Rule-based validation of SLA choreographies

Ul Haq, Irfan; Paschke, Adrian; Schikuta, Erich; Boley, Harold

Publisher's version / Version de l'éditeur:
https://doi.org/10.1007/s11227-010-0492-1
The Journal of Supercomputing, 2010-11-18

Journal of Super Computing manuscript No.
(will be inserted by the editor)

Rule-Based Validation of SLA Choreographies

Irfan Ul Haq · Adrian Paschke · Erich Schikuta · Harold Boley

Received: date / Accepted: date

Abstract For the Service Economy to prosper, IT-based Service Markets are required to perform certain business actions autonomically and autonomously, e.g. helping companies to establish networks of business relationships. Service Markets, to be practically realized require an enabling infrastructure that supports business-to-business (B2B) relationships among business stakeholders, resulting in value chains.

B2B workflow interoperation across Virtual Organisations (VOs) brings about novel business scenarios. In these scenarios, parts of workflows corresponding to different partners can be aggregated in a producer-consumer manner, leading to hierarchical structures of added value. Service Level Agreements (SLAs), which are contracts between service providers and service consumers, guarantee the expected quality of service (QoS) to different stakeholders at various levels along this hierarchy. Automation of service composition in these coalition workflows directly implies the aggregation of their corresponding SLAs. This hierarchical choreography and aggregation poses new challenges regarding SLA description, management, maintenance, validation, trust and security. In this paper we focus on design and architecture of an agent-oriented, rule-based validation framework for hierarchical SLA aggregation, enabling cross-VO workflow cooperation. The framework is based on the Rule Responder architecture, the RBSLA project, a formal model of SLA Views, and a distributed trust model.

I. Ul Haq · E. Schikuta
Department of Knowledge and Business Engineering, University of Vienna, Austria
Tel.: +142-1-427739526
Fax: +142-1-427739518
E-mail: irfan.ul.haq[AT]univie.ac.at

A. Paschke
Institute of Computer Science, Freie University Berlin, Germany

H. Boley
Institute of Information Technology, National Research Council, Canada

Keywords Service Level Agreements · Rule-Based SLA Validation · Service Value Chains · Value Networks · Workflow Management

1 Introduction

The creation of an IT-based Service Economy requires the realization of the notion of Service Markets. A market does not represent a simple buyer-seller relationship, rather it is the culmination point of a complex chain of stakeholders with a hierarchical integration of value along each point in this chain.

With the advent of on-demand service infrastructure (e.g. Cloud Computing and Internet of Services), there is a high potential for third party solution providers such as Composite Service Providers (CSP), aggregators or resellers [1][2] to compose together services from different external service providers in form of workflow activities to fulfill the pay-per-use demands of their customers. A cumulative contribution of such Composite Service Providers will emerge as service value chains. Services are traded under formal contracts known as Service Level Agreements (SLA). SLA in Service Oriented Infrastructure (SOI), is an automatically processable contract between a service and its client; the client being a person, organization or yet another service.

The work presented in this paper aims at the validation of dynamic and automated coalition workflows in a service-enriched environment such as the Grid or Clouds. During a business-to-business workflow composition across Virtual Enterprizes, Service Level Agreements are made among different partners at various points across the choreography. Workflow composition also implies the composition of their corresponding SLAs. SLA composition in workflows has been mostly treated [3] as a single-layer process. This single-layer SLA composition model was insufficient to describe coalition workflows [4] where a multilayered aggregation of services is required that results in supply-chain type of business networks. Therefore the research community has just started taking notice [5] of the importance to describe this hierarchical aggregation of SLAs.

This supply-chain business network of the coalition workflows spun across various VOs may result in the so-called Business Value Network. Business Value Networks [6] are ways in which organizations interact with each other forming complex chains, including multiple providers/administrative domains, in order to drive increased business value. In a supply chain, a service provider may have sub-contractors and some of those sub-contractors may have further sub-contractors, resulting in a hierarchical structure. This leads to a hierarchical structure of SLA contracts between different supply chain partners. Since this SLA hierarchy may span across several VOs with no centralized authority, in the rest of the paper we will call it *Hierarchical SLA Choreography* or simply SLA Choreography, in accordance with the underlying Service Choreography. The SLA Choreography needs to be validated for the reasons of consistency and fault tolerance. The validation of hierarchical SLA Choreography is not a trivial task. The validation is a distributed process crossing

administrative boundaries and and leading to hierarchical dependencies. The validation process is also required to be built upon several intermingled issues such as privacy, trust, security and automatic reasoning. On the one hand it is not sensible to expose information of all the SLAs across the whole hierarchy of services as it will endanger the business processes creating added value, and on the other hand for the sake of automation required in the processes of validation, certain information must be shared among all the stakeholders making the value chain. To achieve this balance between privacy and trust we have introduced the concept of *SLA Views* that plugs in within our validation framework. It provides a privacy similar to that of *workflow views* [4, 8, 9], where every service provider is limited to only their own view. SLA Views also complement the notion of distributed trust among the various partners in a coalition workflow [10].

In this paper we present an agent-enabled rule-based runtime validation framework for hierarchical SLAs, which allows the provisioning, delivery, and monitoring of services in coalition workflows as well as their highly dynamic and scalable consumption.

The framework is based upon:

- the Rule Responder Architecture [11],
- findings of the *RBSLA* project [12],
- a formal model of SLA Views, and
- a distributed trust model [10].

Section 2 introduces the relevant models contributing to our validation framework by introducing the Rule Responder architecture, RBSLA, SLA Views, and Distributed Trust. In section 3 we describe the runtime validation framework for Hierarchical SLA Aggregation, and in section 4, our Delegation-of-Validation approach. Section 5 gives a survey of related research and finally, Section 6, the conclusion and future work.

2 Validation of SLA Aggregation in the Cross-section of Models

Validation of hierarchical SLA aggregation corresponding to cooperative workflows is a distributed problem. The service choreography may be distributed across several Virtual Organizations and under various administrative domains. Figure 1 (left) shows that the SLA Choreography is realized on the basis of a formal model that utilizes the concept of SLA Views, which preserve the privacy of stakeholders (see section 2.3).

