A Conceptual Model for Aggregation and Validation of SLAs in Business Value Networks

Irfan Ul Haq, Altaf Ahmed Huqqani, and Erich Schikuta

Department of Knowledge and Business Engineering, University of Vienna, Austria
Email: {irfan.ul.haq, huqqana3, erich.schikuta}@univie.ac.at

Abstract: The notions of Service Oriented Computing and Virtual Organizations have facilitated novel business scenarios such as Business Value Networks and Extended Enterprises. In these business scenarios, services orchestrate in a producer-consumer manner, forming hierarchical structures of added value. Service Level Agreements (SLAs), which are contracts between consumers and providers, guarantee the expected quality of service (QoS) to different stake holders at various levels in this hierarchy. So far, the aggregation of SLAs has been treated as a single layer composition. There is a strong requirement of a hierarchical SLA aggregation model to enable such value-chain business scenarios. In this paper we propose an architectural model to enable multi-layered SLA orchestration, aggregation, and validation.
Keywords: SLA Aggregation, Service Orchestration, SLA Validation, Business Value Networks.

1. Introduction

The work presented in this paper aims for the business enabled Internet of Services, which opens up the doors for totally new business processes for consumers and producers. In our vision, we believe that in the near future, it may be possible to sell software and resources as service and not as good. For example, "Writing a letter" can be as simple as using a telephone: Forget buying software and hardware! All we need is a simple interface to the services on the Internet, both the wordprocessor functionality (downloadable code) and the necessary physical resources (processor cycles and storage space); and everything is paid transparently via our telephone bill. The research community has identified Service Level Agreements (SLA) as the key factor for enabling such new business models. Therefore we aim for delivering a formalized basis which allows for the specification and management of SLAs for the provisioning, delivery and monitoring of services and their highly dynamic and scalable consumption. Our approach will enable the dynamic creation of Business Value Networks [7] in a service oriented infrastructure autonomously and automatically. A Service Level Agreement (SLA) is a formally negotiated contract between service provider and service consumer to ensure the expected level of service. The service consumer can be a client or another service. In the process of dynamic service composition e.g automated workflows, services are orchestrated to integrate related activities. During these service compositions, Service Level Agreements are made among different partners at various points on the orchestration. These partners include the client, the Virtual Organizations (VO) and the services. Service composition implies the composition of their corresponding SLAs. So far, SLA composition has been considered [8] as a single layer process. This single layer SLA composition model is insufficient to describe multilayered aggregation of services in supply-chain type of business networks. In a supply-chain, a service provider may have sub-contractors and some of those sub-contractors may have further sub-contractors making a hierarchical structure. This supply-chain business network spanned across various Virtual Organisations, may emerge as a Business Value Network. Business Value Networks [7] [20] are ways in which organizations interact with each other forming complex chains including multiple providers/administrative domains in order to drive increased business value. To enable these supply chain networks on Service Oriented Infrastructures (SOI), the case of the SLA aggregation needs to be elaborated and its issues resolved. SLA@SOI [5] is a European project that focusses on SLA issues in SOI. Its agenda promises the provision of such Service aggregators, that offer composed services, manageable according to higher-level customer needs.

In addition to the notion of Business Value Networks, NESSI (Networked European Software and Services Initiative), which is a consortium of over 300 ICT industrial partners [20], has pointed out various other possibilities for similar inter-organizational business models: Hierarchical Enterprises, Extended Enterprises, Dynamic Outsourcing, and Mergers to name a few. The process of SLA aggregation in such enterprises must be a hierarchical process.

There is no SLA aggregation model available till this date that can describe the hierarchical orchestration of SLAs. In the rest of the paper, we will use the term *SLA Orchestration*, in accordance with the Service Orchestration. The SLA hierarchy in Fig.1 is a direct consequence of a supply chain type of service orchestration. These layers also bound the visibility levels of service providers and the client in the sense that the client has concern only with the services immediately connected to it and can not see beyond. Similarly a service can see its coordinating services i.e. its providers and its consumers with which it is making Service Level Agreements and has no information about the rest of the service orchestration. Despite of its privacy concerns, a service is dependent upon the services beneath it in the chain. The effects of this dependency are "bubbled up" through the upper layers. There are many issues needed to be resolved in order to realize such complex business scenarios. Some of the very basic questions are:

- How can we describe this SLA-Orchestration in a more formal manner?
- How can we build trust among different partners of a Business Value Network?

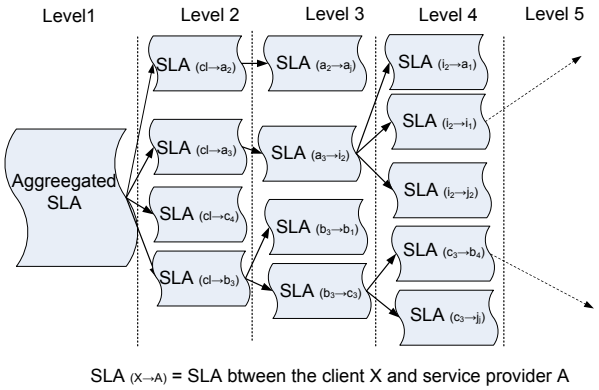

Fig. 1. Hierarchical SLA Orchestration

- Who will manage this SLA orchestration?
- How can we aggregate SLAs in an incremental way within the SLA-Orchestration?
- How to monitor and validate this SLA orchestration?

