A Contract Language for Service-Oriented Dynamic Collaborations

Surya Nepal, John Zic, and Shiping Chen

CSIRO ICT Center, P.O. Box 76,
NSW 1710, Australia
FirstName.LastName@csiro.au

Abstract. Dynamic collaborations are built using contributed resources that have come across the organizational boundaries. These resources include data, application, software, tools as well as infrastructures, and are typically subject to a rich set of access policies. The automated instantiation of a collaboration using such resources including their interoperability is a difficult problem. Existing systems are either built for specific resources, or use manual and ad-hoc approaches. This problem has attracted the Web Services community, where Web Services standards such as WSLA and WS-CDL have been proposed to address similar problems. These approaches are designed to deal with scenarios involving two parties: a service provider and a service consumer. They do not scale well to multiparty nature of dynamic collaborations. This paper proposes a contract language for dynamic multiparty collaborations that captures the contributed resources and negotiated agreements on them, as well as the mechanisms for instantiation and termination of the collaboration. The language itself has been defined using XML Schema and has been implemented in a dynamic collaboration platform to provide a connectivity service.

1 Introduction

Recently, there has been much interest in forming on-demand dynamic collaborations between autonomous, competitive organizations to collaborate on occasion [4]. Such collaborations are built for a specific purpose. For example, a dynamic collaboration among researchers in eResearch domain may be required so as to study a specific climate change problem. The idea of collaborating in this way, however, is not new. For example, the area of virtual organizations [1][2][3] explores mechanisms that enable entities from different organizations to collectively create such virtual enterprises or organizations. This goal is typically achieved through open service discovery, negotiation and execution based on service level agreements (SLAs). A trend is developing, however, where collaborations are much more dynamic and the resources contributed by each organization to the collaboration are governed by a set of complex policies [5][4][18] constructed upon each separate organization's policies. In this situation, open service discovery mechanisms cannot be used, as resource information is deliberately hidden through the use of organizational policies. Collaborations built around this concept are termed *dynamic collaborations*.

E. Bertino and J.B.D. Joshi (Eds.): CollaborateCom 2008, LNICST 10, pp. 545–562, 2009.

One of the key features of a dynamic collaboration is an on-demand contribution of resources from participating autonomous organizations. Recently, resources such as storage and networking infrastructures, tools, software and data are implemented using Web Services technologies so that they can be made available as services. For example, the concept of Software-as-a-Service (SaaS) is introduced for providing software [19] and Infrastructure-as-a-Service (IaaS) [11] for storage and networking infrastructures. Therefore, it is possible to define and share *resources as services* in the context of dynamic collaborations.

The management of workflows of contributed resources during a dynamic collaboration is a critical issue. The management workflows include negotiation of resources, validation of resources, instantiating of resources/collaboration, monitoring resources and releasing resources when the collaboration terminates or when a partner leaves the collaboration. Existing dynamic collaborations deal with these issues either in an ad hoc manner or manually. There is a need of a framework that enables the definition of collaborations in such a way that it can be unambiguously and automatically instantiated and managed.

The problem of interoperability becomes evident while managing the workflows of resources due to involvement of a variety of autonomous organisations as well as the resources contributed by them. This problem has attracted the Web Services community, where standards such as WSLA and WS-CDL have been developed. Keller and Ludwig [17] defined a Web Service Level Agreement (WSLA) [10] framework for defining and monitoring Service Level Agreements (SLAs) between service providers and service consumers in an electronic commerce scenario. Kavantzas et al. [27] define the Web Services Choreography Description Language (WS-CDL) that describes peer-to-peer collaborations of Web Services participants by defining their common and complementary observable behavior.

These languages work well in a type of scenarios that involve only two parties with distinct roles; the service provider offering a service and the service consumer requesting and consuming the service. However, in the context of dynamic collaboration, this does not work well due to the following reasons:

- Dynamic collaborations have *complex policies* defining the interactions, access and use of resources.
- *Multiple parties* are involved in dynamic collaborations;
- *Multiple resources (services)* are contributed to the collaboration by multiple parties;
- *Both roles* of service provider and service consumer are often played by a single party;
- *All parties* must agree with each other's contributions and obliged with their agreements.

