Lifetime Service Level Agreement Management with Autonomous Agents for Services Provision*

Qiang He[1, 3], Jun Yan[2], Ryszard Kowalczyk[1], Hai Jin[3], Yun Yang[1]
[1]*Faculty of Information and Communication Technologies, Swinburne University of Technology Australia*
qhe@ict.swin.edu.au, rkowalczyk@ict.swin.edu.au, yyang@ict.swin.edu.au
[2]*School of Information Systems and Technology, University of Wollongong, Australia*
jyan@uow.edu.au
[3]*School of Computer Science and Technology, Huazhong University of Science and Technology Wuhan, China*
hjin@hust.edu.cn

Abstract

In the Web services environment, Service Level Agreements (SLA) refers to mutually agreed understandings and expectations between service consumers and providers on the service provision. Although management of SLA is critical to wide adoption of Web services technologies in the real world, support for it is very limited nowadays, especially in Web service composition scenarios. There lacks adequate frameworks and technologies supporting various SLA operations such as SLA formation, enforcement, and recovery. This paper presents a novel agent-based framework which utilises the agents' ability of negotiation, interaction, and cooperation to facilitate autonomous SLA management in the context of service composition provision. Based on this framework, mechanisms for autonomous SLA operations are proposed and discussed. Results from simulations show that by integrating agents and Web services the framework can address issues of SLA management drawn from sophisticated service composition scenarios.

Keywords: Agent Technology, Service-Oriented Computing, Service Level Agreement Management, Web Services

1. Introduction

With the popularity of Service-Oriented Computing (SOC), Service Level Agreement (SLA) has become an active research topic. An SLA refers to the contractual obligations between a service consumer and a service provider, which can represent guarantees of Quality of Service (QoS), non-functional requirements of a service consumer and promises of a service provider. An SLA usually contains the following main components [2]:

1. Parties involved, including contracted parties and supporting third parties such as monitoring parties, auditing parties, etc;
2. Service description which specifies the functionality that will be delivered under the agreement. Domain specific information can be included to adapt to various particular domains;
3. Service level objectives, which define the service level indicators (specifying what aspects of the service is guaranteed) and corresponding promised values of the services, such as the response time and the availability; and
4. Penalty in case that the service provider underperforms or is unable to provide the service on the promised level. Violation of different guaranteed service objectives may lead to different penalties;

SLAs can be used by potential service consumers to choose the service providers which use the *SLAs as advertisements. At the same time, by introducing the SLA management system, matured

* This is an extended version of an earlier paper appeared in Proceedings of the 11th International Conference on Computer Supported Cooperative Work in Design (CSCWD'07), Melbourne, Australia, April, 2007.

service providers could consolidate their clients, and newly-rising service providers could attract potential undeveloped clients.

In the sophisticated Web services environment, SLA management is challenging since the relationships among clients and service providers are complicated. Web services are designed to provide transparent automated utilisation of heterogeneous resources and applications. Minimised human intervention is desirable in every function of Web services, which is also true for SLA management. The automated formation of SLA requires precise and unambiguous definition of the agreement as well as customisable engines to support automated negotiation over the details of the agreement for both contracting parties. Contracted SLAs would potentially be violated due to the inherent unreliability of the underlying Internet and internal infrastructures of service providers. Therefore, to ensure that a service provider is adherent to its promised service level guarantees, automated support for SLA monitoring, including state measurement and compliance verification, is required. Furthermore, some additional operations are also desirable, such as credible SLA profiling for reputation evaluation of the service providers and automatic SLA recovery in case of failure of the service provision.

To address some of these issues, this paper proposes a novel SLA management framework which integrates agent and Web services technologies to support autonomous management of SLA. Based on this framework, mechanisms for operations, including SLA formation, enforcement and recovery, are addressed.

The remainder of the paper is organised as follows. In Section 2, the requirements of SLA management are analysed with a motivating example. Section 3 presents major related work. Section 4 presents the details of the proposed framework, while Section 5 presents the operational mechanisms that support the framework. Section 6 presents the simulation work. Finally, Section 7 concludes the paper and outlines items for future work.

2. Requirements Analysis with Motivating Scenario

One of the distinctive features of SOC is its ability to compose existing services (i.e., component services) with little effort into a network of services in order to dynamically create new and value-added services (i.e., composite services) [3][20]. This is very powerful to solve complex problems, as it is increasingly difficult to have a single specific service provider to provide a complete solution. In a service composition application, the component services are usually provided by a group of service providers. Therefore, SLA management in such a scenario becomes very complex, as it involves management of relationships among one service consumer and many service providers. To illustrate the motivations of this research, a simplified goods delivery service is presented and analysed in this section.

This goods delivery service involves a customer and five logistics companies. The goods are required to be delivered from Sydney to Darwin. In order to obtain a reasonable price for the delivery, the surface transportation is selected. The goods will traverse Sydney, Canberra, Melbourne, Adelaide, Perth and finally reach Darwin. Each leg of the delivery may be undertaken by a different logistics company providing local delivery services.

Generally speaking, two of the customer's major concerns are the time consumption and the cost of the global delivery process. In this scenario, the time consumption and the cost are from five local deliveries which compose the global delivery. The customer expects to receive the goods in an acceptable time frame at a reasonable price. Thus, the customer needs to control the individual time consumption and the cost of each local delivery service in order to ensure that the total time consumption and the cost are within the required range. These values will be unambiguously negotiated and defined over in the SLAs between the customer (as service consumer) and the logistics companies (as service providers). Finally, five SLAs defining the quality of the local delivery services will be contracted between the customer and the logistics companies. Other negotiable parameters can be dealt with in the same way.

Based on this motivating example, the following critical issues of SLA management are identified. Web services are various, as well as their properties which are specified as parameters. Service providers should present the parameters of the services in the advertisement in a formal manner for the service consumers to identify, compare and select the service providers. In the goods delivery example, the customer needs to select the service providers based on the advertised time consumption and prices of the delivery services which are presented in a uniform and comparable way. Thus, the first issue is the standardised approach to describe the quality of a service, (i.e., service level) which is accepted by the involved entities, i.e., the service consumer and the service provider. Then the SLA operations can be performed, such as formation, enforcement and recovery. Usually, the service consumers prefer to describe their service requirements in a domain-specific way. Thus, the automatic mapping from the service requirements of the service consumer to a standardised service level description should be supported so that it will not become a technical barrier to the service consumers. WSDL (Web Services Description Language) [10] provides an approach to establish the fundamental description of the Web services, including the locations (where to invoke the Web services) and the operations (what the Web services can do). However, WSDL is insufficient to describe the Web services as commodities because it has nothing to do with the quality of Web services.

In the delivery service example, the quality of local delivery services should be determined in advance in order to guarantee the customer's global QoS requirements which can be computed by aggregating the QoS of the component services [15]. The customer needs to negotiate SLAs with the logistics companies over each leg of the delivery to satisfy its global time and price constraints. Such negotiation is complicated and time consuming because it involves sophisticated interactions among the involved parties. This leads to the second issue, that is, for the service composition to be efficient, related SLA operations, such as SLA negotiation, must be automated.

The third issue is the persistent SLA management. Due to the dynamically changing environment, possibilities of amendment and recovery of SLAs are inevitable. There are three major exceptions that may occur during service provision: (1) the execution of a service fails to achieve the expected outcome; (2) a contracted service provider is no longer capable of providing the service; and (3) the contracted SLA is broken because the QoS guarantees are violated. To deal with the exceptions, the service consumer may need to renegotiate over the amendment of the SLA with the contracted service provider, or select a new service provider to recover the broken SLA. For example, when one of the contracted logistics companies cannot deliver the goods on time, renegotiation is needed to rearrange the local delivery. Alternatively, the customer could stop using the failed logistics company and immediately select another one to carry out the delivery service.