All these questions are mutually related so their answers are also highly inter-dependent.

In this paper we propose a conceptual architecture to model SLA aggregations and its related issues across heterogeneous Virtual Organizations, in order to enable Business Value Networks and similar business scenarios. The key components of this architecture are:

- a formal model to describe SLA aggregation
- a distributed trust model and
- a runtime validation model

section 2 introduce our architecture and explains how its different components are related with each other. In section 3 we describe the aggregation and formal model, in section 4 the distributed trust model, and in section 5 the validation model. Section 6 gives a survey of the related work and finally, Section 7 winds up the paper with the conclusion and the future work.

2. An Architecture for Hierarchical SLA Aggregation and Validation

Figure 2 shows our proposed architecture that tackles the questions raised in Section I by elaborating relationships among these issues and proposing solutions to resolve these issues. The overall goal of this architectural model is to elaborate the role of SLAs as an enabling technology for Business Value Networks. The proposed architecture consists of three main components:

1. Formal Model for Hierarchical SLA Orchestration and Aggregation
2. Distributed Trust Model
3. Rule based Validation Model

A formal model of SLA orchestration is required not only for a better understanding of the problem but also to provide a comprehensive platform for computation design and implementation of the system. It is also required to devise formal functions describing the hierarchical aggregation of SLAs. At the same time the formal model must be in compliance with the WS-Agreement standard. In the formal model, we introduce the concept of SLA-views. The inspiration for the SLA-views concept comes from the notion of business pro-

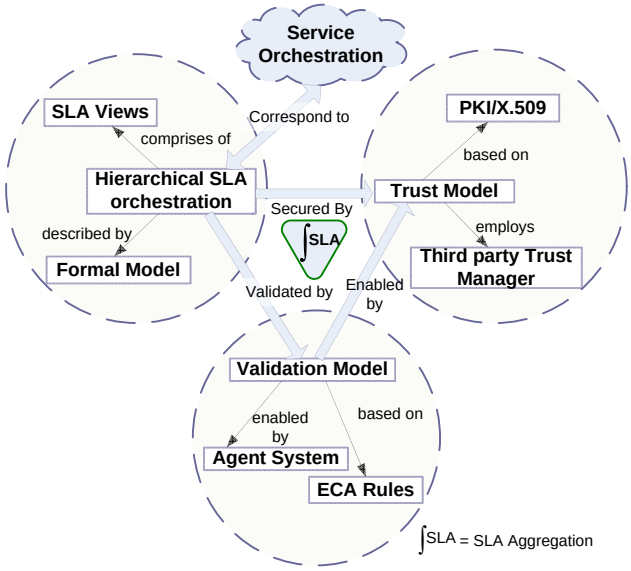

Fig. 2. Architecture for Hierarchical SLA Aggregation and Validation

cess views [17][18]. Each business partner has its own view comprising of its local SLA information. The aggregated effect of these views emerges as the overall SLA orchestration. From a service provider's point of view, it is not possible to expose the complete information of SLAs spanning across the whole chain of services to the consumer. Not only it does not make sense to reveal the information of a business partner's sub-contractors but also it would endanger business processes creating added value. With the help of SLA Views, the SLA information pertaining to different providers is veiled at various levels in the SLA orchestration. At the same time the partners of a Business Value Network, need to share their resources on the basis of mutual trust. Such a balance between trust and privacy of the stake holders requires a distributed trust model. The Distributed Trust model binds all the stake holders together with a Business Value Network. Some of the direct implications of this distributed trust may be realized during the validation, the fault tolerance and the renegotiation processes. Especially, the validation process is frequently invoked to confirm the availability and reliability of the dynamic SLA Orchestrations. The Validation model is a rule based intelligent system and coordinates very closely with the Trust model and the formal descriptive model. A distributed query is decomposed across the SLA orchestration getting validated in various SLA-Views which are scattered across different VOs and are connected via the distributed trust model. In the next sections we will discuss these components in detail.

3. Formal Model for Hierarchical SLA Orchestration and Aggregation

WS-Agreement [3], a standard SLA language from OGF (Open Grid Forum) [4], defines the structure of an SLA to consist of three parts: the Name, the Context and the Terms. Every SLA has an official name. Agreement Context contains information about the initiator, responder and provider of the agreement; expiration time of the agreement; and its template Id. Service Terms define the functional attributes of the agree-

ment whereas the Guarantee Terms contain the non functional attributes. Guarantee terms further describe the conditions, service level objectives and business value list related to the agreement. Business value list may express the importance of meeting an objective as well as information regarding penalty or reward. In our architecture, we will base the SLA aggregation process on the Service Terms and the process of Validation will be carried out by expressing the Guarantee Terms in logical rules. Let us define and construct formal relationships among these terms.

Definition 1 (Service Term) A service term denoted by $term_s$ is an element of the set Service Terms denoted by $STerms$. A $term_s \in STerms$ is a tuple such that,

$$term_s < name, value, type_a >$$

where name and value denote the name and value of a service term and $type_a$ describes its aggregation type.

We have taken the liberty to implant a new mandatory element to the WS-Agreement standard, namely, $type_a$. The $type_a$ element corresponds to the aggregation function that helps us automate the aggregation of SLAs. We postpone its definition to the latter part of the paper where we will discuss the aggregation process.