This hierarchical choreography of heterogeneous services is only possible through a well defined distributed trust schema. Another challenge in this regard, discussed in detail in [7], is the step-wise aggregation of SLAs for the series of service providers at different levels in the service chain. The complete information of aggregated SLA at a certain level in the service chain is known by the corresponding service provider and only a filtered part is exposed to the immediate consumer. This is the reason why during the validation process, the

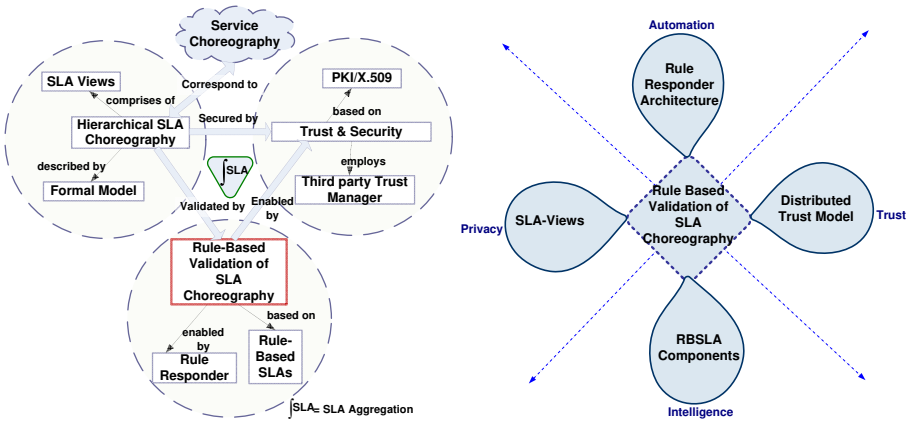

Fig. 1 Validation of SLA Choreographies: (left) With Context to Aggregation, (right) As a Cross-section of Models

composed SLAs are required to be decomposed in an incremental manner down towards the supply chain of services and get validated in their corresponding service providers's domain. A validation framework for the composed SLAs, therefore, faces many design constraints and challenges: a trade-off between privacy and trust, distributed query processing, and automation to name the most essential ones. The aforementioned challenges bring in a cross-section of models depicted in Figure 1(right). The privacy concerns of the partners are ensured by the SLA-View model (see section 2.3), whereas the requirement of trust can be addressed through a distributed PKI (Public Key Infrastructure) based trust model. There are two rule based systems contributing in terms of automation and intelligence. Rule Responder [13] weaves the outer shell of the validation system by providing the required infrastructure for the automation of role description of partners as well as steering and redirection of the distributed validation queries. The knowledge representation techniques from the RBSLA (Rule based Service Level Agreements) project [12] contribute at the core of validation system. Different parts of the WS-Agreement compliant SLAs can be transformed into corresponding sets of logical rules, which can compose together during the process of SLA composition and can be decomposed into separate queries during the process of validation. We will discuss these models one by one to find out how they contribute to our proposed validation approach.

2.1 Rule Responder Architecture

Rule Responder (http://responder.ruleml.org) is a rule-based enterprise service middleware for distributed rule inference services and intelligent rule-based (Complex) Event Processing on the Web. It utilizes modern enterprise service technologies and Semantic Web technologies with intelligent agent services that access external data sources and business vocabularies (ontologies),

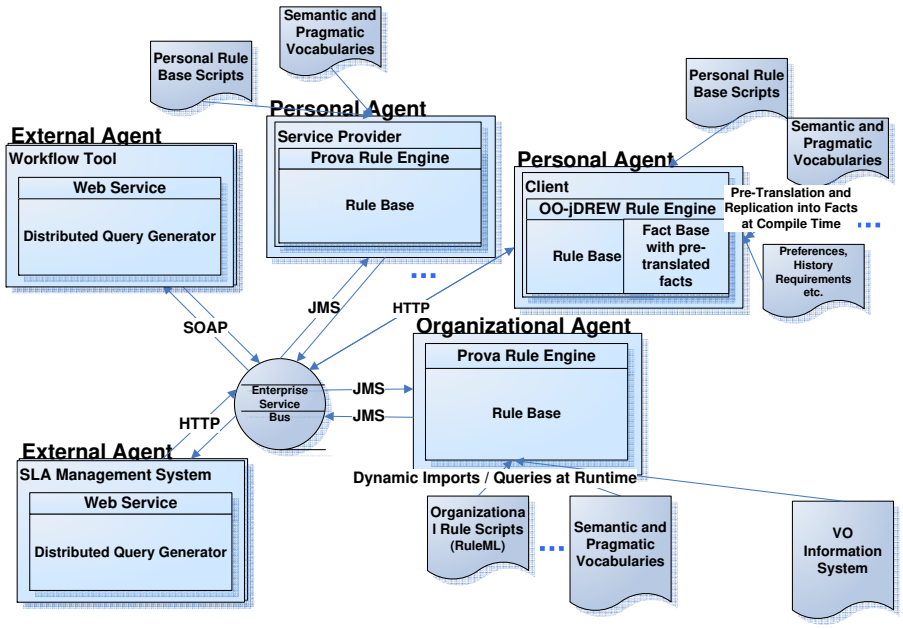

Fig. 2 Rule Responder Services for SLA Validation

receive and detect events (complex event processing), and make rule-based inferences and autonomous pro-active decisions and reactions based on these representations (enterprise decision management). For a description of the syntax, semantics and implementation of the underlying logical formalisms and its usage in IT Service Management (ITMS) see [14]. Rule Responder adopts the approach of multi agent systems. There are three kinds of agents:

- Organisational Agents
- Personal Agents
- External Agents

A virtual organization is typically represented by an organizational agent and a set of associated individual or more specific organizational member agents. The organizational agent might act as a single agent towards other internal and external individual or organizational agents. In other words, a virtual organization's agent can be the single (or main) point of entry for communication with the "outer" world (external agents). Similar to an organizational agent, each individual agent (personal and external) is described by its syntactic resources of personal information about the agent, the semantic descriptions that annotate the information resources with metadata and describe the meaning with precise business vocabularies (ontologies) and a pragmatic behavioural decision layer which defines the rules for using the information resources and vocabularies/ontologies to support human agents in their decisions or react autonomously as automated agents/services. The flow of information

is from external to organisational to personal agent. Figure 2 shows the Rule Responder agents contributing to SLA validation. Two external agents outside of VO invoke the organizational agent by sending HTML and SOAP messages. Typical examples of external agents are web browser, client service or a workflow tool. In our scenario Rule Responder provides the rule-based enterprise service middleware for highly flexible and adaptive Web-based service supply chains. Rule Responder utilizes RuleML [15] as Platform-Independent Rule Interchange Format. The Rule Markup Language (RuleML) is a modular, interchangeable rule specification standard to express both forward (bottom-up) and backward (top-down) rules for deduction, reaction, rewriting, and further inferential-transformational tasks. It is defined by the Rule Markup Initiative, an open network of individuals and groups from both industry and academia. Figure 2 shows Enterprise Service Bus (ESB), the Mule open-source ESB [16], as Communication Middleware and Agent/Service Broker to seamlessly handle message-based interactions between the responder agents/services and with other applications and services using disparate complex event processing (CEP) technologies, transports and protocols. ESB provides a highly scalable and flexible application messaging framework to communicate synchronously but also asynchronously with external services and internal agents which are deployed on the bus. A large variety of more than 30 transport protocols provided by Mule can be used to transport the messages. Currently the Prova [14], OO jDREW [17], and Euler [18] rule engines are implemented as three rule execution environments.