The fourth issue is caused by the requirement of automatic SLA operations. On the one hand, the customers and the domain-specific applications used usually do not comprehend how to perform the SLA operations and actually they do not need to. All they concern about is that the services they consume conform to the level promised by the service providers. So the SLA operations should be performed by someone autonomous on behalf of the customers. On the other hand, the Web services are passive and static until invoked because they only expose predefined interfaces for service consumers to access [14]. Some of the SLA operations cannot be directly performed by Web services. For example, the customer needs to negotiate the SLAs with someone who knows the scheduling of the logistics companies and can autonomously make decision on their behalf. That is, autonomous representatives of the service consumers and the service providers with the essential knowledge of the SLA operations and the Web services should take the responsibility of performing the SLA operations.

To effectively address the above issues, an autonomous SLA management framework is needed. Within such a framework, all the entities are self-organised and autonomous. In the goods delivery example, after the customer enters its requirements for the time consumption and the price of the delivery service, all the SLA operations should be performed automatically.

3. Related Work

Over the past years, SLA-related research has attracted growing attention from both industry and academia. Significant effort has been placed on specification of SLA and a number of approaches have been released. WS-Policy [24] defines a model to express the static properties of Web services entities, such as security, privacy and authentication details, enabling service consumers to select Web services more meaningfully. WSPL [1] implements more complicated operations on policies and a rich set of comparison operators rather than just simple equality matching. It also supports the merging of mutually accepted policies which enables simple match schema for service consumers and providers. WSLA [17] [19] adopts the template mechanism as the carrier for parties to rapidly and conveniently establish an agreement. It also provides a set of standard extensions that allow users to define agreements with guarantees for common metrics, e.g. response time, delay and throughput. WS-Agreement [2] aims at standardising the terminology, concepts, and the overall agreement structure. It provides a more expressive language than WSLA to support the truly complex nature of the relationship between a service consumer and a service provider. WS-Agreement specification is designed to be independent of negotiation models and management models so that it can be composed with various negotiation protocols.

Apart from SLA specification, research on SLA management has also been carried out. Focusing on standardising the representation of SLA, IBM proposes Cremona [18], which is a WS-Agreement based middleware that defines mechanisms to implement interactions between organisational domains. A simple two-step request-response schema is provided for SLA negotiation in Cremona. Similarly, [11] presents a model and protocol for negotiating SLA over accessing resources in distributed environments. Devoting to presenting the requirements of precision and flexibility for SLA specification [16] and analysing the SLA monitoring model from the XML-specific aspect [23], HP proposes an automated and distributed SLA monitoring engine that considers both provider side and client side measurement of SLA and deals with the scenarios where Web service providers work and contract with each other to fulfil the customer's request. [13] addresses applying provider policy and admission control of service requests onto the underlying network infrastructure via SLA negotiated by agents to facilitate adaptive coordination customers and service providers in the telecommunications domain. Since 2005, research has been carried out in using agent technology to manage the SLA in Grid and Web services environments. To name a few, [22] proposes a multi-agent infrastructure and an SLA negotiation protocol based on the Iterated Contract Net Protocol for job scheduling on the Grid, which divides the SLA negotiation into two levels, meta-SLA negotiation and sub-SLA negotiation. [8] and [9] describe an agent-based framework for SLA negotiation in Web services composition with end-to-end QoS constraints. On the basis of the framework, the decision-making mechanisms needed to facilitate QoS-constrained SLA negotiation are discussed in [6], including the decomposition of overall user preferences, the selection of negotiation partners and the generation of offers and counter-offers. Based on the same framework as [9], a utility-function-based decision making model is proposed in [26], which is used by the autonomous negotiation agents to evaluate the offers and make various negotiation decisions during SLA negotiation. These approaches have made contributions to the SLA management research. However, it is clear that comprehensive SLA management over the lifetime of SLA, including SLA formation, enforcement, recovery and so on, is still at its infancy.

4. Agent-based SLA Management Framework

It is recognised that the agent technology is a promising technology to address the issues identified in Section 2, as it offers the complementary abilities to the Web services technology, such as intelligent operation, negotiation and interaction, which support autonomous SLA operations such as SLA formation, enforcement, recovery and so on. With the ability to manage SLAs, agents can provide a coordination framework while Web services provide the resources [7]. In the goods delivery example described earlier, the customer submits its service requirements in the application system and waits until the SLAs are provided which satisfy the requirements of respective local delivery services. Then the customer confirms the SLAs and waits for the arrival of the goods. The customer has no concerns about how the SLAs are formed. When submitted, the requirements of the global delivery service are sent with a service request to the agent that is on behalf of the customer. The agent then maps the service requirements to a standardised SLA description, selects appropriate logistics companies from a service registry (where services and service providers are registered) and negotiates SLAs with them. In this way, the autonomous agents facilitate the automatic coordination of the resources and make the SLA operations transparent to the service consumers.

The SLA management framework proposed in this paper is a reusable architecture that can be utilised to produce customised applications. This framework identifies and represents major functionalities which would be commonly observed in various application domains. In the framework and the corresponding operational mechanisms, agents play two distinct roles, i.e., the SLA initiator and responder. An SLA initiator is an SLA requestor which initialises the SLA formation process when needed. In most cases, an SLA initiator would be the service consumer. In other cases, the service consumer delegates the request to a third party which subsequently becomes an SLA initiator. For example, in sophisticated scenarios where a service composition manager is needed to combine multiple Web services to fulfil a service request, the service composition manager acts as an SLA initiator on behalf of the service consumer. An SLA responder is normally a service provider which could announce its service level capability as an advertisement. The SLA initiators can fetch these advertisements in a service registry such as UDDI, or directly from the SLA responders. In the proposed framework, both the SLA initiator and responder are agents which consist of respective components. Automatic SLA operations such as formation, enforcement, and recovery can be fulfilled through autonomous interactions among these agents.

An SLA initiator comprises several components and interacts with several interfaces, as depicted in Figure 1.:

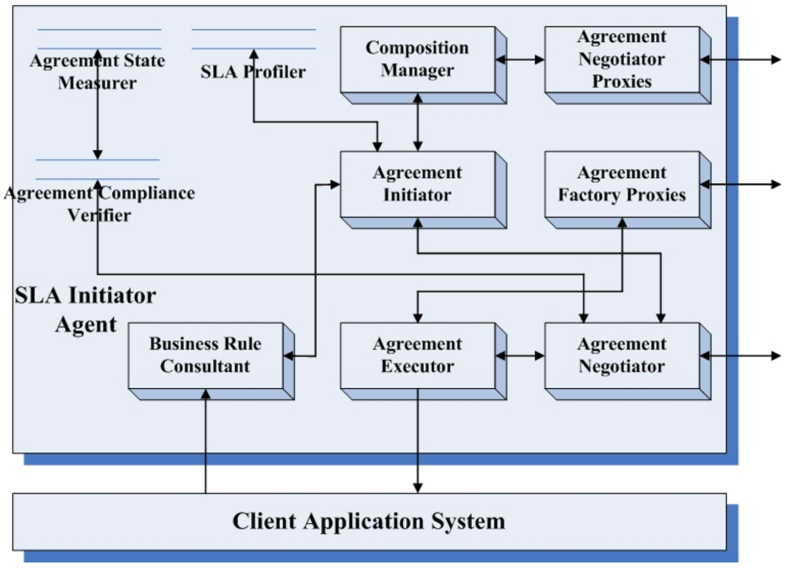


Fig. 1. Architecture of SLA initiator agent

- The Business Rule Consultant component interacts with the client application to map its requirements of the Web services to a standardised service description.
- The Agreement Initiator component initiates the SLA formation, by selecting appropriate service providers.
- The Agreement Negotiator component negotiates the SLA with the SLA responder's Agreement Negotiator component over the detailed parameters. It also takes the responsibility of renegotiation when SLA recovery is needed.
- The Agreement Executor component announces the client application system to enforce the agreement.
- The Agreement Factory Proxies component contains a set of references to the remote SLA responders' Agreement Factory components.
- The Agreement Negotiator Proxies component contains a set of references to the remote SLA responders' Agreement Negotiator components. It sends the agreements received from the SLA responder to the Agreement Executor component.
- The Composition Manager component interacts with the Agreement Initiator component and multiple Agreement Negotiator components of the SLA responders to manage the Web service composition.
- The SLA Profiler interface is usually implemented by a supporting third party where historical data of services provision are stored and analysed. The SLA initiator can use this interface to report the runtime information and evaluation of the service provisions for profiling. This interface can also be used by the Agreement initiator component to retrieve information about service providers, such as historical performance, predicted performance and reputation information.
- The Agreement State Measurer interface could be implemented by the SLA initiator itself, or by supporting third parties to compute and measure the runtime state of the service provision

according to the agreement.