Definition 2 (Guarantee Term) A guarantee term denoted by $term_g$ is an element of the set Guarantee Terms i.e, $GTerms$. A $term_g \in GTerms$ is a tuple such that:

$$term_g < SLO, condition_q, BVL >$$

where SLO that represents Service Level Objectives, $condition_q$ that represents Qualifying Conditions and BVL representing Business Value List may be expressed as a set of logical rules based on Horn Logic and Event Condition Action(ECA) rules. We will explain these terms with examples in section VII. Combining the above two definitions, now we can define the notion Terms of the WS-Agreement.

Definition 3 (Term) A $term \in Terms$ is a pair such that

$$term = (term_s, term_g)$$

where $term_s \in STerms$ and $term_g \in GTerms$ and $Terms = STerms \cup GTerms$.

Following the above definitions, SLA can now be formally defined as:

Definition 4 (SLA) A service Level Agreement (SLA) denoted by sla is a tuple

$$sla < Context, Terms >$$

where $Terms = STerms \cup GTerms$

and Context is a list of strings. Context defines the names of the SLA provider, the consumer and the initiators. It also contains the duration of the SLA. To construct a formal model that describes the SLA orchestration on the whole while catering the privacy concerns of the service providers at the same time, we introduce the concept of SLA-Views.

The concept of Views originated from the databases but has been widely used in business workflows [9][17][18]. In workflows, a view can be a subset of that workflow or can be a representation of that workflow in aggregated or abstracted fashion [12]. But in case of SLAs, we need to consider a few differences. An SLA-Orchestration is not a workflow so the rules of workflows are not applicable on it. For instance, in a workflow, rules such as: there should be a single start and single exit or every split should have a join, do not apply on SLA aggregation structure. A view in an SLA-Orchestration represents the visibility of a business partner. Every service provider is limited only to its own view. A partner (for example a service) makes two kinds of SLAs: the SLAs for which it acts as a consumer and the SLAs for which it is a provider. For clarity, we name these two types as the consumer-oriented SLAs and the producer-oriented SLAs respectively. In Figure 3, SLAs are connected to small circles,which we call *Aggregation Points*, by certain edges called *Dependencies*. There are two types of dependencies. Consumer-oriented SLAs are connected to the aggregation points from below by the *sink dependencies* and the producer-oriented SLAs are connected from above by *the source dependencies*. To understand the overall picture of the SLA Orchestration, we need to formalize these concepts.

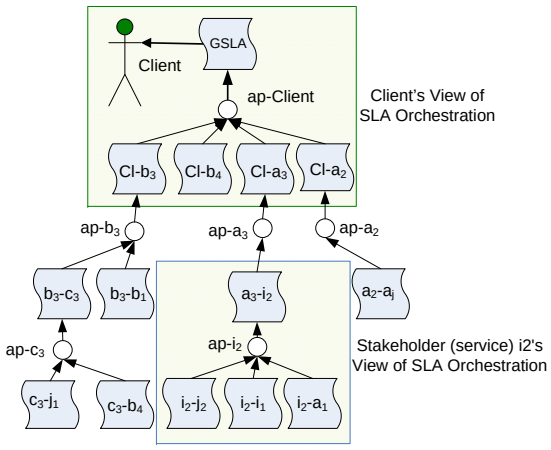

Fig. 3. SLA-Views

Definition 5 (Aggregation Point) An Aggregation Point ap is an object such that

$$ap < aggsla >$$

where $aggsla$ is the aggregated SLA produced by aggregating the consumer-oriented SLAs connected to it. In Figure 3, *ap-i_2* is an aggregation point. An aggregation point is the point where the consumer-oriented SLAs (of the consumer service) are aggregated and on the basis of their aggregated content, the service is able to decide what it can offer as a provider.

Definition 6 (Dependency) A dependency Dep is a set that is the union of two sets namely Dep_{src} and Dep_{sink} i.e

$$Dep = Dep_{src} \cup Dep_{sink}$$

In Fig. 3,Dependencies are shown as edges joining the aggregation point *ap-i_2* with three consumer-oriented SLAs and one producer-oriented SLA.

Definition 7 (Source Dependency) A source dependency $dep_{src} \in Dep_{src}$ is a tuple

$$dep_{src} < ap, sla >$$

where *ap* is the aggregation point and *sla* is the producer-oriented SLA. In Figure 3, it is represented by the directed edge from the aggregation point *ap-i_2* to the producer-oriented SLA *a3-i2*. Dep_{src} is the set of all source dependencies within the SLA-Orchestration. Let

$$source : (ap) \rightarrow dep_{src}$$

$source(ap_i)$ is the unique $s \in Dep_{src}$ with $s.ap = ap_i$. This means that the function source maps each aggregation point ap_i to a unique source dependency s.