In order to start to address this problem, we recently [28] proposed an extension to WSLA, called WSLA+, for dynamic collaborations. However, the extended language involves semantic interpretations of WSLA elements different than what they were intended for as well as could not fulfill all the above requirements specifically on specifying policies on services. In order to fulfill these requirements, this paper presents a framework for Web Service Collaborative Context Definition Language for dynamic collaboration, called WS-CCDL. It enables collaborating partners to

unambiguously define the requirements for the collaboration as well as agreements for all the resources contributed to the collaboration. The semantics of the framework is defined briefly as a short paper in [29]. In this paper, we describe the language itself, which is defined using XML Schema and an implemented prototype system in an environment to deliver a connectivity service between multiple partners in a collaboration.

The rest of the paper is structured as follows. Section 2 describes the background and motivation of this work using examples from a variety of applications. We then describe the framework and its runtime model in Sections 3 and 4, respectively. Section 5 describes the connectivity service implemented using the framework. The final section draws the concluding remarks and future works.

2 Background and Motivation

Dynamic collaborations bring together complementary sets of competencies from competing enterprises to address new market opportunities in a rapidly evolving services economy [4]. In addition to enterprise applications, dynamic collaborations between cross-organizational entities occur in other application domains such as Global Command and Control Systems in military application [5], coalitions formed among civilian organizations as responses to emergency situations, coalitions between researchers in different institutions for eResearch applications and health care facility and practitioners' coalitions for eHealth applications. The valuable contributions of dynamic collaboration technologies have been recognized by both industries and governments as evident from the special issue Journal on Dynamic Collaboration by NEC [4], the Australian Government's funding support for collaborative platforms within eResearch [18], European collaborative project ECOSPACE [23] and DARPA funded research in dynamic collaboration [14][15][16]. We next describe three motivating examples of collaboration from different application domains.

Post-production industry - Figure 1 shows an example of a dynamic collaboration formed between three distributed, autonomous postproduction houses in order to produce a movie [6]. As can be seen in the figure, three companies contribute

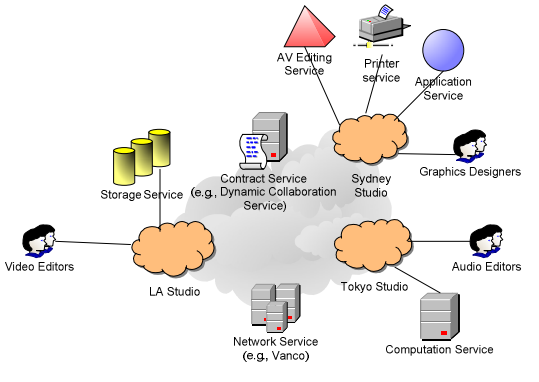


Fig. 1. An Example of Dynamic Collaboration in a movie postproduction industry

different resources as services in the collaboration. The contributed services include Audio Visual editing service, Audio Visual application service, storage service, printer service, computation service as well as the services of audio video specialists. The collaboration also uses external third party provided services such as contract service and network service.

Retail enterprise - [20] explains the scenarios in retail and hospitality industries where collaborations with partners provide effective ways of gaining competitive advantages by making use of existing resources. They include streamlining the product recall process, online web conferencing and virtual war room. The partners within the retail industries include retail outlets, suppliers, transport companies, etc.

eHealth – Collaborative eHealth programs such as Baltic eHealth [21] and Kentucky eHealth [22] are taking advantage of collaboration in health care sectors. The aim of these programs is to provide secure, private and confidential healthcare services by taking advantages of expertise within the regions.

3 WS-CCDL Framework

Figure 2 shows the main concepts and their relationships within WS-CCDL. As a convention throughout this paper, we denote sets of entities using upper-case letters, with the entities themselves in lower-case letters. The main entity, called *contract*, captures the entities requirements, participants, contributions and agreements. The resources, activities, policies and attributes are used to describe these entities.