2.2 RBSLA

The Rule Based Service Level Agreements (RBSLA) [12, 19, 14] project focuses on sophisticated knowledge representation concepts for service level management (SLM) of IT services. At the core of its contract and service level management tool are rule-based languages to describe contracts such as service level agreements or policies in a generic way. The research draws on basic knowledge representation concepts from the area of artificial intelligence (AI) and knowledge representation (KR) and as well as on new standards in the area of web services computing and the semantic web. A particular interest is the investigation of expressive logic programming techniques and logical formalisms such as defeasible logic, deontic logic, temporal event/action logics, transaction and update logics, description logics as a means of deriving formal declarative contract specifications with which to reason about ideal and actual behaviours relating to agreed contract norms (permissions, obligations and prohibitions and their violations (contrary-to-duty obligations) or exceptions (defeasible prima facie obligations). The important advantages of our approach are the automated verification, validation and consistency checks of large possibly distributed and interchanged rule sets, the automated chaining, (scoped) reasoning and execution of rules and distributed contract modules

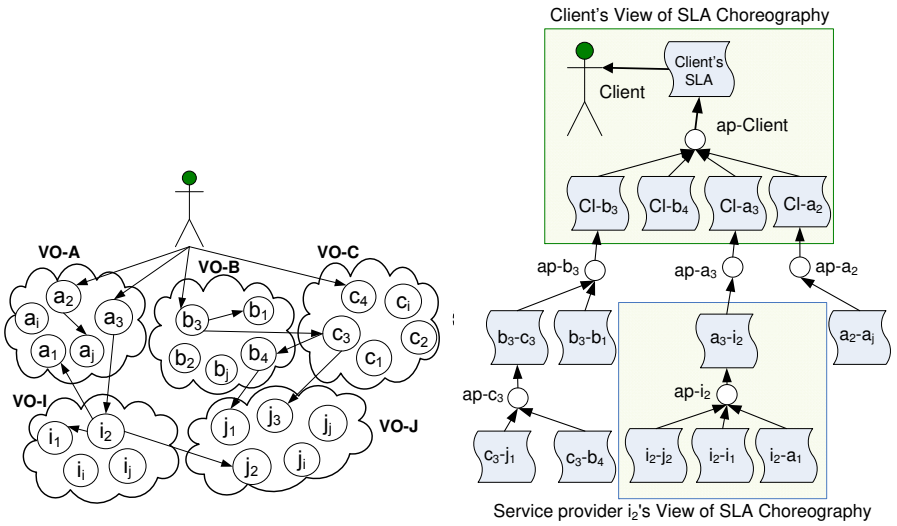

Fig. 3 SLA Choreography comprising of SLA Views

as well as the flexibility in the dynamic extension with new contract rules (dynamic transactional updates).

2.3 SLA Views

The concept of Views comes from the databases and has been very successfully adapted in business workflows. Workflow views are employed to separate different administrative domains in workflow coalitions [4].

An SLA Choreography 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 Choreography structure. Therefore the views of SLA Choreography are quite different from the workflow views. A view in an SLA Choreography represents the visibility of a business partner. Every service provider is limited only to its own view. On the left side of Figure 3, services from different VOs have been shown to form a choreography. On the right side of the Figure, the network of SLAs corresponding to this service choreography has been depicted. The SLAs are encapsulated within their respective views. Two SLA views have been highlighted for the client and the service i_2 respectively.

This scheme can be generalized for all the other partners of this SLA Choreography. A partner (for example a service) makes two kinds of SLAs: the consumer-oriented SLAs and the producer-oriented SLAs. In Figure 3, SLAs are shown to be connected to small circles, representing the *Aggregation Points* via certain edges called *Dependencies*. There are two types of dependencies. Consumer-oriented SLAs can be connected to the aggregation points from below by the *consumer role dependencies*, indicating that the *ap* has a

consumer role with respect to that SLA, whereas the producer-oriented SLAs are connected to the aggregation point from above by the *producer role dependencies*. It must be noticed that the producer and consumer roles of SLAs are reflected through their respective dependencies with reference to a particular aggregation point (ap). Thus one SLA may have two roles with respect to two aggregation points, it is connected to. The notion of SLA View does not need to take into account any loops or cyclic graphs. An SLA View corresponds to a unique producer-oriented SLA. This important property plays a crucial role to track down the precise value chain corresponding to a specific composite service within an SLA Choreography.

To understand the overall picture of the SLA-Choreography, we need to formalize these concepts. (For a more rigorous and complete formal model elaborating SLAs and SLA Views, please see [7])

Definition 1 (Aggregation Point). An Aggregation Point *ap* is an object such that

$$ap =< aggsla, KB >$$

where *aggsla* is the aggregated SLA produced by aggregating the consumer-oriented SLAs connected to it. KB denotes the *Knowledge Base* consisting of business rules, aggregation rules, policies and facts. The business rules and the aggregation rules inside KB play an important role during the negotiation, aggregation and validation processes.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.

Now let us define dependencies which have been shown in Figure 3(a) as edges joining the aggregation point with the producer and consumer oriented SLAs. The Aggregation Point *ap-i*$_2$ is connected with three consumer-oriented SLAs and one producer-oriented SLA through dependencies.

Definition 2 (Producer Role Dependency). A *producer role dependency* dep_{pr} is a tuple

$$dep_{pr} =< ap, sla >$$

where *ap* is the aggregation point and *sla* is the producer-oriented SLA. In Figure 3(a) it is represented by the directed edge from the aggregation point *ap-i*$_2$ to the producer-oriented SLA, $sla_{a_3-i_2}$.

Each $dep_{pr} \in Dep_{pr}$, where Dep_{pr} is the set of all producer role dependencies within the SLA-Choreography. Let

$$prodrole : (AP) \rightarrow Dep_{pr}$$

$prodrole(ap_i)$ is the unique $s \in Dep_{pr}$, for which a unique producer-oriented SLA exists with $s = (ap_i, sla_i)$. This means that the function *prodrole* maps each aggregation point ap_i to a unique SLA through a unique producer role dependency *s*.