- The Agreement Compliance Verifier interface also has two potential implementers, the SLA initiator or supporting third parties. It verifies whether the service provision conforms to the agreement based on the result of the measurement performed by the Agreement State Measurer interface. When incompliance occurs, it notifies the Agreement Negotiator component.

An SLA responder agent also comprises several components and interacts with several interfaces, as shown in Figure 2:

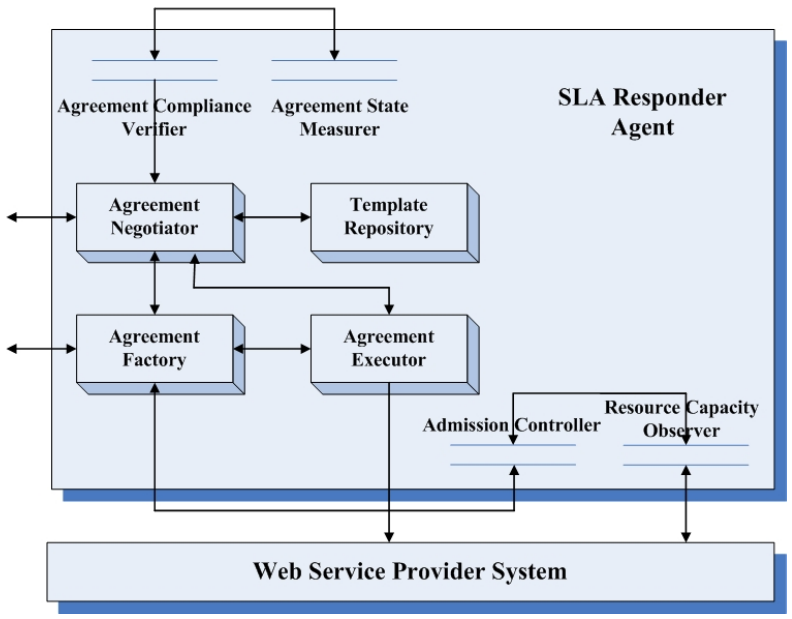


Fig. 2. Architecture of SLA responder agent

- The Template Repository component stores the predefined SLA templates for services at different promised levels.
- The Agreement Negotiator component, similar to that of SLA initiator's, performs the SLA negotiation on behalf of the SLA responder.
- The Agreement Factory component processes the agreement with static and dynamic information, such as party information and agreed SLA parameters, provided by the Template Repository component and the Agreement Negotiator component.
- The Agreement Executor, component is similar to that of SLA initiator's, but deals with the Web service provider system.
- The Resource Capacity Observer interface, usually implemented by the Web service provider system, verifies the resource capacity and reports to the Admission Controller.
- The Admission Controller interface, usually implemented by the Web service provider system, decides whether a service request could be fulfilled based on verifying current resource capacity information provided by the Resource Capacity Observer interface.
- The Agreement State Measurer interface, the same as of SLA initiator's, is in charge of computing and measuring the runtime service provision according to the agreement.
- The Agreement Compliance Verifier interface, which performs similarly as SLA initiator's, verifies if the service provision conforms to the agreement. When incompliance occurs, it triggers the SLA recovery mechanism by notifying the Agreement Negotiator component of the incompliance.

Based on this framework, corresponding operational design is detailed next in Section 5.

5. Operational Design

5.1 Lifecycle of the SLA

Automatic and comprehensive SLA management over the lifetime of SLA is essential for the entire service provision procedure. The lifecycle of an SLA is depicted in Figure 3. With the service requirements provided to the SLA initiator agent, the SLA formation will start with mapping the service requirements to a standardised service description. Then the SLA initiator selects an appropriate service provider and negotiates the SLA with it. Or, the SLA initiator can negotiate with multiple candidate service providers at the same time and select the one that provides the best service. For service composition, an SLA initiator needs to negotiate the SLAs with multiple service providers over respective component services in order to fulfil its global service requirements. Once the negotiation succeeds, the SLA can be generated and enforced, which involves SLA monitoring and profiling. SLA monitoring is needed to verify if the service provisions are in line with the contracted SLAs. SLA monitoring includes the measurement of the metrics of the service provision and the evaluation of the compliance of the contracted SLA. SLA profiling includes collecting runtime information of the service provision and evaluating the compliance of the SLA for establishing the reputation system of the service providers. Usually, SLA monitoring and profiling are commissioned to supporting third parties due to authority and credibility issues. If exceptions occur during the service provision, the SLA recovery mechanism will be triggered. The recovery depends on the particular situation of the exception. The SLA initiator may either renegotiate over the amendment of the original SLA with the SLA responder, or select and negotiate with another service provider to replace the failed one, which will actually start another SLA formation. When the service provision is finally completed, the SLA enforcement will be completed to end the lifecycle of an SLA.