A *contract* is a collaborative context that specifies not only the requirements for the collaboration but also captures the contributions made by the participants as well as the agreements between them for contributed resources. The contract includes collaboration requirements, contributions by participants, agreements among participant and the list

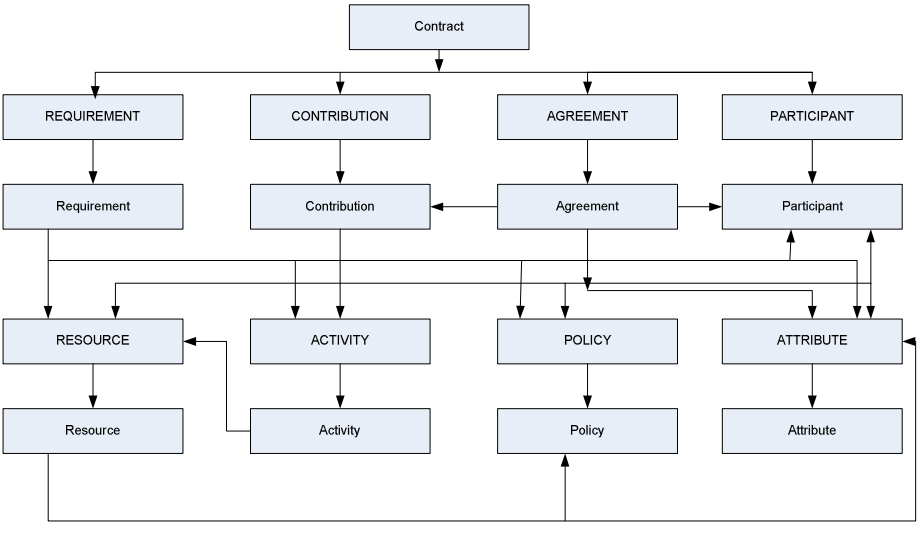


Fig. 2. Overview of main WS-CCDL concepts and relationships

of participants. The high-level elements of the contract are shown in a snapshot of XML-schema below.

```
<xs:element name="econtract" type="ccdl:econtractType"/>
 <!-- high level element -->
<xs:complexType name="econtractType">
 <xs:sequence>
      <xs:element ref="ccdl:participants"/>
      <xs:element ref="ccdl:requirements"/>
      <xs:element ref="ccdl:contributions"/>
      <xs:element ref="ccdl:agreements"/>
 </xs:sequence>
 <xs:attribute name="econtractId" type="xs:anyURI" use="required"/>
 <xs:attribute name="Version" type="ccdl:VersionType" default="1.0"/>
 <xs:attribute name="contractAgreementProtocol" type="xs:anyURI" use="required"/>
</xs:complexType>
```

In addition to the four core elements, the contract must also specify the negotiation and agreement protocol.

A dynamic collaboration is built among a number of participating organizations. In general, we categorize the participants as follows:

Initiator is a participant who initiates the collaboration. The initiator specifies the initial requirements for the collaboration.

Invited participants are the participants that are invited by the initiator to participate in the collaboration.

These participants are further categorized as follows.

Signatory participants are authorized by the collaborating organisations to negotiate on their behalf. Both initiator and invited participants are signatory participants.

Contributing participants are contributed by the signatory participant to participate in the collaboration. For example, a participant can delegate a storage service provider to provide a storage service to the collaboration. We capture such participants as resources in our framework. We do further discussion on resources later in this section.

The signatory participants are defined using their identity, role, addressing mechanism, etc. as follows. Also, the participants must be specified with an authentication algorithm.

```
<xs:element name="participants" type="ccdl:participantsType"/>
 <xs:complexType name="participantsType">
      <xs:sequence>
           <xs:element ref="ccdl:participant" maxOccurs="unbounded"/>
      </xs:sequence>
 </xs:complexType>
<xs:complexType name="participantType">
      <xs:sequence>
           <xs:element ref="ccdl:identity"/>
           <xs:element ref="ccdl:role"/>
           <xs:element ref="ccdl:organisation"/>
           <xs:element ref="ccdl:addressing"/>
           <xs:element ref="ccdl:section"/>
```

```
        </xs:sequence>
        <xs:attribute name="authenticationAlg" type="xs:anyURI" use="required"/>
        <xs:attribute name="name" type="xs:string" use="required"/>
        <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:complexType>
```

As discussed earlier, an initiator – the participant who initiates the collaboration - expresses the need of the collaboration through requirement. The requirements specify not only the purpose of the collaboration, but also durations, activities and resources needed for establishing and continuing the collaboration. In order to capture these initial statements about the collaboration, we define the requirement as follows.

```
<xs:element name="requirement" type="ccdl:requirementType"/>
 <xs:complexType name="requirementType">
        <xs:sequence>
            <xs:element ref="ccdl:activities" minOccurs="0"/>
            <xs:element ref="ccdl:resources" minOccurs="0"/>
            <xs:element ref="ccdl:policies" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string" use="required"/>
        <xs:attribute name="date" type="xs:date" use="required"/>
        <xs:attribute name="time" type="xs:time" use="required"/>
        <xs:anyAttribute namespace="##any" processContents="lax"/>
 </xs:complexType>
```

It is also important to note that these requirements are negotiable. That is, a participant may disagree with some of the requirements and may negotiate the changed requirements with other participants using negotiation protocols. Examples of such negotiation protocols will be discussed in the next section.