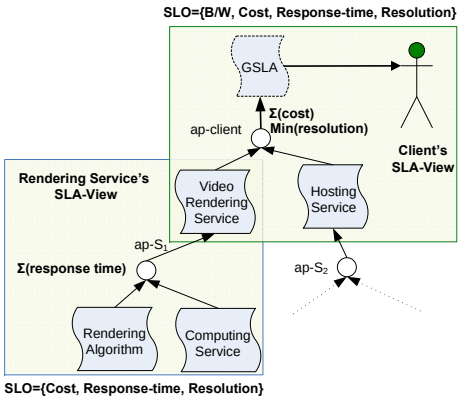

Fig. 4. Example Scenario for Aggregation and Validation

Definition 8 (Sink Dependency) A sink dependency dep_{sink} is a tuple

$$dep_{sink} < sla, ap >$$

where *ap* is the aggregation point and *sla* is the consumer-oriented SLA. In Figure 3, it is represented by the directed edge from the consumer-oriented SLA i_2-i_1 to the aggregation point ap-i_2. The aggregation point ap-i_2 is connected with three sink dependencies. Dep_{sink} is the set of all sink dependencies. Let

$$sink : (ap) \rightarrow P(dep_{src})$$

$sinks(ap_i)$ is the set $S_{sink} \subseteq Dep_{sink}$ i.e. $S_{sink} \in P(Dep_{sink})$, such that

for each $s_i \in S_{sink}$ $s_i.ap = ap_i$

where $P(Dep_{sink})$ is the power set of Dep_{sink}. This means that the function sink() maps each aggregation point ap_i to a set of sink dependencies S_{sink} and each member of this set is mapped to the unique aggregation point, i.e. ap_i.

Based on the above concepts, now we are in a position to provide a formal definition of the SLA-View.

Definition 9 (SLA-View) An SLA-View denoted by $slaview$ is a tuple such that

$$slaview < sla_p, dep_{sr}, apoint_o, SLA_c, Dep_{sn} >$$

where sla_p = producer-oriented SLA, SLA_c= Set of consumer-oriented SLAs, dep_{sr}= source dependency, Dep_{sn}= set of sink dependencies, and $apoint_o$= aggregation point.

In Figure 3, the SLA-Views of the client and a service are highlighted.

Definition 10 (SLA-Orchestration) An SLA_{orch} is a tuple such that:

$$SLA_{orch} < SLA, APoints, Deps >$$

where SLA is set of all sla within an SLA-Orchestration, $APoints$ is set of aggregation points ap and $Deps$ is set of dependencies dep. Another way to describe the SLA-Orchestration is in terms of SLA-Views, i.e.

$$SLA_{orch} = \cup_{i=1}^{n} slaview_i$$

This means that the whole SLA-Orchestration may be seen as an integration of several SLA-Views.

In the aggregation process, terms of the consumer-oriented SLAs are aggregated. WS-agreement has no direct support for such an aggregation so we introduced an attribute for aggregation type namely, "$type_a$" in the Definition 1. WS-Agreement gives the liberty to incorporate *any* external schema. Therefore $type_a$ can be made an essential part of the service terms and will describe how the corresponding service will behave during the aggregation process. We can define $type_a$ in a formal way, as follows:

Definition 11 (The function type$_a$) A $type_a \in Types$ is a function that maps a set of tuples to a single tuple which is the aggregation of that set.

$$type_a : tuples(term) \rightarrow term$$

$$type_a(term_1, ...term_n) = term_{agg}$$

We define $type_a$ as an aggregation function that aggregates n terms into one term. Its result is $aggsla$ in the aggregation point (please see Definition 5). Each term in $aggsla$ is computed by applying the type function for that term to the values of the terms for all the dependent (consumer-oriented) SLAs which define that term. In the present context, we define four types of terms namely sumtype, maxtype, mintype and neutral but new types can be added according to the situation. The function sumtype can be defined as follows.

Definition 11.1 (The function sumtype)

$$sumtype \in Types(\Leftrightarrow sumtype : tuples(term) \rightarrow term$$

$$sumtype(term_1, ...term_n) = \sum_{i=1}^{n} term_i$$

$type_a$ is an aggregation function that aggregates n number of terms into one term. sumtype is of the type of $type_a$ and takes the summation of all terms. Examples include terms for storage space, memory, availability and cost. The functions for mintype, maxtype are also aggregation functions and can be defined in a similar fashion. Neutral may be another function that does not change any value and represents all the terms separately.

Example 1: Formalization of Aggregation

We have demonstrated these aggregation function in scenario depicted in Figure 4. Suppose there is a client who requires a video rendering service coupled with high speed computation facility. The client would also like to host this video online and expects a high download demand which requires high bandwidth. During the aggregation process, aggregation functions depending upon the $type_a$ of the *service term* will be applied at each aggregation point. For instance, cost and response time are declared *sumtype*, so summation function will be performed at the aggregation points. The response time at *ap*-S_1 is the sumtype function for the response time is a function of the temporal complexity of the rendering algorithm, the response time of the computational service, the size of the input data and the latencies of communication. This aggregation of consumer oriented SLAs is strictly from the service provider's point of view and a variation in values, in cost for example is quite likely to happen in the provider oriented SLA. We have also shown the resolution is a *mintype* function and the resultant resolution at *ap-client* will be the minimum from the rendered one and the one offered by the hosting service. During the aggregation of the service terms, the corresponding guarantee terms are accordingly aggregated. At each aggregation point, the Service Level Objectives (SLOs) can be represented as a set of logical rules. We have also shown two sets of SLOs corresponding to the two SLA-Views in the Figure. We will see in section 5, how a set of SLOs will be represented as a conjunction of premises in a logical rule.