Definition 3 (Consumer Role Dependency). Similarly a *consumer role dependency* dep_{cr} is a tuple

$$dep_{cr} =< sla, ap >$$

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 consumer role dependencies. Each $dep_{cr} \in Dep_{cr}$, where Dep_{cr} is the set of all consumer role dependencies within the SLA Choreography.

Definition 4 (Dependency). A dependency Dep is a set that is the union of two sets namely Dep_{pr} and Dep_{cr}, which are pairwise disjoint, i.e.

$$Dep = Dep_{pr} \cup Dep_{cr}$$

$$Dep_{pr} \cap Dep_{cr} = \phi$$

Based on these definitions we see in Figure 3 that the producer-oriented SLA (a_3-i_2) is dependent on the terms of the corresponding consumer-oriented SLAs, aggregated at *ap*-i_2 . For example the bandwidth and space aggregated at *ap*-i_2 would be the upper limit of what service i_2 can offer to service a_3. At the same time service i_2 will have to decide about its profit on the basis of the information about total cost in the aggregated SLA using business rules from within its KB. The aggregation point in this sense is also a decision point for a service.

With having all the related concepts formalized, now we are in a position to provide a formal definition of the SLA-View.

Definition 5 (SLA-View). An SLA-View denoted by *slaview* is a tuple such that

$$slaview_i =< sla_{p_i}, dep_{pr_i}, ap_i, SLA_{c_i}, Dep_{cr_i} >$$

where sla_{p_i} is a producer-oriented SLA, SLA_{c_i} is a set of consumer-oriented SLAs, dep_{pr_i} is a producer role dependency between ap_i and sla_{p_i} and Dep_{cr_i} is the set of consumer role dependencies between the members of SLA_{c_i} and the ap_i. Each aggregation point ap_i in the SLA-Choreography corresponds to a unique *sla-view*$_i$.

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

Definition 6 (SLA Choreography). An SLA_{chor} is a tuple such that

$$SLA_{chor} =< SLA, APoints, Deps >$$

where SLA is set of all *sla* within an SLA Choreography, $APoints$ is set of aggregation points *ap*, and $Deps$ is set of dependencies *dep*. During the aggregation process, terms of the consumer-oriented SLAs are aggregated. WS-agreement has no direct support for such an aggregation but it gives the liberty to incorporate *any* external schema. We introduce an attribute for aggregation type namely, "$type_a$". The attribute $type_a$ 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 7 (Aggregation function type$_a$) A $type_a \in Types$ is a function that maps a set of terms to a single term, which is the aggregation of that set:

$$type_a : P(Terms) \rightarrow Terms$$

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

We define $type_a$ as an aggregation function that aggregates n terms into one term. Each aggregated term 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. We can define different types of terms namely sumtype, maxtype, mintype, andtype, ortype, and neutral but new types can be added according to the situation. The aggregation terms based on the logical operators are specially helpful in aggregating guarantee terms. For instance in case of reward and penalty expressions, logical operators are very useful to represent the aggregated sum of the terms. The aggregation process is an incremental process, with aggregation functions applied at each step i.e. every SLA view in the chain [7].

2.3.1 Special Case: Aggregation of Guarantee Terms

Guarantee Terms (GTs) can also be aggregated together similar to the Service Description Terms (SDTs) as described in the previous section. However there are certain peculiarities to be considered when it comes to the Guarantee Terms. First of all, Guarantee Terms are optional terms in context with the WS-Agreement standard. Secondly, even if two aggregating SDTs have GTs associated with themselves, the GTs may refer to different service level parameters, i.e. they provide guarantees in form of Service Level Objectives (SLOs) about different properties of the service.

An example is a service consumer who wants to aggregate two similar storage services. The aggregation of SDTs will give him the total sum of available disk space, but what if two vendors are providing GTs describing entirely different aspects, e.g. access time and availability of service? These two aspects are not related hence can not be aggregated. To solve this problem there can be many approaches:

- a solution to this problem can be found if the SLA negotiation process somehow facilitates the consumer to ask for guarantees upon the desired properties of services thus helping him setting up the identical guarantees with its different service providers. Not only this type of mechanism is very difficult to achieve, but by restricting the variability of SLA contents, it also turns out to be contrary to the automation requirements of the process for which it was originally designed for.
- a similar approach can be based on the renegotiation for a revision of SLA with new guarantees. This approach may not be successful every time because the service provider may not be in a position to offer the required type of guarantee.
- some popular (or straightforward) guarantees may be standardized to be always offered for the relevant services by all the service providers. This approach will definitely improve the situation but there will be always new services with innovative properties expressed by unseen guarantees.

- if the guarantees translate to the quality of service then in some situations it may be desirable to use ORType aggregation in order to segregate the services on the basis of their guarantees. For this purpose the service terms should be declared as ORType.
- the most straightforward and safe method is to leave the guarantees disaggregated and the situation should be reported to the service provider to take some decision. In this way, we may allow each service provider in the supply chain to figure out and set up its own Guarantee Terms during the aggregation process based on its personal business rules.

The last approach also conforms to our formal model. We assume the aggregation point to be the decision center of the service provider as well. Within this decision center, the aggregation of SLAs is performed to facilitate the formation of business objectives of the service provider. Therefore, when a stakeholder in the supply chain acts as a service provider, it needs to layout its business strategy at least once before starting the provision of services. Our aggregation model thus only promises a semi-automatic aggregation of Guarantee Terms. In that context, the aggregation of Guarantee Terms is purely a business issue and is interlinked with the business goals of the service provider. This approach also resolves another very crucial issue of aggregating reward and penalty expressions. Within the aggregation point, the reward and penalty expressions must be expressed in accordance with the business rules of the service provider. A subset of those business rules may be dedicated especially to facilitate the aggregation process. We represent the aggregated Guarantee Terms in form of logical rules. In section 3, we will explain how to apply backward chaining mechanism on these rules as part of our validation framework.

2.4 Distributed Trust Model

Trust not only plays a crucial role in reducing SLA violations in workflow compositions but it has also been shown [20] that maximizing participants trust even helps runtime scheduling to survive in dynamic and open environment. We need to choose a suitable trust model that integrates seamlessly with our aggregation and validation model. During service choreography, services may form temporary composition with other services, scattered across different VOs. Whose parent VO will act as the root CA in this case? Public Key Infrastructure (PKI) is a popular distributed trust model that offers certificate containing 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 public keys are distributed among all the trusted parties, packaged in digital certificates, building trust chains. A solution for dynamic ad hoc networks is the inclusion of a *Third Party Trust Manager* acting as a root CA. We propose a PKI based trust model with 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. 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 [21] proxy certificates 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). This can be achieved by the Cross-CA Hierarchical [21] [22]Trust Model where the top most CA, called the root CA provides certificates to its subordinate CAs and these subordinates can further issue certificates to other CAs (subordinates), services or users. In [23], we have proposed a more rigorous hybrid trust model integrated with our validation framework. It combines the PKI and reputation based trust approaches to foster trust among stakeholders. However the details of this trust model are beyond the scope of this paper. 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.