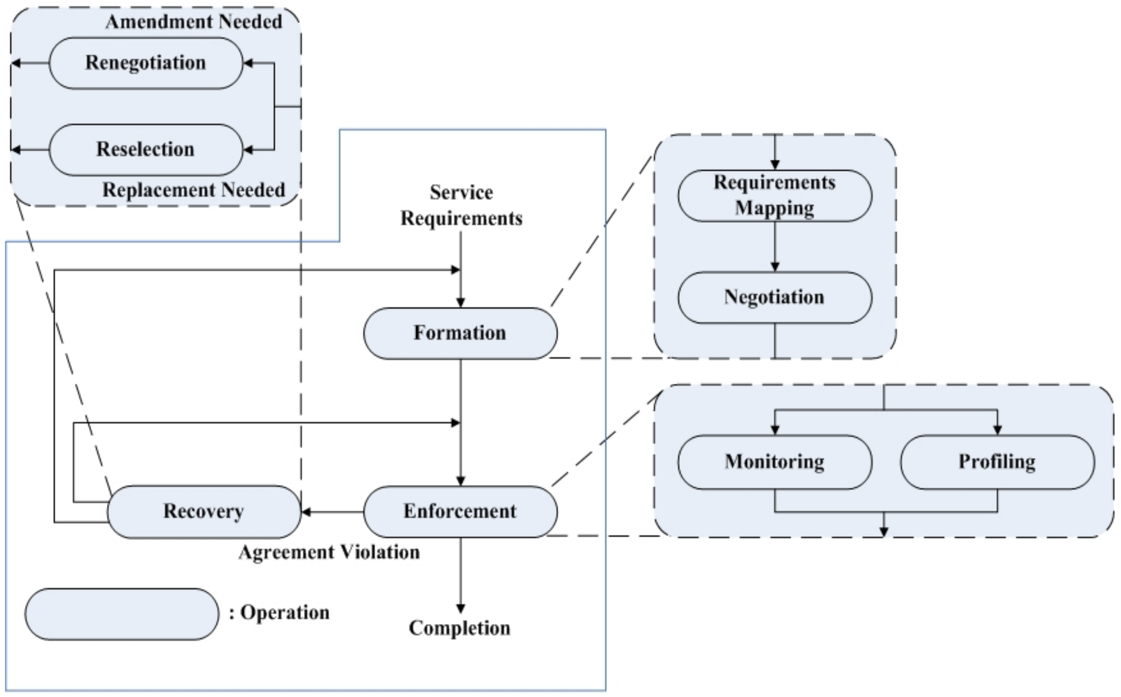


Fig. 3. Lifecycle of SLA

5.2 SLA Specification

Different Web services description schemas, such as WS-Policy, WSPL, WSLA and WS-Agreement, present different advantages and also reveal respective limitations. Based on the comparison of these specifications, WS-Agreement is used in this research, as it is the most recent and comprehensive approach. WS-Agreement defines a language for advertising the capabilities of service providers, monitoring the service provision, evaluating compliance and creating agreements. It is flexible, scalable and expressive in SLA representation so that it can be conveniently extended for a wide range of applications by integrating domain-specific service descriptions. Figure 4 illustrates a sample SLA in accordance with WS-Agreement, for the local delivery service from Sydney to Canberra, which is the first leg of the global delivery described in Section 2. This sample agreement specifies that the goods must be delivered within 2 days at the price of 30 Australian dollars. Agreements for other local delivery services are omitted due to space limit. Full definition of the corresponding domain-specific XML schema used to integrate delivery-specific information into the agreements can be found in Appendix.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsag:Agreement xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:wsag="http://schemas.ggf.org/graap/2007/03/ws-agreement"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:gps="http://www.ict.swin.edu.au/namespaces/gps"
   wsag:AgreementId="GoodsDeliveryExample123">
   <wsag:Terms>
      <wsag:All>
         <wsag:ServiceDescriptionTerm wsag:Name="GoodsTypeTerm"
            wsag:ServiceName="GoodsDeliveryExample">
            <gps:GoodsType>Blanket</gps:GoodsType>
         </wsag:ServiceDescriptionTerm>
         <wsag:ServiceDescriptionTerm wsag:Name="GoodsQuantityTerm"
            wsag:ServiceName="GoodsDeliveryExample">
            <gps:GoodsQuantity Unit="kg">500</gps:GoodsQuantity>
         </wsag:ServiceDescriptionTerm>
         <wsag:ServiceDescriptionTerm wsag:Name="PriceTerm"
            wsag:ServiceName="GoodsDeliveryExample">
            <gps:Price Currency="AUD" Unit="Dollar">30</gps:Price>
         </wsag:ServiceDescriptionTerm>
         <wsag:ServiceDescriptionTerm wsag:Name="TimeConsumptionTerm"
            wsag:ServiceName="GoodsDeliveryExample">
            <gps:TimeConsumption Unit="Day">2</gps:TimeConsumption>
         </wsag:ServiceDescriptionTerm>
         <wsag:ServiceDescriptionTerm wsag:Name="DeliverySourceTerm"
            wsag:ServiceName="GoodsDeliveryExample">
            <gps:DeliverySource>Sydney</gps:DeliverySource>
         </wsag:ServiceDescriptionTerm>
         <wsag:ServiceDescriptionTerm wsag:Name="DeliveryDestinationTerm"
            wsag:ServiceName="GoodsDeliveryExample">
            <gps:DeliveryDestination>Canberra</gps:DeliveryDestination>
         </wsag:ServiceDescriptionTerm>
      </wsag:All>
   </wsag:Terms>
</wsag:Agreement>
```

Fig. 4. Sample SLA for goods delivery service

If there are multiple SLA specifications available, agents can communicate with each other beforehand to achieve an agreement over which SLA specification is adopted. The standardisation of the SLA specification is beyond the scope of this paper.

5.3 Operational Mechanisms

5.3.1 SLA Formation

SLA formation usually starts with the SLA initiator retrieving a predefined agreement template from either the service provider or the service registry where the service providers registered their services and corresponding agreement templates. The template contains fixed, predetermined information, and negotiable elements, as well as constraints for the SLA initiator to follow when creating the agreement. Then the SLA initiator fills in the blanks in the template according to its practical service requirements and sends the template to the SLA responder. The SLA responder could choose to accept the agreement or further amend the values in the template and send back the template to the SLA initiator, which starts the negotiation. The negotiation will continue until an agreement is reached or the time constraint of the negotiation is violated.

When the SLA formation starts, on the SLA initiator's side, the domain specific client application system inputs service requirements into the Business Rule Consultant, which then maps the service requirements to a standardised service description and activates the Agreement Initiator to undertake the agreement negotiation by invoking function startAgreementFormation(). The Agreement Initiator accesses the service registry using function getServiceInformation() to retrieve the service information of the service providers, such as the service type and the service level guarantees. The templates for agreements may also be returned if the SLA initiator asked. Alternatively, the Agreement Initiator can ask the SLA responder for the templates by invoking getTemplates(). Based on the service information and SLA templates, the Agreement Initiator can find out the service providers it is most interested in. The selection depends on what the SLA initiator requires, as specified by the standardised service description provided by the Business Rule Consultant. When the selection is done, the Agreement Initiator sends a "MSG_STARTNEGOTIATION" message to the Agreement Negotiator, which subsequently sends a "MSG_REQUESTAGREEMENT" message to the SLA responder to start the negotiation. When the negotiation succeeds, the Agreement Executor will receive the notification, a "MSG_ENFORCEAGREEMENT" message, from the Agreement Negotiator and the finalised agreement instance from the Agreement Factory Proxies. Figure 5 illustrates the interactions among the components in the SLA initiator agent in the SLA formation process.

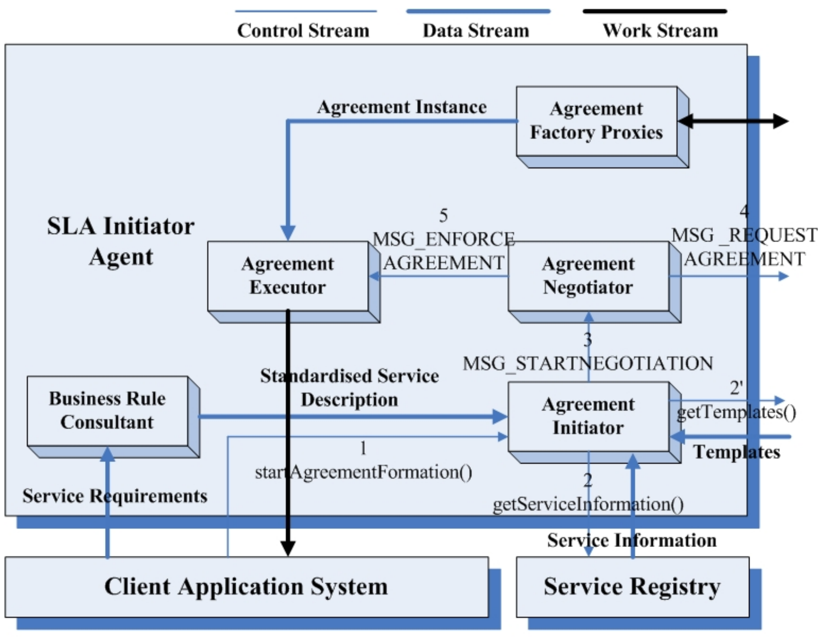


Fig. 5. Interactions in SLA initiator agent for SLA formation

On the SLA responder's side, upon receiving an agreement request, the Agreement Factory asks the Admission Controller whether the request can be accommodated by invoking function requestAdmission(). The Admission Controller makes the decision by comparing the agreement request with the current resource capacity returned by function getResourceAvailability() provided by the Resource Capacity Observer. If the requirements cannot be met, a "MSG_REQUESTREJECTED" message is sent to the SLA initiator. Otherwise, the Agreement Factory advises the Agreement Negotiator through a "MSG_STARTNEGOTIATION" message to start the negotiation with the SLA initiator. Subsequently, the negotiation starts with the Agreement Negotiator sending a "MSG_STARTNEGOTIATION" message to the SLA initiator. If the Agreement Negotiator succeeds in making a deal with the SLA initiator in the negotiation, it notifies the Agreement Factory to generate the agreement by sending a "MSG_GENERATEAGREEMENT" message. Finally, the Agreement Factory sends the agreement instance to the SLA initiator and notifies the Agreement Executor to enforce the agreement with a "MSG_ENFORCEAGREEMENT" message and a copy of the agreement instance. Figure 6 illustrates the interactions among the components in the SLA responder agent upon the receipt of an agreement request.