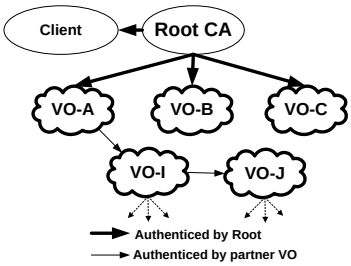

Fig. 5. PKI based Distributed Trust Model.

4. Trust Model

In order to understand what kind of Trust Model is suitable for SLA Orchestrations, let us review the situation. SLA Orchestrations may span across various VOs forming a distributed infrastructure. This means that the corresponding trust model should be a distributed trust model. One of the best candidates is the Cross-CA Hierarchical Trust Model [16][28]. This model employs the Public Key Infrastructure based certificates for authentication. Every member must possess such a certificate that will be issued from a Certification Authority (CA). In the Cross-CA Hierarchical Trust Model the top most CA is called the root CA that provides certificates to its subordinate CAs. These subordinates can further issue certificates to other CAs (subordinates), services or users. Unless a CA authenticates a candidate, it can not earn the membership of that organization. There is one root CA that authenticates all the CAs within the organization. But in case of Business Value Networks where multiple VOs are cooperating, some questions arise. For instance, a very common question would be: Which VO will have the root CA? The answer could be very simple if one of the VOs is playing the central role. But in dynamic ad hoc enterprizes, this is not that simple. During service orchestrations, services may form temporary composition with any other service present in any VO. Whose parent VO will act as the root CA in this case? A solution for dynamic ad hoc networks is the inclusion of a Third Party Trust Manager acting as a root CA. In our architecture we have suggested a Third Party Trust Manager that will act as a root CA and authenticate member VOs. Some of those authenticated members may further authenticate other members and services and so on. This Trust Model will be based on the PKI based system. Public Key Infrastructure (PKI) provides a secure and reliable way to authenticate an entity through certificates. The certificate contains the name of the certificate holder and the holder's public key, as well as the digital signature of a Certification Authority (CA) for authentication. The PKI technology is used to authenticate a user, having a verified copy of public key across many domains. The requirement of distributed PKI based trust management system arises because of the SLA details lying across multiple Virtual Organizations. It employs the concept of public and private keys. The public keys are distributed among all the trusted parties packaged in digital certificates. Digital Certificates are issued by a Certification Authority (CA). These certificates help to build a trust chain. A CA can also issue certificates to other CAs. The VOs authenticated by the third party root CA may be termed as Partner VOs. This situation is depicted in figure 5. The SLA management framework depicted as the Client, is also authenticated by the root CA.

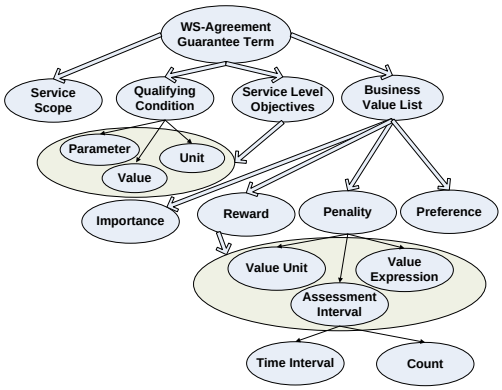

Fig. 6. Structure of Guarantee Term in WS-Agreement

In case of PKI based authentication each service will have a possession of an X.509 format certificate that will be issued by its corresponding Certification Authority (CA). The authentication layer in each VO middle-ware may be based on Grid Security Infrastructure (GSI) where all resources need to install the trusted certificates of their CAs. GSI uses X.509 proxy certificates (PCs) to provide X.509 Public Key Cryptography standard (PKCs) for identification [16] and to enable Single sign-on and Delegation.With Single Sign-On, the user does not have to bother to sign in again and again in order to traverse along the chain of trusted partners (VOs and services). SLA views integrate very closely with the trust model to maintain a balance between trust and security. While the trust model promises trust and security, the SLA views protect privacy.

5. Validation Model

Service Level agreements are frequently validated throughout their life cycle. Runtime Validation ensures that the Service Level Objectives and guarantees are in complete conformance with the expected levels. Figure 6 describes the structure of Guarantee Terms in WS-Agreement. The most important constituents are Service Level Objectives that define the desired quality of service; Qualifying Conditions that expresse assertions over service attributes and penalty and reward expressions.

The process of validation is performed by first representing the Guarantee Terms as logical rules and then using these rules as distributed queries. During the validation process, queries are decomposed making their premises as subgoals. This backward chaining propagates throughout the SLA Orchestration. If all the subgoals are satisfied then the validation is successful. This whole scheme is enabled by

1. the distributed trust model,
2. the SLA views, and
3. Rule Responder [24] based Multi Agent System

Referring to distributed trust model, facilitated with the single sign on and delegation, the validation process may be termed as the *Delegation of Validation*. The trust model facilitates the distributed query to travel across various domains through a single sign-on and delegation mechanism. The agent enabled rule based system has been inspired from the Rule Responder Project [24]. We apply the findings of this project by in-

troducing User Level Agent (ULA), which generates the distributed query or validation request. The distributed query is received and redirected by two types of agents: (i) *VO Level Agents (VOLA)* and (ii) *Provider Level Agent (PLA)*. VOLA intercepts the query at the boundary of a VO and redirects it towards the PLA, which is a service provider's agent and has access to service provider's view.