3 Rule based Validation Framework for Hierarchical Aggregation of SLAs

Service Level agreements are frequently validated throughout their life cycle. Runtime Validation ensures that the service guarantees are in complete conformance with the expected levels. WS-Agreement [24] defines a detailed structure of Guarantee Terms with the most important constituents being: Service Level Objectives that define the desired quality of service, Qualifying Conditions that express assertions over service attributes, and Penalty and Reward expressions. These terms are represented as logical rules following the RBSLA specifications. These rules are composed together during the process of SLA aggregation [7], introduced in the section 2.3. The process of validation is performed by 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 Choreography. If all the subgoals are satisfied then the validation is successful.

We have discussed how SLA Views contribute to the process of hierarchical SLA aggregation across SLA choreographies. A distributed validation process is required for this aggregation of SLA choreographies to frequently validate it for the sake of maintenance and fault tolerance. Due to the consumer-oriented aggregation structure of SLA choreography and because the aggregation details are obscured at different levels of hierarchy, a distributed top-down validation mechanism is a good strategy for the complete validation of a hierarchical SLA aggregation. A top-down validation approach has several advantages in connection with its implementation:

- interfaces can be validated before going into details of modules,
- in case of a problem on higher levels, one does not need to go into lower levels,

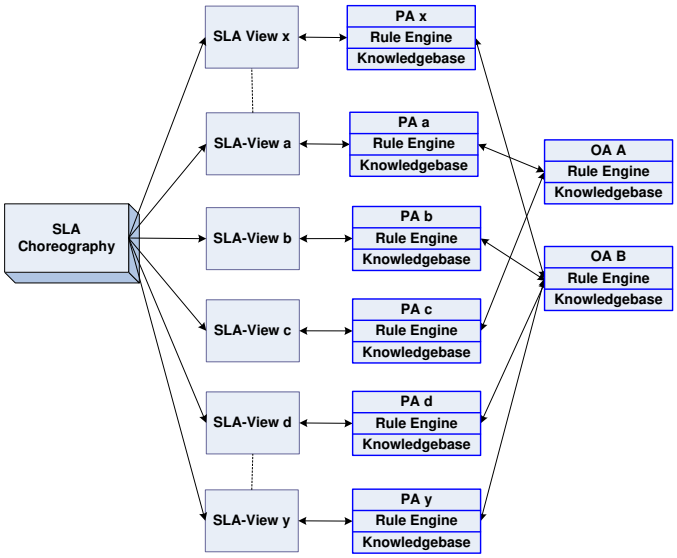

Fig. 4 Every SLA-View corresponds to a Personal Agent

– since in the view based SLA aggregation, the top level represents the client's perspective therefore this approach can better translate the on-demand validation queries initiated from the client.

Figure 4 depicts how the Rule Responder and SLA-Views work together to enable this scheme. Each SLA-View that in fact represents a service provider in the SLA Choreography, is connected to a Personal Agent (PA). SLA choreography is composed of various SLA views. A PA receives the queries from the Organizational Agent (OA) and having the complete information of its consumer oriented SLAs in its knowledge-base, performs the local validation and delivers back the responses on behalf of the service providers.

It must be highlighted that the overall collaboration between VOs is based on choreography, while the internal collaboration model within a VO (one closed enterprise service network) can be either choreography with no central authority or an orchestration with orchestration workflows defined in the organizational agent as under control of a central authority within this particular VO. Rule Responder can span across several VOs and can support both of the collaboration models. Rule Responder supports Platform-dependent Rule Engines as Execution Environments. Each agent service might run one or more arbitrary rule engines to execute the interchanged queries, rules and events and derive answers on requests and reactions on detected events.

The complete request pattern starting from the External Agent has been depicted in Figure 5. OA intercepts the query at the boundary of a VO and redirects it towards the corresponding PA. Rule Responder architecture supports various multi-agent communication protocols including Agent Communication Language (ACL) [25]. The trust model facilitates the distributed query

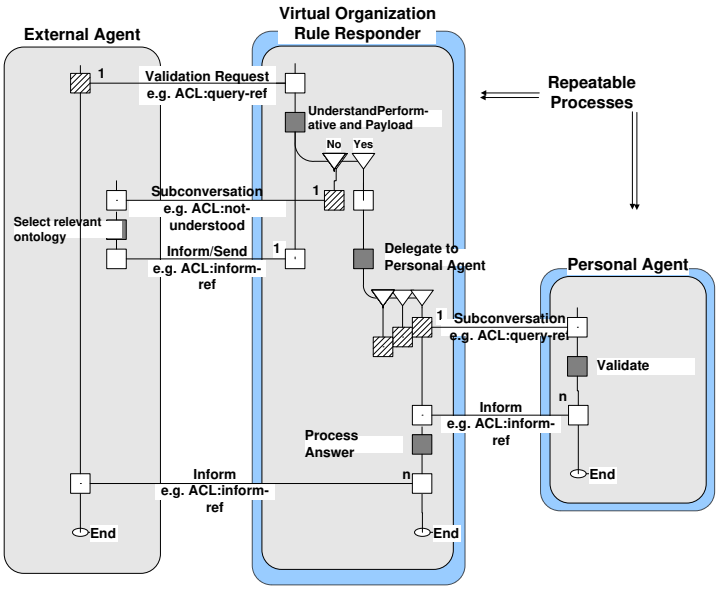

Fig. 5 Role Activity Diagram for a simple Query-Answer Conversation

to travel across various domains through a single sign-on and delegation mechanism. Referring to this multi-agent architecture coupled with the notion of SLA Views and the distributed trust, the validation process is termed as the *Delegation of Validation*.

4 Delegation of Validation

The aggregation of SLAs is a distributed mechanism and the aggregation information is scattered throughout the SLA choreography across various SLA views. To be able to validate the complete SLA aggregation, the validation query is required to traverse through all the SLA views lying across heterogeneous administrative domains and get validated locally at each SLA view. The multi-agent architecture of Rule Responder provides communication middleware to the distributed stakeholders namely the client, the VOs and various service providers. The *Delegation of Validation* process empowered by the *single sign-on and delegation* properties of the distributed trust model, helps the distribute query mechanism to operate seamlessly across different administrative domains.