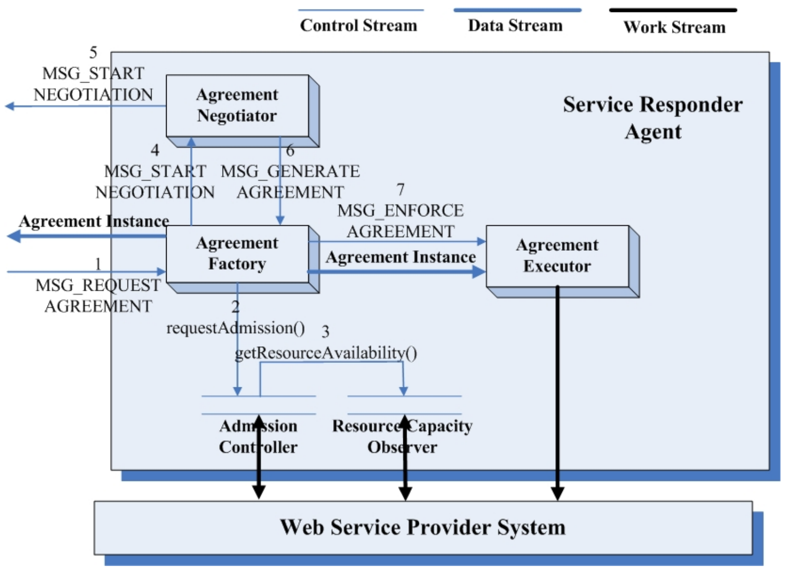


Fig. 6. Interactions in SLA responder agent for SLA formation

5.3.2 SLA Enforcement

To flexibly solve complex problems, the ability to integrate existing services from different service providers is required for the SOC environment. For such integrations and corresponding SLAs to be effective, monitoring and profiling of the SLAs are essential in the process of SLA enforcement. There are many auditing architectures that SLA monitoring can be based on [4]. However, due to the issue of credibility, commissioning SLA monitoring to an authorised supporting third party, such as AlertSite (http://www.AlertSite.com), is recommended. Similar to SLA monitoring, SLA profiling also has the credibility issue, and is supposed to be commissioned to a third party. The operations for SLA monitoring and profiling are out of the scope of the SLA initiator and responder's responsibility. Hence, only how SLA profiling is utilised in the framework will be discussed in this section.

The reputation of Web service providers is a desirable criterion for service consumers to judge and select the service providers [25]. The reputation of a service provider can be evaluated by judging its past performance. Therefore, in a practical environment, it is necessary to profile the performance of the service providers and to rank them. By doing so, the service consumers can select the service providers based on not only the advertised SLA templates but also the reputation information provided by authoritative SLA profiling organisations. Another significance of SLA profiling is that it is a feasible mechanism to improve Web service composition. The historical cooperation performance of the service providers can be used to predict their performance in future cooperation.

SLA profiling could be either centralised or distributed. Centralised SLA profiling architecture is easier to deploy, while distributed architecture would have better scalability, availability, fault tolerance, and is more compatible with the distributed Web services environment. The SLA profiling system can be deployed along with the service registry, usually UDDI, to simplify the information deployment and retrieval. The criteria to rank the service providers, the algorithms to estimate their future collaboration and the ways to deploy the SLA profiling system are domain specific problems.

The SLA initiator can consult the SLA profiling system when selecting service providers before the SLA negotiation phase. Figure 7 shows how the components work when the SLA initiator utilises SLA profiling at the beginning of SLA formation. After mapping the service requirements of the client application, the Agreement Initiator invokes the requestReputationInformation() function of the SLA Profiler for the reputation information about service providers. The SLA Profiler can provide a list of candidate service providers (when the SLA Profiler is deployed as long with the service registry) and their reputation information. If the SLA Profiler is implemented independent of the service registry or in a distributed manner, the SLA initiator can first ask for a list of candidate service providers from the service registry and then perform the selection based on the reputation information provided by the SLA Profiler. After that, the Agreement Initiator can send a “MSG_STARTNEGOTIATION” message to trigger the Agreement Negotiator to start interacting with the selected service provider. The Agreement Negotiator can also contact multiple service providers for comparison in order to choose the one that meets the service requirements the best. The next step goes to the agreement negotiation described earlier.

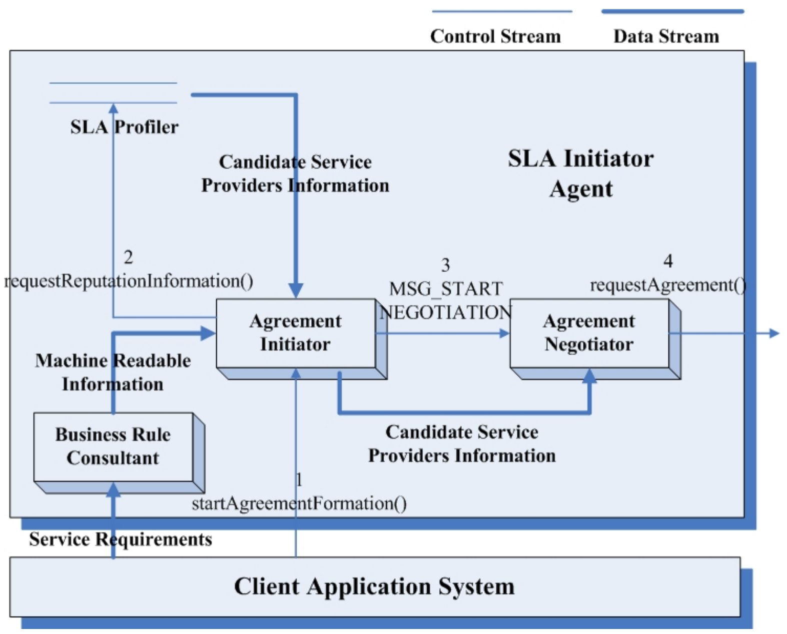

Control Stream
Data Stream
SLA Profiler
Candidate Service Providers Information
SLA Initiator Agent
2
requestReputationInformation()
3
MSG_START NEGOTIATION
4
requestAgreement()
Agreement Initiator
Agreement Negotiator
Machine Readable Information
Candidate Service Providers Information
Business Rule Consultant
1
startAgreementFormation()
Service Requirements
Client Application System

Fig. 7. Interactions in SLA initiator agent when utilising the SLA profiling

5.3.3 SLA Recovery

During service provision, many events, e.g., link failure, resource unavailability and service provider’s intended cancellation, may lead to incompliance of the contracted agreement. If not handled properly, these may result in severe consequences, such as service interruption, business loss and jeopardy of service provider’s reputation. The incompliance detection can be event-driven or schedule-driven. The agreement can contain an element like “EvaluationEvent” associated with a service level object, which defines the case in which the expression of the service level objective is to be evaluated. Alternatively, the detection task can be arranged according to an element “Schedule” referring to a schedule defined in the agreement. In the proposed framework, the above two detection mechanisms can be performed simultaneously by the Agreement Compliance Verifier. When the Web service provider intends to amend or cancel the agreement initiatively at runtime, the Web service system will inform the SLA responder agent of the event which will trigger the Agreement Compliance Verifier to perform the agreement evaluation. When unexpected events occur, such as resource breakdown and link failure, the incompliance can be detected by the scheduled detection.

SLA recovery is performed differently, depending on whether the detected SLA incompliance is caused by a cancellation on purpose or by an unexpected event. If the Web service provider system is incapable of satisfying the service provision, it will send a “MSG_INCAPABILITYDETECTED” message to the Agreement Compliance Verifier to announce its incapability and the need to amend or cancel the agreement. Upon the receipt of a “MSG_STARTAGREEMENTRECOVERY” message from the Agreement Compliance Verifier, the Agreement Negotiator will be triggered to contact the SLA initiator to renegotiate the agreement by sending a “MSG_AMENDAGREEMENT” or “MSG_CANCELAGREEMENT” message. On the SLA initiator side, upon the receipt of a “MSG_AMENDAGREEMENT” message, the SLA initiator will decide whether to start the renegotiation over the amendment or not, followed by a claim for compensation and renegotiation or service provider reselection. If it is a “MSG_CANCELAGREEMENT” message from the SLA responder, the SLA initiator can immediately claim for compensation and start searching for new service providers.

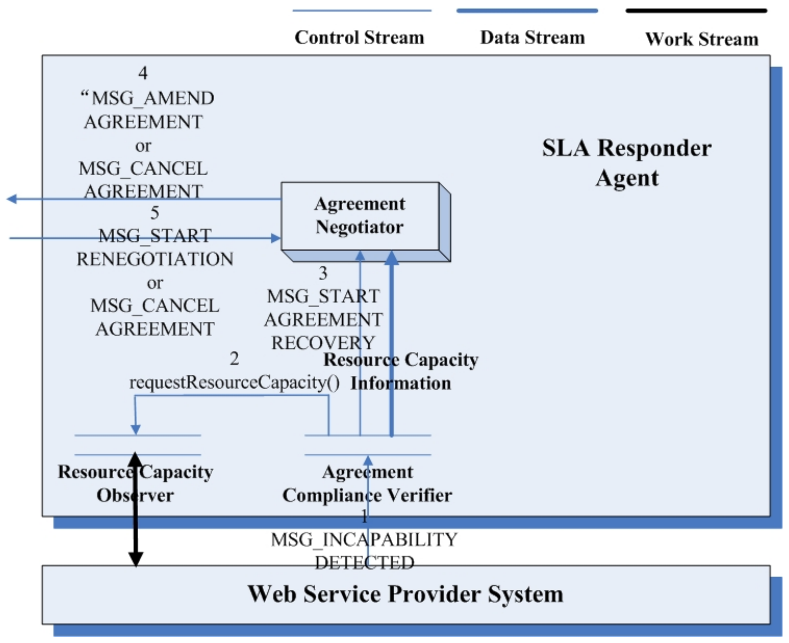

Control Stream
Data Stream
Work Stream
SLA Responder Agent
4 “MSG_AMEND AGREEMENT or MSG_CANCEL AGREEMENT