The validation process completes when the validation query is propagated to all the SLA-views in the SLA orchestration. To understand how the Guarantee Terms can be expressed as rules and how the rules are transformed into distributed query, we refer to the formal definition of Guarantee Terms. In Definition 2, we defined Guarantee Terms as $term_g < SLO, condition_q, BVL >$. Now we will describe how these arguments can be formally expressed.

SLOs can be conveniently expressed through conjunctive derivation rules. Lets revisit the scenario depicted in Figure 4. In that scenario, we learned how *service terms* are aggregated using *sumtype*, *mintype*, *maxtype* and neutral functions. These functions are applied on the basis of aggregation type of a service term, identified by $type_a$ attribute. We also learned that the SLOs can be represented as conjunctive premises of logical rules.

Example 2: Formalization of Validaton

In our scenario, the user is interested to render her videos and then host them on the web. Her requirements include a maximum cost of 45 Euros, maximum response time of 5 seconds, minimum resolution of 640x480 pixels and the minimum bandwidth (from the hosting service) of 50 Mbps. In Figure 7, we have depicted this scenario from validation point of view and have expressed the SLOs of final aggregated SLA in form of a derivation rule at the top of the Figure. It must be kept in mind that the SLOs refer to an established SLA and their ranges are meant to be guarded in order to maintain desired levels of service. VOLA and PLA are agents which automate the distributed querying process. In Figure 7, the set of SLOs belonging to the final aggregated SLA are represented in form of a logical rule. The predicates lt and gt denote lesser-than and greater-than respectively. This conjunctive set of SLOs is expressing the minimum required Bandwidth i.e 50 Mbps, minimum cost, i.e. 45 Euros, and maximum response time, i.e. 5 seconds, and the minimum resolution i.e. 640X480 pixels in the following rule:

$SLO() \leftarrow \neg gt(Cost, 45, euro) \wedge \neg gt(Resptime, 5, sec) \wedge \neg lt(Resol, 640X480, pixels) \wedge \neg lt(BW, 50, mbps)$.

During the validation process, this rule will be decomposed such that each premise will become a subgoal thus forming rule chains. The initial steps of decomposition procedure are depicted at the bottom of the Figure. In the Figure, VO-level-Agents (VOLA) have been shown to receive and track the distributed query whenever it enters a new VO. For each service provider there is a Provider Level Agent (PLA). A PLA after finishing its job, would redirect the distributed query to the service provider's PLA that comes next in the hierarchical chain. The process continues until the whole SLA Orchestration is validated. In our scenario, VOLA-B receives a subgoal $\neg gt(Resptime, 5, sec)$ representing the requirement that the total response time of the system should not be more that

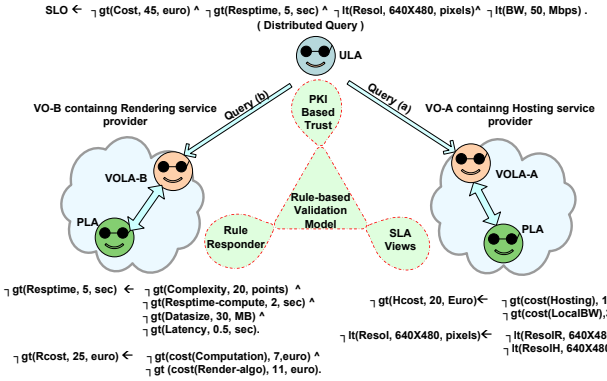

Fig. 7. Validation through distributed query decomposition

5 seconds. The SLO depends on several factors such as the complexity of the rendering algrithm, size of the data, latency and response time of the computational hardware which is expressed as the new subgoal:

$\neg gt(Resptime, 5, secs) \leftarrow \neg gt(Complexity, 20, points) \wedge \neg gt(Resp\text{-}compute, 2, sec) \wedge \neg gt(Datasize, 30, mb) \wedge \neg gt(Latency, 0.5, sec)$

During the aggregation process the premises in the above rule were being summed up because their corresponding $type_a$ attribute has been declared as a *sumtype* function. The SLO expressing the cost will be divided between the two service providers as shown in the Figure 7. The service cost at the level of VOLA-A should be less than 20 and is dependent on the sum of the cost for hosting and the cost for local bandwidth. The varying upper limit of cost at different levels reflect the profit margins of different providers e.g. the provider in VOLA-A has a profit margin of 5 Euros.

It should be noted that in accordance with the WS-Agreement standard, there are three arguments in each SLO, denoting: the SLO name, its value and its unit respectively. VOLA-B receives its subgoal and connects it with the corresponding derivation rule in a similar fashion. The delegation of validation will continue until the whole SLA orchestration is validated. At each level, the corresponding reward and penalty conditions will also be checked and if required then appropriate action will be taken.