Now we explain how the Guarantee Terms from a WS-Agreement, expressed as rules, are transformed into distributed queries. We discussed in the section 2.3 how the aggregation functions are applied on the basis of aggregation type of a service term, identified by $type_a$ attribute. SLOs can also be aggregated as conjunctive premises of derivation rules. It is also important to

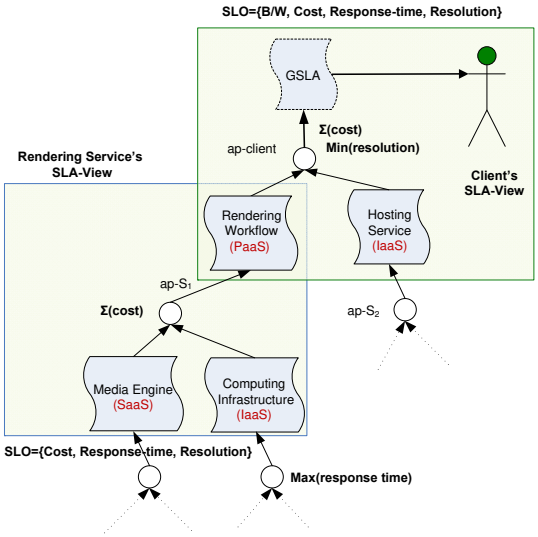

Fig. 6 Example Scenario for SLA Views

realize that the SLOs refer to an established SLA and their ranges are meant to be guarded in order to maintain desired levels of service.

In this section we will present a motivational scenario (see Figure 6) about a dynamic workflow composition based on the aggregation of Cloud based service. Arfa is a graphics designer and she has just finished designing an animation involving thousands of high resolution images. Now she needs to carry out hi-tech multi-media operations such as rendering and editing. She plans to utilize online services to accomplish these tasks. A media workflow service allows Arfa to define a series of activities involving video rendering, compression etc. Afterwards she would like to host the final compressed video on a dedicated server to visualize the output of her work. From Arfa's viewpoint, she is only composing two workflow services, i.e. "rendering workflow" service and the "hosting service" identified as the "Platform as a Service" (PaaS) and the "Infrastructure as a Service" (IaaS) respectively by her Cloud based service providers. What Arfa's View does not cover is the fact that both of these services are themselves result of an aggregation of even more basic services thus extending a supply chain type of structure beneath them. The rendering workflow subdivides into services such as the "media engine" and the "computing infrastructure" provided by different service providers in a public Cloud. On further investigation it may be revealed that the "media engine" is composed of even more basic services such as the "graphics engine" the "sound engine", and the computing Infrastructure too resells different qualities of computing services with varying response times and calculation speeds thus the list goes on. The SLA-Choreography resulting from this simple scenario is shown in Figure 6. There are two services, namely the rendering

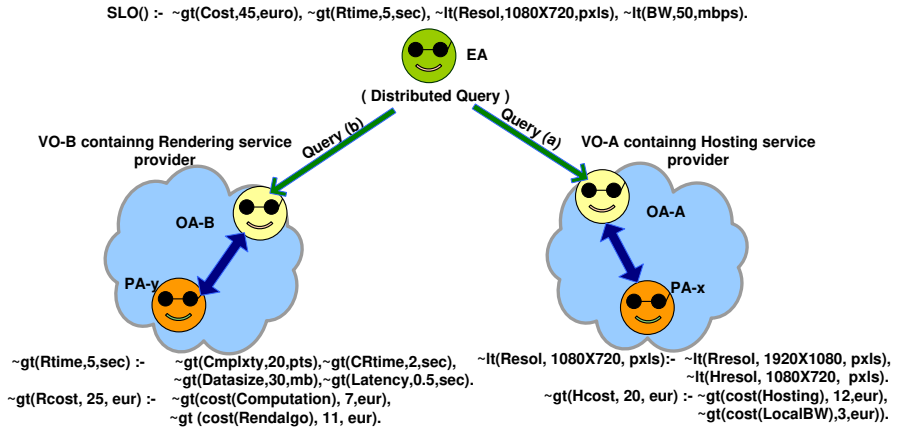

Fig. 7 Validation through distributed query decomposition

workflow service and the hosting service. The host video service downloads the video from a specified location, archives it and makes it available online. An authenticated user can play the video in a YouTube like style.

In the 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 1080X720 pixels and the minimum bandwidth (from hosting service) of 50 Mbps.

In the Figure 7, we have depicted this scenario from validation point of view. The user requirements are shown on the top of the figure, expressed as a derivation rule composed of SLOs of the final aggregated SLA. It must be noted that in Figure 7, we have intentionally chosen to represent these rules in a highly abstract format. This is only for the convenience of reading and comprehension. However later in this section we also explain how to formally represent and implement these rules.

The agents OA and PA in the Figure 7 representing the Rule Responder architecture, are shown to automate the distributed query processing. For the sake of simplicity, we have outlined the Rule Responder architecture just from agent-oriented perspective, and have abstracted various essential details such as the Rule-bases, the knowledge resources and the role of Enterprize Service Bus (ESB). The predicates *lt* and *gt* denote lesser-than and greater-than respectively. The user requirements are expressed as a set of premises in the following derivation rule:

```
SLO() :- ~gt(Cost,45,euro), ~gt(Rtime,5,sec),
         ~lt(BW,50,mbps),   ~lt(Resol,1080X720,pxls).
```

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. During the validation process, this rule will be decomposed such that each premise will become a subgoal. This subgoal will be sent as a message to the PA corresponding to the next SLA view in the hierarchy

where it will emerge as a conclusion of one of the rules in the local rule set, thus forming a distributed rule chain. The initial steps of decomposition procedure are depicted at the bottom of the figure. In the figure, Organizational Agents (OA) have been shown to receive and track the distributed query whenever it enters a new VO. For each service provider, there is a Personal Agent (PA). A PA, after finishing its job, should report to the corresponding OA that will redirect the distributed query to the service provider's PA that comes next in the hierarchical chain. The process continues until the query has found all the goals expressed in terms of logical rules. Active rules tracking these goals or SLOs, are then invoked locally within the administrative domains of the corresponding SLA views. The true or false results are conveyed back following the same routes.