5 MSG_START RENEGOTIATION or MSG_CANCEL AGREEMENT
Agreement Negotiator
3 MSG_START AGREEMENT RECOVERY
Resource Capacity Information
2 requestResourceCapacity()
Resource Capacity Observer
Agreement Compliance Verifier
1 MSG_INCAPABILITY DETECTED
Web Service Provider System

Fig. 8. Interactions in SLA responder agent for SLA recovery

When the incompliance is caused by an unexpected event such as a link failure, the contracted parties will be informed by whoever detected the event, typically a supporting third party monitor party. In this case, the SLA responder contacts the SLA initiator to handle the event. The rest of SLA recovery is the same as in the case of amendment or cancellation on purpose. Figure 8 shows the interactions among the components in the SLA responder agent.

6. Simulation and Demonstration

In order to evaluate the feasibility of the proposed framework, some initial simulations have been conducted, including the goods delivery example described in Section 2. The goal of the simulations is to demonstrate how actual applications can be developed based on the framework proposed in this paper using a service-oriented approach. In the simulation, a system developed in the project named Adaptive Service Agreement and Process Management (ASAPM) (http://www.it.swin.edu.au/centres/ciamas/tiki-index.php?page=ASAPM) is exploited. The ASAPM project aims at providing adaptive service process management, including service composition planning, enactment, QoS monitoring, exception handling, and mediated composition re-planning by enabling adaptive service agreement management. The ASAPM system is implemented using the JADE development environment [5] and uses WS2JADE [21] as a middleware for the JADE agents to access Web services. Thus, the agent deployed in ASAPM is FIPA-compliant and can use agent communication language (ACL) [12] to communicate with one another. Figure 9 presents a scenario in ASAPM in which the agent of service consumer negotiates SLA for service A with provider A-1 and provider A-2. The agent of service consumer can either negotiate directly with the agent representing the Web service provider using ACL messaging (e.g., negotiation with A-1) or negotiate with the negotiation Web service of the Web service provider via WS2JADE (e.g., negotiation with A-2). In the latter case, the mapping between ACL messages and SOAP messages is handled by the proxy agent created by WS2JADE. The details of the agent interfaces and the negotiation Web service design can be found in [26].

As described in Section 2, in the goods delivery example, two of the major aspects the customer concerns about are the time consumption and cost of the global delivery. Due to the fact that the delivery involves multiple logistics companies, the individual time consumption and cost of each leg should be negotiated over and determined in advance. This can be done with SLA negotiation between the agent of the customer and the agents of the logistics companies. Besides, the transfer between the logistics companies must be appropriately arranged, because any delay caused by improper transfer may impact the fulfilment of the global service requirements. The time delay of the transfers can be minimised by deliberately specifying the pickup time and the time consumption of every single local delivery service in the SLAs. There are two available ways for the customer to arrange the negotiation and the actual delivery. The customer can negotiate with the first logistics company and then finish the first local delivery from Sydney to Canberra and then negotiate with the second logistics company which is capable of completing the second leg of the global delivery from Canberra to Melbourne, etc. This is a "negotiate and enact step by step" approach. Alternatively, the customer can negotiate with the logistics companies over the pickup time and the time consumption for each local delivery service before actual service provision. The local delivery services cannot and will not be provided until all the negotiations succeed. This is a "negotiate all and enact" approach. Comparing the two possible approaches, the following conclusions can be drawn. First, the "negotiate all and enact" mode is more efficient. If the time consumption of each leg can be determined in advance, proper transfer timings could be picked such that the delay caused by the transfer can be reduced or even eliminated. 2) The "negotiate all and enact" mode is more cost effective. In many business scenarios, advance booking costs much less than urgent booking. Furthermore, another hybrid approach can be adopted. Delivery services for significant legs can be negotiated and determined in advance while the others are left to on-the-fly negotiation at runtime. In this simulation, the "negotiate all and enact" approach is adopted.

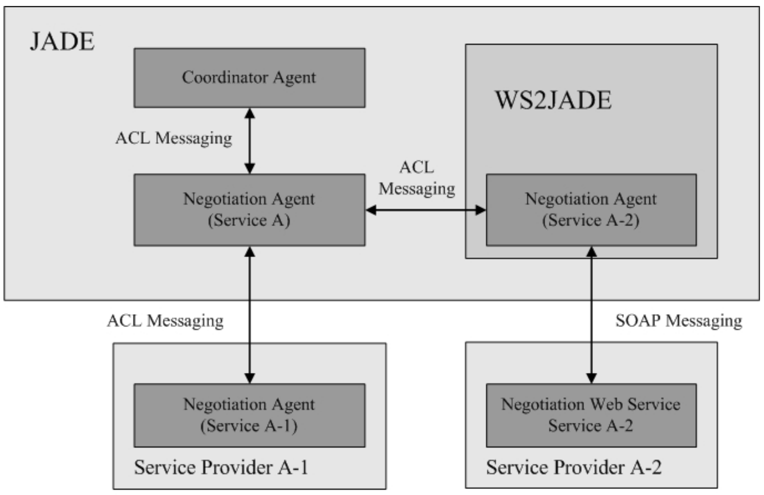
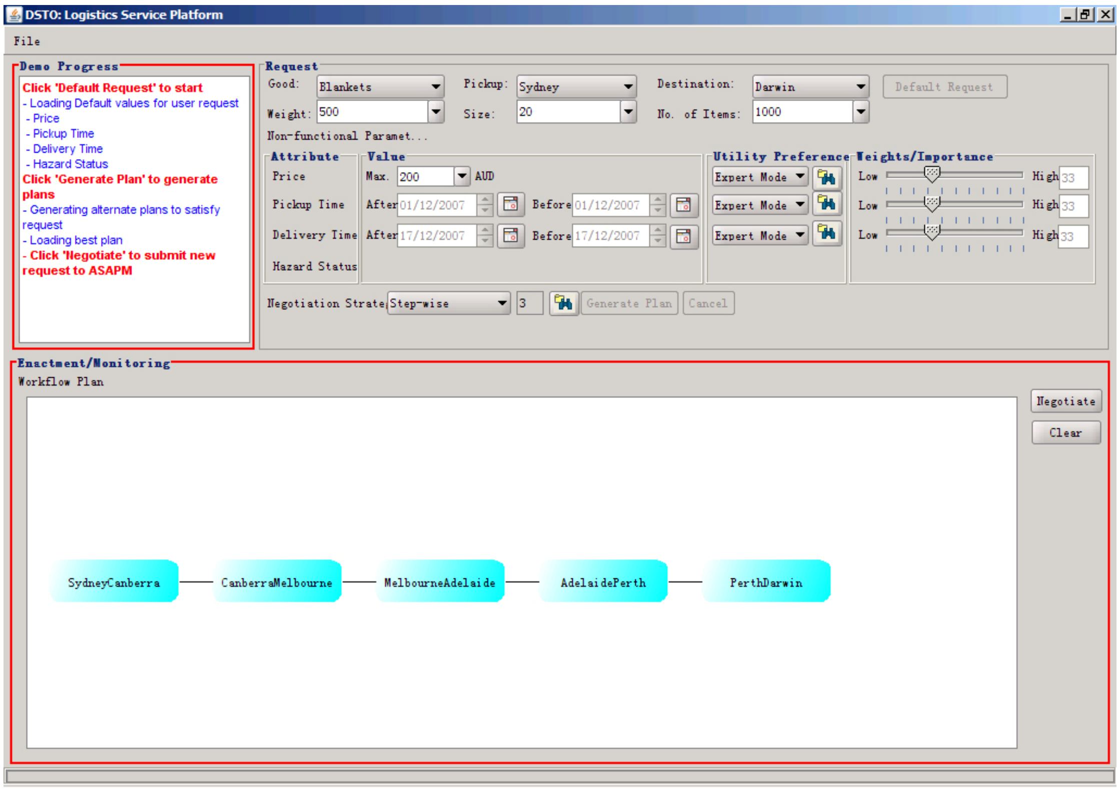

JADE
Coordinator Agent
ACL Messaging
WS2JADE
ACL Messaging
Negotiation Agent (Service A)
Negotiation Agent (Service A-2)
ACL Messaging
SOAP Messaging
Negotiation Agent (Service A-1)
Service Provider A-1
Negotiation Web Service Service A-2
Service Provider A-2

Fig. 9: Negotiation in ASAPM system

The framework proposed in this paper can fully meet the requirements of the goods delivery application. With agents deployed for the customer and the logistics companies, the customer just needs to enter its service requirements into the client application which will then generate a plan (an instance of the business process). In this example, 1000 items of blankets are expected to be delivered from Sydney to Darwin, within 16 days at a maximum price of 200 dollars. The customer agent negotiates with the agents of the five involved logistics companies over the pickup time, the delivery time and the prices of respective local delivery services. By deliberately arranging the pickup times and delivery times of the local delivery services, the delay caused by the transfer between logistics companies can be neglected. The efficiency and the result of the negotiation are determined by the decision making strategies of the agents. In this example, the heuristic decision making strategy is adopted, which means the agents will gradually increase (or decrease) their offers or counter-offers by making concession step by step to get close to a mutually-accepted agreement. The degree of the concession made in each step is also determined by the decision making strategy of the agent. When all the negotiations succeed, the composite delivery service can be provided by executing the individual local delivery services one by one according to respective SLAs. The user interface of the client application is presented in Figure 10.