Qualifying Conditions and penalty and reward expressions can be expressed through Event Condition Action (ECA) rules. For example, if we want to express the statement "If the response time of the service is not less than 30 seconds then there is a penalty of 5 Euros", we can write its WS-Agreement equivalent as follows:

```
<wsag:Penalty>
<wsag:AssesmentInterval>
  <wsag:TimeInterval> 30 </wsag:TimeInteval>
  <wsag:Count>1</wsag:Count>
</wsag:AssesmentInterval>
<wsag:Value Unit>Euro</wsag:ValueUnit>
<wsag:ValueExpr>5</wsag:ValueExpr>
</wsag:Penalty>
```

This can also be represented by ECA rules:

$timer(second, T) \leftarrow Timer(T) \wedge seconds(T) mod 30 = 0$

$event(Violate) \leftarrow \neg ping(Service1, RT) \wedge RT > 30$

$action(Penalty) \leftarrow penalty(Obligation, 5)$

Now combining together:
$ECA(monitor) \leftarrow timer(second, T) \wedge event(Violate) \wedge action(Penalty)$

Similar approach can be used for the renegotiation, fault tolerance and breach management processes. During renegotiation, the distributed query traverses in the same way towards the service providers, offering those terms which are desired to be renegotiated. During fault tolerance and breach management, violations are localized through a similar invocation of the distributed query. The advantage of having derivation rules is that they are much more structured and can be very easily transformed in a rule based markup language, thus becoming processable in autonomic and autonomous way.

6. Related Work

The related work spans across many research areas. These research areas include aggregation and formalization of SLAs, workflow views, trust models for VOs and Rule based SLA validation. In fact, we have introduced a new concept namely SLA views in this paper for which we had to dig in the research on workflow views to extract the relevant knowledge.

6.1 SLA Aggregation

SLAs were originally used by Information Technology (IT) organizations and adopted by telecommunications providers [19] to manage the quality of service(QoS) expectations and the perception of their contributions to the company's productivity and bottom-line success. Telecommunication Information Networking Architecture Consortium (TINA-C) [2] defines a service architecture providing guidelines for constructing deploying and withdrawing TINA services. WSLA [1], WS-Agreement[3] and SLANG [14] are some popular languages of Service Level Agreement. Service composition direclty implies the SLA composition. However a little research [8], [13] has been carried out towards dynamic SLA composition. Blake and Cummings [8] have defined three aspects of SLAs which are Compliance, Sustainability and Resiliency. They assume services to exist only at one level. The research area corresponding to the management of such aggregated SLAs is still wide open. Ganna Frankova [13] has highlighted the importance of this issue but she has just described her vision instead of any concrete model. Unger et al's work [27] is directly relevant to our focus of research. They focus on aggregation of SLAs in context of Business Process Outsourcing (BPO). They synchronize their work with Business Process Execution Language (BPEL) and WS-Policy. Their model is based on SLO aggregation of SLAs. One of the limitation of their approach is that they take into account services related to one process in one enterprise because they focus on BPO. Our approach describes cross-VO SLA aggregation and strictly adheres to WS-Agreement.

6.2 Formal Description of SLA

Aiello et al [6] present a formal description of SLA. Their approach is based on WS-Agreement. They extend the WS-Agreement standard by introducing a new catagory of terms called Negotiation Terms. They build automaton representation of SLA states to describe the negotiation process. Their formal model is too vague and they do not explain how this model will describe the sub-entities in WS-Agreement. Unger et al [27] present a rigorous formal model for SLA aggregation. They follow BPEL and WS-Policy whereas our formal model adheres to WS-Agreement standard.

6.3 Workflow Views

The concept of Workflow Views is used to maintain the balance between trust and security among business partners [9][17][18]. Schulz et al [25] have introduced the concept of view based coalition workflows. Chiu et al [10] present a meta model of workflow views and their semantics based on supply chain e-service but their model lacks an integrated cross-organizational perspective. Other authors [26][15], however, do propose a global view or a decomposition process based on the views. But none of them have focussed on the dynamic workflows in their approach. Chiu et al [11] describe a contract model based on workflow views. They demonstrate how management of contracts can be facilitated. Based on the views of participating organizations they construct an e-contract model that defines e-contracts in plain text format. Modern Service Level Agreements are XML based, more complex and more dynamic due to features such as negotiation, renegotiation, and fault tolerance, etc.

6.4 Distributed cross-VO Trust Models

The Grid Security Infrastructure (GSI) and the security modules of middle-ware, provide a set of security protocols for achieving mutual entity authentication between a user (actually a user's proxy) and resource providers [28]. Each party has a public-key based cryptographic credential in the formulation of a certificate. GSI uses X.509 proxy certificates (PCs) to enable Single sign-on and Delegation [16]. PCs can be created on the fly without requiring any intervention from conventional CAs. In the cross-CA Hierarchical trust model [16][28], the top most CA is called the root CA that provides certificates to it subordinate CAs. These subordinates can further issue CA to other CAs (subordinates), services or users.

6.5 Rule based Validation of SLAs

Adrian Paschke and Martin Bichler have done an extensive work on RBSLA (Rule Based SLA) project [23]. RBSLA transforms SLAs into logical rules to automate their management and monitoring. They discuss knowledge representation of SLAs with complex business rules and policies. RBSLA [22][23] uses a combination of Horn Logic, Deontic Logic and ECA (Event-Condition-Action) rules. RBSLA also covers many related areas such as the breach management, authorization control, conflict detection and resolution, service billing, reporting, and other contract enforcements. RBSLA employs query driven, backward reasoning for SLA management. RBSLA is a useful tool to transform text based SLAs. Our approach adheres to the WS-Agreement standard which is a structured document thus making the challenge of automation more convenient. Oldham et al [21] have extended WS-Agreement by building a rule based ontology on the WS-

Agreement. Their SWAPS schema [21] transforms constructs from the Guarantee terms into predicate based markup language. They admit that their schema is limited to a specific domain. In our architecture we propose a distributed validation model for an SLA orchestration in Business Value Networks.