To validate all the guarantee terms of the final (client's) aggregated SLA, the aggregation chunks within all the SLA Views, scattered through the whole SLA Choreography, are required to be validated. In our scenario, OA-B receives a subgoal $\sim gt(Rtime, 5, sec)$ representing the requirement that the total response time of the system should not be more than 5 seconds. This SLO depends on several factors such as the complexity of the rendering algorithm, size of the data, latency and response time of the computational hardware which is expressed as the new subgoal:

```
~gt(Rtime,5,sec) :- ~gt(Cmplxty,20,pts), ~gt(CRtime,2,sec),
                    ~gt(Datasize,30,mb), ~gt(Latency,0.5,sec).
```

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 OA-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 OA-A has a profit margin of 5 Euros.

As we have discussed earlier, the rules shown in the Figure 7 have been highly abstracted for reading convenience. In practice, we need to take into consideration many additional details. To highlight these issues, we begin with the formal representation of the SLO state that CRtime should be less then 2 seconds:

```
slo(Serv2, CRtime, <2, sec)
```

Serv2 is the name of the service with which this SLO is associated. Every SLO must have a reference point similar to Serv2. This particular SLO represents a state that is initiated if there is an event CRtime, which is a variable bound to a measurement value, which is greater then 2 seconds:

```
 initiates(CRtime, slo(Serv2,CRtime,<2,sec), T) :- CRtime < 2.
terminates(CRtime, slo(Serv2,CRtime,<2,sec), T) :- CRtime >= 2.
```

These two lines describe the initiation and termination of the SLO state. The SLO itself is associated with a specific service Serv2 and describes the user's requirement that the response time of the service should not exceed 2 seconds.

In other words, if the response time is lower than 2 seconds, the SLO is fulfilled, if it is greater than 2 seconds, the SLO is violated. The event is the measurement of CRTime at a particular time point such as:

```
happens(CRtime,T):- sysTime(T), ping(Serv2,CRtime).
```

Since CRtime is an event, we need to make it happen. In this case, in the happens rule we simply measure the response time in terms of systems time lapsed by pinging the service. It is now possible to ask queries if the SLO state holds at a particular point in time or not (i.e., violation of the SLO):

```
holdsAt(slo(Serv2,CRtime,<2,sec), 2001-10-26T21:32:52.12679)?
```

The result is true or false depending on the measurement result in the happens event rule. It is now possible to define SLO state processing rules such as SLO Rtime, which is the response time of CRtime.

```
holdsAt(slo(Serv1,Rtime,<5,sec),T) :-
                          holdsAt(slo(Serv2,Cmplxty,<20,pts),T),
                          holdsAt(slo(Serv2,Datasize,<30,mb),T),
                          holdsAt(slo(Serv2,Latency,<0.5,sec),T),
                          holdsAt(slo(Serv2,CRtime,<2,sec), T).
```

and ask if this derived SLO it violated at a certain point in time,

```
not(holdsAt(slo(Serv1,Rtime,<5,sec), 2001-10-26T21:32:52.12679))?
```

The delegation of validation, continuing across various levels, reaches the SLA views originating the corresponding SLOs, and the SLOs get validated there. At each level, the corresponding reward and penalty conditions are also checked and if required, appropriate action is taken. The distributed Rule Responder agent architecture acts as an enabling technology for the SLA Views concept. One of its important features is that we can implement principles of autonomy, information hiding and privacy with the agent approach. For instance, the details how a particular service level objective is measured and computed in a personal agent might be hidden (e.g. a third-party monitoring service) and only the result if the service level is met or not might be revealed to the public. Another important aspect is that the monitoring/validation might run in parallel, i.e. several service provider (PAs) might be queried by an OA in parallel via messaging. For instance, a complex SLOs might be decomposed by the OA into several subgoals which are then sent in parallel to the different services (PAs) which validate them.

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 named “Serv7” is larger than 60 seconds then there is a penalty of 5 Euros”, we can write its equivalent in WS-Agreement as follows:

```
<wsag:Penalty>
  <wsag:AssesmentInterval>
    <wsag:TimeInterval> 60
```

```
    </wsag:TimeInteval>
    <wsag:Count> 1 </wsag:Count>
  </wsag:AssesmentInterval>
  <wsag:ValueUnit> Eur </wsag:ValueUnit>
  <wsag:ValueExpr> 5 </wsag:ValueExpr>
</wsag:Penalty>
```

This can also be represented by ECA rules:

```
timer(sec,T) :- Timer(T), interval(1,min).
event(Serv7,Violate) :- ping(Serv7,RT), RT>60.
action(Serv7,Penalty) :- penalty(Serv7,Obligation,5).
```

Now combining together and generalizing for any service x:

```
ECA(?x, Monitor) :- timer(sec,T),
                    event(?x,violate),
                    action(?x,penalty).
```

The above rule is activated according to the timer(sec, T) which is defined by the following rule, invoked after every minute:

```
timer(sec,T) :- Timer(T), interval(1,min).
```

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 combination of ECA rules and using derivation rules to implement the different parts of an ECA rule provides high expressiveness and can be very easily transformed in a rule based markup language such as RuleML [15]. RuleML allows to declaratively implement the functionality of each part of a Reaction Rule (event, condition, action etc.) in terms of derivation rule sets (with rule chaining), thus making them processable in autonomic and autonomous way.

5 Related Work

The concept of Workflow Views is utilized to maintain the balance between trust and security among business partners [26]. Schulz et al [4] have introduced the concept of view based coalition workflows. Chiu et al [27] 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 [26,28], however, do propose a global view or a decomposition process based on the views. But none of them have focused on the dynamic workflows in their approach. Chiu et al [29] describe a contract model based on workflow views. They construct an e-contract model that defines e-contracts in plain text format. Static and dynamic verification of temporal constraints

[30,31] is very crucial in workflows to avoid any temporal violations during the workflow life cycle. Eder et al [32] employ the concept of views to calculate the temporal consistency of interorganizational workflows by using abstraction and aggregation operators of views but their approach is also limited to static or predefined workflows.

A little research has been carried out towards dynamic SLA composition of workflows [3,33,34]. The research area corresponding to the management of such aggregated SLAs is still wide open. Ganna Frankova [33] has highlighted the importance of this issue but she has just described her vision instead of any concrete model.

RBSLA [12] transforms SLAs into logical rules to automate their management and monitoring. The authors discuss knowledge representation of SLAs with complex business rules and policies. RBSLA [19,12] 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. Oldham et al [35] have extended WS-Agreement by building a rule based ontology on the WS-Agreement. Their SWAPS schema [35] transforms constructs from the Guarantee terms into predicate based markup language. They admit that their schema is limited to a specific domain.

The Grid Security Infrastructure (GSI) and the security modules of middleware, provide a set of security protocols for achieving mutual entity authentication between a user (actually a user's proxy) and resource providers [22]. GSI uses X.509 proxy certificates (PCs) to enable Single sign-on and Delegation [21].

