 2512 (2002) 486–502.

[22] M. Mohabey, Y. Narahari, S. Mallick, P. Suresh, S.V. Subrahmanya, An intelligent procurement marketplace for web services composition, Proc. of the IEEE/WIC/ACM International Conference on Web Intelligence, Nov. 2007, pp. 551–554.

[23] N.J. Muller, Managing service level agreements, International Journal of Network Management 9 (3) (May 1999).

[24] F. Naurmann, U. Leser, J.C. Freytag, Quality-driven integration of heterogeneous information systems, Proc. of the 25th International Conference on Very Large Databases, September 1999, pp. 447–458, Edinburgh, Scotland, UK.

[25] N. Oldham, K. Verma, A.P. Sheth, F. Hakimpour, Semantic WS-Agreement partner selection, Proc. of the 15th International World Wide Web Conference (WWW), 2006.

[26] F.R. Reiss, T. Kanungo, Satisfying database service level agreements while minimizing cost through storage QoS, Proc. of the IEEE International Conference on Services Computing (SCC), July 2005, pp. 13–21.

[27] A. Sahai, V. Machiraju, M. Sayal, A. Moorsel, F. Casati, Automated SLA monitoring for web services, Proc. of the IEEE/IFIP International Workshop on Distributed Systems: Operation and Management (DSOM), October 2002, pp. 28–41, Montreal, Canada.

[28] D. Skene, Lamanna, W. Emmerich, Precise service level agreements, Proc. of the 26th International Conference on Software Engineering (ICSE), 2004, pp. 179–188, Edinburgh, UK.

[29] I. Sorteberg, O. Kure, The use of service level agreements in tactical military coalition force networks, IEEE Communications Magazine 43 (11) (November 2005) 107–114.

[30] W. Sun, Y. Xu, F. Liu, The role of XML in service level agreements management, Proc. of the International Conference on Services Systems and Services Management, June 2005, pp. 1118–1120.

[31] K.M. van Hee, L.J. Somers, M. Voorhoeve, A modeling environment for decision support systems, Decision Support Systems 7 (1) (1991) 241–251.

[32] G. Xiaohui, K. Nahrstedt, A scalable QoS-aware service aggregation model for peer-to-peer computing grids, Proc. of the 11th IEEE International Symposium on Higher Performance Distributed Computing (HPDC), 2002, pp. 73–82.

[33] J. Yan, R. Kowalczyk, J. Lin, M.B. Chhetri, S.K. Goh, J. Zhang, Autonomous service level agreement negotiation for service composition provision, Future Generation Computer Systems 23 (6) (July 2007) 748–759.

[34] T. Yu, K.J. Lin, Service selection algorithms for composing complex services with end-to-end QoS constraints. *Proc. 3rd International Conference on Service Oriented Computing (ICSOC2005)*, The Netherlands Amsterdam, 2005.

[35] L. Zeng, B. Benatallah, A.H.H. Ngu, M. Dumas, J. Kalagnanam, H. Chang, QoS-aware middleware for web services composition, IEEE Transactions on Software Engineering 30 (5) (May 2004) 311–327.

[36] Y. Zhang, K.J. Lin, J.Y.J. Hsu, Accountability monitoring and reasoning in service oriented architectures, Service-Oriented Computing and Applications 1 (2007) 35–50.

[37] M. zur Muehlen, J. Nickerson, K.D. Swenson, Developing web services choreography standards — the case of REST vs. SOAP, Decision Support Systems 40 (1) (2005).

M. Brian Blake received the BS degree in electrical engineering from the Georgia Institute of Technology, Atlanta and the PhD degree in information technology with a concentration in information and software engineering from George Mason University, Fairfax, Virginia. He is currently Professor of Computer Science and Associate Dean of Engineering at the University of Notre Dame, Indiana. As a professor, he has published more than 120 journal and refereed conference papers in the domains of workflow and agent-based systems, service-oriented computing, distributed data management, and software engineering education. His investigations cover the spectrum of software engineering: design, specification, proof of correctness, implementation/experimentation, performance evaluation, and application.

Ajay Bansal received his B. Tech. degree in Computer Science from NIT (previously known as REC), Warangal, India, M.S. in Computer Science from Texas Tech Univ., Lubbock and Ph.D. in Computer Science from the University of Texas at Dallas. He is currently a lecturer in the College of Technology and Innovation at Arizona State University. He has over 3 years of industry experience. His research interests include Programming Languages, Logic Programming, Declarative Programming, Automated Reasoning, Service-Oriented Computing, Semantic Web Services, and Language-based Security.

Srividya Kona Bansal received her B. Tech. degree in Computer Science from NIT (previously known as REC), Warangal, India, M.S. in Computer Science from Texas Tech Univ., Lubbock and Ph.D. in Computer Science from the University of Texas at Dallas. She is currently an assistant professor in the College of Technology and Innovation at Arizona State University. She has over 5 years of industry experience. Her research interests include Service-Oriented Computing, Semantic Web, Software Engineering, Engineering Education, and Bioinformatics.