During the delivery, exceptions may occur due to failures of any local delivery services, and then incur SLA violations. For example, a contracted logistics company cannot manage to arrange the conveyances at the agreed pickup time. Besides charging the compensation from the failed logistics company, the agent of the customer also needs to bring this local delivery service back to normal by either amending the SLAs contracted with the failed logistics company or selecting another logistics company to replace the failed one. The customer does not need to know about the exception and the recovery process as long as the SLA for the global delivery service can still be fulfilled after the recovery. In the ASAPM system, if any of the logistics companies states the incapability of delivering the goods according to the contracted SLA, the system can renegotiate the SLA with the failed logistics company, or just stop using the failed logistics company, and select and negotiate with a new logistics company with attempt to keep the customer's requirements fulfilled. If the SLA cannot be recovered, the customer must be notified of the situation.


DSTO: Logistics Service Platform

Demo Progress

Click 'Default Request' to start
- Loading Default values for user request
 - Price
 - Pickup Time
 - Delivery Time
 - Hazard Status
Click 'Generate Plan' to generate plans
- Generating alternate plans to satisfy request
- Loading best plan
- Click 'Negotiate' to submit new request to ASAPM

Request

Good: Blankets
Pickup: Sydney
Destination: Darwin
Weight: 500
Size: 20
No. of Items: 1000

Non-functional Paramet...

Attribute	Value		Utility Preference	Weights/Importance
Price	Max. 200 AUD		Expert Mode	Low High 33
Pickup Time	After 01/12/2007	Before 01/12/2007	Expert Mode	Low High 33
Delivery Time	After 17/12/2007	Before 17/12/2007	Expert Mode	Low High 33
Hazard Status				

Negotiation Strate: Step-wise 3

Enactment/Monitoring

Workflow Plan

Fig. 10. User interface of ASAPM system

This simulation shows that the four issues identified in Section 2 can be well addressed with the integration of agent and Web service technology in the proposed framework. With common acceptance of the WS-Agreement as the standardised approach to describe the quality of the Web services, the service providers can publish the advertisements of their services while the service consumers can select appropriate service providers based on these advertisements. Automatic SLA management can be achieved by autonomous SLA operations. These operations are facilitated by agents negotiating, interacting and cooperating autonomously. The SLA initiator can map the service consumers' service requirements to standardised service descriptions and negotiate the SLAs with the SLA responders. In service composition scenarios, SLA initiator can negotiate SLAs with multiple SLA responders which provide the component services to guarantee service consumers' global requirements of the composite services. When exceptions occur during the service provision, in order to recover the SLA, the SLA initiator can choose to renegotiate with the service provider over the amendments of the SLA or select and then negotiate with a new service provider to replace the failed one. Although the framework has no specific mechanisms for SLA monitoring and profiling, with the support of third parties, credible SLA monitoring and profiling can be implemented. We believe that the framework will be valuable for lifetime SLA management for services provision.

7. Conclusions and Future Work

As the Web services technology gained intensive attention in developing business applications, management of service level agreement (SLA) associated with Web services has become an urgent and thriving research topic. Adequate SLA management support should allow for autonomous, dynamic, and flexible SLA operations. This research advocates that utilising the agent technology in solving complicated SLA management issues is a fruitful line to follow. An innovative agent-based framework and its corresponding mechanisms for SLA formation, enforcement and recovery, in the context of service composition provision, are presented in this paper. The main functionalities the framework needs to provide and the interactions among the components are identified and discussed. The simulation has demonstrated that the proposed approach is promising.

In the future, integration of the proposed approach and the existing standards such as WSDL, UDDI and WS-Agreement will be further investigated. In addition, more sophisticated behavioural models will be studied, such as SLA suspension and resumption. Rapid SLA recovery, which allows a service consumer to preserve multiple backup service providers, will be explored.

Acknowledgement

This work is partly funded by the Australian Research Council Discovery Project Scheme under grant DP0663841. We are grateful for the contribution from SukKeong Goh and Mohan Baruwal Chhetri (Swinburne University of Technology) for their help with the simulation.

References

[1] A. Anderson, An introduction to the Web Services Policy Language, In: Proc. 5th IEEE International Workshop on Policies for Distributed Systems and Networks, POLICY’04, IEEE Computer Society, Yorktown Heights, New York, USA, 2004, pp. 189-192.

[2] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, M. Xu, Web Services Agreement specification (WS-Agreement), http://www.ogf.or g/documents/GFD.107.pdf, Grid Resource Allocation Agreement Protocol (GRAAP) WG, 2007.

[3] A. Alamri, M. Eid, A. EI Saddik, Classification of the State-of-the-art Dynamic Web Services Composition Techniques, International Journal of Web and Grid Services, 2 (2006) 148-166.

[4] A. C. Barbosa, J. P. Sauvé, W. Cirne, M. Carelli, Evaluating Architectures for Independently Auditing Service Level Agreements, Future Generation Computer Systems, 22 (2006) 721-731.

[5] F. Bellifemine, G. Caire, A. Poggi, G. Rimassa, JADE A White Paper, http://jade.tilab.com/pa pers/2003/WhitePaperJADEEXP.pdf, Telecom Italia Labs, 2003

[6] J. Brzostowski, M. B. Chhetri, R. Kowalczyk, Three decision-making mechanisms to facilitate Negotiation of Service Level Agreements for Web Service Compositions, In: Proc. Joint Conference of the INFORMS Section on Group Decision and Negotiation, the EURO Working Group on Decision and Negotiation Support, and the EURO Working Group on Decision Support Systems, GDN ‘07, Montreal, Canada, 2007, pp. 37-44.

[7] P. A. Buhler, J. M. Vidal, H. Verhagen, Adaptive Workflow = Web Services + Agents, In: L. J. Zhang (Ed), Proc. International Conference on Web Services, ICWS’03, CSREA Press, Las Vegas, Nevada, USA, 2003, pp. 131-137.

[8] M., B. Chhetri, S. Goh, J. Lin, J. Brzotowski, R., Kowalczyk, Agent-based Negotiation of Service Level Agreements for Web Service Compositions, In: Proc. Joint Conference of the INFORMS Section on Group Decision and Negotiation, the EURO Working Group on Decision and Negotiation Support, and the EURO Working Group on Decision Support Systems, GDN’07, Montreal, Canada, 2007, pp. 81-93.

[9] M. B. Chhetri, J. Lin, S. K. Goh, J. Yan, J. Y. Zhang, R. Kowalczyk, A coordinated architecture for the agent-based Service Level Agreement negotiation of Web service composition, In: Proc. 17th Australian Software Engineering Conference, ASWEC’06, IEEE Computer Society, Sydney, Australia, 2006, pp. 90-99.