6 Conclusion and Future Work

In this paper, we presented the design of a validation framework for hierarchical SLA aggregations corresponding to cross-VO workflow compositions. This rule based validation framework employs a top-down validation mechanism based on distributed query processing. The validation framework assumes the hierarchical aggregation of SLAs [7]. The validation framework also assumes unique consumers for the providers of a value chain in the hierarchy. In the future, we plan to implement the distributed rule based validation system based on RuleML, through iterative development phases, and adhering to the WS-Agreements standard.

References

1. Ludwig, A.: COSMA -An Approach for Managing SLAs in Composite Services. In: Lecture Notes in Computer Science, Springer Berlin-Heidelberg (2008)
2. Buyyaa, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I.: Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility . Future Generation Computer Systems, Volume **25** (2010) 599–616

3. Blake, M.B., Cunnings, D.J.: Workflow composition of service level agreements, International Conference on Services Computing (SCC2007) (2007)
4. Schulz, K.A., Orlowska, M.E.: Facilitating cross-organisational workflows with a workflow view approache. Data and Knowledge Engineering **51** (2004) 109–147
5. SLA@SOI: http://www.sla-at-soi.org/index.html (12 March 2009)
6. NESSI-Grid: http://www.soi-nwg.org/doku.php?id=sra:description (last access: March12, 2009)
7. Haq, I.U., Huqqani, A.A., Schikuta, E.: Aggregating Hierarchical Service Level Agreements in Business Value Networks. In: Lecture Notes in Computer Science. Volume Volume 5701/2009., Springer Berlin-Heidelberg (2009) 176–192
8. Chebbi, I., Dustdar, S., Tata, S.: The view based approach to dynamic inter-organizational workflow cooperation. Data and Knowledge Engineering **56** (2006) 139–173
9. Liu, D.R., Shen, M.: Workflow modeling for virtual processes: an order-preserving process-view approach. Information Systems **28** (2002) 505–532
10. Haq, I.U., Huqqani, A.A., Schikuta, E.: A conceptual model for aggregation and validation of SLAs in Business Value Networks, accepted in the 3rd International Conference on Adaptive Business Information Systems , Leipzig, Germany (2009)
11. Paschke, A., Boley, H., Kozlenkov, A., Craig, B.: Rule responder: RuleML-based agents for distributed collaboration on the pragmatic web, Proceedings of the 2nd international conference on Pragmatic web Tilburg, The Netherlands (2007)
12. Paschke, A., Bichler, M.: Knowledge representation concepts for automated SLA management. Int. Journal of Decision Support Systems (DSS) (March 2006)
13. Paschke, A., Harold, B., Kozlenkov, A., Craig, B.: Rule Responder: A RuleML-Based Pragmatic Agent Web for Collaborative Teams and Virtual Organizations, http://ibis.in.tum.de/projects/paw/ (2007)
14. Paschke, A.: Rule-Based Service Level Agreements - Knowledge Representation for Automated e-Contract, SLA and Policy Management. Idea Verlag GmbH, Munich (2007)
15. Boley, H.: The Rule-ML Family of Web Rule Languages. In: 4th Int. Workshop on Principles and Practice of Semantic Web Reasoning, Budva, Montenegro (2006)
16. Mule: Mule Enterprise Service Bus, http://mule.codehaus.org/display/MULE/Home (2006)
17. Ball, M., Boley, H., Hirtle, D., Mei, J., Spencer, B.: The OO jDrew Reference Implementation of RuleML. In: RuleML 2005, Galway (2005)
18. Roo, J.D.: Euler proof mechanism.
19. Paschke, A., Bichler, M.: SLA representation management and enforcement, The 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (2005)
20. Wang, M., Kotagiri, R., Chen, J.: Trust-based robust scheduling and runtime adaptation of scientific workflow. Concurrency and Computation: Practice and Experience **21(16)** (2009) 1982–1998
21. Lioy, A., M.Marian, N.Moltchanova, Pala, M.: PKI past, present and future. International Journal of Information Security, Springer Berlin (pages 1829) 2006
22. Zhao, S., Aggarwal, A., Kent, R.D.: PKI-based authentication mechanisms in grid systems, International Conference on Networking, Architecture, and Storage (2007)
23. Haq, I.U., Alnemr, R., Paschke, A., Schikuta, E., Boley, H., Meinel, C.: Distributed trust management for validating sla choreographies. In: SLAs in Grids workshop, CoreGRID Springer series. (2009)
24. Ludwig et al: Web Service Agreement (WS-Agreement). GFD.107 proposed recommendation (last access: July 12, 2008)
25. FIPA: FIPA Agent Communication Language, http://www.fipa.org/, accessed Dec. 2001 (2000)
26. Shen, M., Liu, D.R.: Discovering role-relevant process-views for disseminating process knowledge. Expert Systems with Applications **26** (2004) 301–310
27. Chiu, D., Cheung, S., Till, S., Karalapalem, K., Li, Q., Kafeza, E.: Workflow view driven cross-organisational interoperability in a web service environment. Information Technology and Management **5** (2004) 221–250

28. Li, Q., Chiu, D., Shan, Z., Hung, P., Cheung, S.: Flows and views for scalable scientific process integration. In: First International Conference on Scalable Information Systems, Hong Kong. (2006)
29. Chiu, D., Li, K.K.Q., Kafeza, E.: Workflow view based e-contracts in a cross-organisational e-services environment. Distributed and Parallel Databases **12** (2002) 193–216
30. Chen, J., Yang, Y.: Activity completion duration based checkpoint selection for dynamic verification of temporal constraints in grid workflow. International Journal of High Performance Computing Applications, **319-329** (2008) 22(3)
31. Chen, J., Yang, Y.: Temporal dependency based checkpoint selection for dynamic verification of temporal constraints in scientific workflow systems. In: accepted in ACM Transactions on Software Engineering and Methodology. (2009)
32. Eder, J., Tahamatan, A.: Temporal consistency of view based interorganizational workflows, 2nd International United Information Systems Conference, Austria (2008)
33. Frankova, G.: Service level agreements: Web services and security. Springer Verlag, Berlin Heidelberg (2007) 556–562
34. Unger, T., Leyman, F., Mauchart, S., Scheibler, T.: Aggregation of Service Level Agreement in the context of business processes, Enterprise Distributed Object Computing Conference (EDOC '08) Munich, Germany (2008)
35. Oldham, N., Verma, K., Sheth, A., Hakimpour, F.: Semantic WS-Agreement partner selection, Proceedings of the 15th international conference on World Wide Web, Edinburgh, Scotland (2006)