7. Conclusion and Future Work

We have presented a conceptual architecture to enable SLA Orchestrations in Business Value Networks. It consists of a formal model for SLA Orchestration, and SLA-Views, a distributed trust model and a rule based validation system. This approach allows automatic and dynamic creation of Business Value Networks in Service Oriented Infrastructure. In the future, we plan to elaborate the distributed rule based validation system while adhering to the WS-Agreements standard. We plan to integrate our validation model as a use case in the Rule Responder Project [24]. We also plan to implement the complete system through an iterative design phase.

Acknowledgements

We are extremely thankful to Prof. Harold Boley (University of New Brunswick, Canada) for a fruitful discussion and valuable suggestions in connection with the Rule Based Validation Model, which helped a lot to improve this paper.

References

[1] WSLA language specification, version 1.0, IBM corporation january 2003.
[2] Service architecture version 5.0, tina baseline, TINA Consortium.
[3] Web Service Agreement (WS-Agreement) Specifications, Open Grid Forum (OGF), 2007.
[4] Open Grid Forum (OGF), [http://www.ogf.org/], last access: Feb 9, 2009.
[5] SLA@SOI Project, [http://www.sla-at-soi.org/index.html], last access Feb. 9, 2009.
[6] M Aiello, G Frankova and D Malfatti, What's in an agreement? a formal analysis and an extension of WS-Agreement, Lecture Notes in Computer Science, catagory Security and SLA, Springer Berlin Germany.
[7] V Allee, Reconfiguring the value network. Journal of Business Strategy, Vol. 21, No. 4, Jul-Aug 2000.
[8] M B Blake and D J Cunnings, Workflow composition of service level agreements, International Conference on Services Computing (SCC2007).
[9] I Chebbi, S Dustdar and S Tata, The view based approach to dynamic inter-organizational workflow cooperation. Data and Knowledge Engineering, Vol. 56, 2006, pp. 139–173.
[10] D Chiu, S Cheung, S Till, K Karalapalem, Q Li and E Kafeza, Workflow view driven cross-organisational interoperability in a web service environment. Information Technology and Management, Vol. 5, 2004, pp. 221–250.
[11] D Chiu, K K Q Li and E Kafeza, Workflow view based e-contracts in a cross-organisational e-services environment. Distributed and Parallel Databases, Vol. 12, 2002, pp. 193–216.
[12] J Eder and A Tahamatan, Temporal consistency of view based interorganizational workflows, 2nd International United Information Systems Conference, Austria.
[13] G Frankova, Service Level Agreements: Web services and security. Springer Verlag, Berlin Heidelberg, 2007, pp. 556–562.
[14] D Lamanna, J Skene and 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). IEEE-CS Press.
[15] Q Li, D Chiu, Z Shan, P Hung and S Cheung, Flows and views for scalable scientific process integration, First International Conference on Scalable Information Systems, Hong Kong.
[16] A Lioy, M.Marian, N.Moltchanova and M Pala, PKI past, present and future. International Journal of Information Security, Springer Berlin, pages 1829, p. 2006.
[17] D R Liu and M Shen, Workflow modeling for virtual processes: an order-preserving process-view approach. Information Systems, Vol. 28, 2002, pp. 505–532.
[18] D R Liu and M Shen, Business-to-business workflow interoperation based on process-views. Decision Support Systems, Vol. 38, 2004, pp. 399–419.
[19] Marilly, E Martinot, O Betge-Brezetz and S Delegue, Requirements for service level agreement management, IEEE Workshop on IP Operations and Management, ALCATEL CIT, Marcoussis, France.
[20] NESSI-Grid, Grand vision and strategic agenda (sra 3.0), 2008.
[21] N Oldham, K Verma, A Sheth and F Hakimpour, Semantic WS-Agreement partner selection, Proceedings of the 15th international conference on World Wide Web, Edinburgh, Scotland.
[22] A Paschke and M Bichler, SLA representation management and enforcement, The 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service.
[23] A Paschke and M Bichler, Knowledge representation concepts for automated SLA management. Int. Journal of Decision Support Systems (DSS), March 2006.
[24] A Paschke, H Boley, A Kozlenkov and B Craig, Rule responder: RuleML-based agents for distributed collaboration on the pragmatic web, Proceedings of the 2nd international conference on Pragmatic web Tilburg, The Netherlands.
[25] K A Schulz and M E Orlowska, Facilitating cross-organisational workflows with a workflow view approache. Data and Knowledge Engineering, Vol. 51, 2004, pp. 109–147.
[26] M Shen and D R Liu, Discovering role-relevant process-views for disseminating process knowledge. Expert Systems with Applications, Vol. 26, 2004, pp. 301–310.
[27] T Unger, F Leyman, S Mauchart and T Scheibler, Aggregation of Service Level Agreement in the context of business processes, Enterprise Distributed Object Computing Conference (EDOC '08) Munich, Germany.
[28] S Zhao, A Aggarwal and R D Kent, PKI-based authentication mechanisms in grid systems, International Conference on Networking, Architecture, and Storage.