[10] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, Web Services Description Language (WSDL) 1.1, http://www.w3.org/TR/2001/NOTE-wsdl-20010315, World Wide Web Consortium, 2001.
[11] K. Czajkowski, I. Foster, C. Kesselman, V. Sander, S. Trecke, SNAP: A Protocol for Negotiating Service Level Agreements and Coordinated Resource Management in Distributed Systems, In: D. G. Feitelson, L. Rudolph, U. Schwiegelshohn (Eds.), Proc. 8th Workshop on Job Scheduling Strategies for Parallel Processing, JSSPP'02, Edinburgh, Scotland, 2002, pp. 153-183.
[12] FIPA, Agent Communication Language Specification, http://www.fipa.org/specs/aclspecs.tar.gz
[13] D. Greenwood, G. Vitaglione, L. Keller, M. Calisti, Service Level Agreement management with adaptive coordination, In: Proc. International Conference on Networking and Services, ICNS'06, IEEE Computer Society, Silicon Valley, California, USA, 2006, pp. 45.
[14] M. N. Huhns, Agents as Web Services, Internet Computing, 6 (2002) 93-95.
[15] S. Y. Hwang, H. Wang, J. Tang, J. Srivastava, A Probabilistic Approach to Modeling and Estimating the QoS of Web-service-based Workflows, Information Science, 177 (23) (2007) 5484-5503.
[16] L. J. Jin, V. Machiraju, A. Sahai, Analysis on Service Level Agreement of Web services, Research Report HPL-2002-180, http://www.hpl.hp.com/techreports/2002/HPL-2002-180.pdf Hewlett-Packard Laboratories, 2002.
[17] A. Keller, H. Ludwig, The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services, Journal of Network and Systems Management, 11 (2003) 57-81.
[18] H. Ludwig, A. Dan, R. Kearney, Cremona: An architecture and library for creation and monitoring of WS-Agreements, In: M. Aiello, M. Aoyama, F. Curbera, M. P. Papazoglou (Eds.): Proc. 2nd International Conference Service-Oriented Computing, ICSOC'04, ACM, New York, NY, USA, 2004, pp. 65-74.
[19] H. Ludwig, A. Keller, A. Dan, R.P. King, R. Franck, Web Service Level Agreement (WSLA) language specification, Version 1.0, http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf International Business Machines Corporation (IBM), 2003.
[20] N, Milanovic, M. Malek, Current Solutions for Web Service Composition, IEEE Internet Computing, 8 (2004) 51-59.
[21] X. T. Nguyen, R. Kowalczyk, M. B. Chhetri, A. Grant, WS2JADE: A Tool for Run-time Deployment and Control of Web Services as JADE Agent Services, Software Agent-Based Applications, Platforms and Development Kits, (2005) 223-251.
[22] D. Ouelhadj, J. Garibaldi, J. MacLaren, A multi-agent infrastructure and a service level agreement negotiation protocol for robust scheduling in grid computing, In: P. M. A. Sloot, A. G. Hoekstra, T. Priol, A. Reinefeld, M. Bubak (Eds.), Advances in Grid Computing, European Grid Conference, EGC'05, Amsterdam, The Netherlands, 2005, pp. 651-660.
[23] A. Sahai, A. Durante, V. Machiraju, Towards automated SLA management for Web services, Research Report HPL-2001-310, www.hpl.hp.co.uk/techreports/2001/HPL-2001-310R1.pdf Hewlett-Packard Laboratories, 2002.
[24] A.S. Vedamuthu, D. Orchard, F. Hirsch, M. Hondo, P. Yendluri, T. Boube, Ü. Yalçınalp, Web Services Policy framework (WS-Policy), Version 1.5, http://www.w3.org/TR/ws-policy/, World Wide Web Consortium, 2007.
[25] Z. Xu, P. Martin, W. Powley, F. Zulkernine, Reputation-Enhanced QoS-based Web Services Discovery, In: Proc. IEEE International Conference on Web Services, ICWS'07, IEEE Computer Society, Salt Lake City, Utah, USA, 2007, pp. 249-25
[26] J. Yan, R. Kowalczyk, J. Lin, M. B. Chhetri, S. Goh, J. Y. Zhang, Autonomous service level agreement negotiation for service composition provision, Future Generation Computer Systems 23 (2007) 748-759.

Appendix. XML Schema for Description of Goods Delivery Service

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<xs:schema targetNamespace="http://www.ict.swinburne.edu.au/namespaces/gps"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:gps="http://www.ict.swinburne.edu.au/namespaces/gps" elementFormDefault="qualified"
attributeFormDefault="qualified">
    <xs:complexType name="TGoodsDeliveryServiceDescription">
        <xs:complexContent>
            <xs:restriction base="xs:anyType">
                <xs:sequence>
                    <xs:element name="GoodsType" type="gps:TGoodsType"/>
                    <xs:element name="GoodsQuantity" type="gps:TGoodsQuantity"/>
                    <xs:element name="Price" type="gps:TPrice"/>
                    <xs:element name="PickupTime" type="xs:dateTime"/>
                    <xs:element name="TimeConsumtion" type="gps:TTimeConsumption"/>
                    <xs:element name="DeliverySource" type="gps:TDeliverySource"/>
                    <xs:element name="DeliveryDestination" type="gps:TDeliveryDestination"/>
                </xs:sequence>
            </xs:restriction>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="TGoodsType">
        <xs:simpleContent>
            <xs:extension base="xs:string"/>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="TGoodsQuantity">
        <xs:simpleContent>
            <xs:extension base="xs:float">
                <xs:attribute name="Unit" type="gps:TQuantityUnit"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:simpleType name="TQuantityUnit">
        <xs:restriction base="xs:string">
            <xs:enumeration value="ton"/>
            <xs:enumeration value="kg"/>
            <xs:enumeration value="g"/>
            <xs:enumeration value="mg"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="TPrice">
        <xs:simpleContent>
            <xs:extension base="xs:float">
                <xs:attribute name="CurrencyType" type="gps:TCurrencyType"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:simpleType name="TCurrencyType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="USD"/>
            <xs:enumeration value="RMB"/>
            <xs:enumeration value="RMB"/>
            <xs:enumeration value="EUR"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="TTimeConsumption">
        <xs:simpleContent>
            <xs:extension base="xs:integer">
```

```
                <xs:attribute name="Unit" type="gps:TTimeConsumptionUnit"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:simpleType name="TTimeConsumptionUnit">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Year"/>
            <xs:enumeration value="Month"/>
            <xs:enumeration value="Week"/>
            <xs:enumeration value="Day"/>
            <xs:enumeration value="Hour"/>
            <xs:enumeration value="Minute"/>
            <xs:enumeration value="Second"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="TDeliverySource">
        <xs:simpleContent>
            <xs:extension base="xs:string"/>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="TDeliveryDestination">
        <xs:simpleContent>
            <xs:extension base="xs:string"/>
        </xs:simpleContent>
    </xs:complexType>
</xs:schema>
```