Within a collaboration, participants may engage in a number of activities, and each activity may need different set of participants and resources. For example, a collaboration in the post-production industry may have activities like video enhancing, audio enhancing, audio-visual mixing, preparing scenes, etc. In our framework, we represent such activities as follows.

```
<xs:element name="activities" type="ccdl:activitiesType"/>
 <xs:complexType name="activitiesType">
        <xs:sequence>
            <xs:element ref="ccdl:activity" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
 </xs:complexType>
<xs:complexType name="activityType">
 <xs:sequence>
        <xs:element name="activityid" type="ccdl:idType"/>
        <xs:element name="activityname" type="xs:string"/>
        <xs:element name="resources" type="ccdl:resourceType"/>
        <xs:element ref="ccdl:policies" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="participant" type="xs:string" minOccurs="0"
                    maxOccurs="unbounded">
            <xs:keyref name="activityref" refer="ccdl:participantKey">
                <xs:selector xpath="ccdl:econtract/participants/identity"/>
                <xs:field xpath="id"/>
```

```
            </xs:keyref>
        </xs:element>
    </xs:sequence>
        <xs:attribute name="date" type="xs:date" use="required"/>
        <xs:attribute name="time" type="xs:time" use="required"/>
        <xs:attribute name="duration" type="xs:time" use="required"/>
        <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:complexType>
```

It is important to note here that the activity definition includes both the description for the activity as well as the resources required to perform these activities. Whether the overall resource requirements may capture these resources depends on what kind of resource satisfaction algorithm would be used and this will be discussed in the next section in details.

A unique characteristic of a dynamic collaboration is that it is formed by combining each participant's contributed resources. In our framework, signatory participants contribute resources towards the collaboration. The resources are contributed at both activity and collaboration levels. That is, one participant may choose to contribute resources in such a way that it can be used by all activities within a collaboration, whereas other participants may choose to contribute resources only for a specific activity. The resources include contributing participants, software, data, tools and information resources. The resources are formally represented as follows.

```
<xs:element name="resources" type="ccdl:resourcesType"/>
 <xs:complexType name="resourcesType1">
    <xs:sequence>
        <xs:element ref="ccdl:resource" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
 </xs:complexType>
 <xs:element name="resource" type="ccdl:resourceType">
    <xs:key name="resourceKey">
        <xs:selector
                xpath="ccdl:econtract/collaborationcontext/resources/resid/identity"/>
        <xs:field xpath="@id"/>
    </xs:key>
 </xs:element>
 <xs:complexType name="resourceType">
    <xs:sequence>
        <xs:element name="resid" type="ccdl:idType"/>
        <xs:element name="resname" type="xs:string"/>
        <xs:element ref="ccdl:policies" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##any" processContents="lax"/>
 </xs:complexType>
```

A collaborator must provide enough information about any contributed resources so that they can be accessed by other collaborating participants such as addresses, interfaces and protocols. With the emergence of Web Services, it is now possible to define these resources as services and contribute them to the collaboration. The details on how to model and define *a resource as a service* is outside the scope of this paper.

However, we refer readers to the implementation of storage and networking *infrastructures as services* in [11] for further details.

Fundamentally, we maintain that collaborations are driven by policies, and this is presented in the above formalization. Different organizations participating in the collaboration each operate under their own set of policies. Furthermore, the collaboration is built and operated under a set of policies defined for different entities and activities. For example, a collaborator can define policies that determine who can participant in the video editing activities. Similarly, a participant contributing resources can specify a set of policies for their contributed resources. For example, only project leaders can access the edited videos. In order to represent policies within our framework, we define policy as follows.

```
<xs:element name="policies" type="ccdl:policiesType"/>
 <xs:complexType name="policiesType">
     <xs:sequence>
         <xs:element ref="ccdl:policy" minOccurs="0" maxOccurs="unbounded"/>
     </xs:sequence>
 </xs:complexType>
 <xs:element name="policy" type="ccdl:policyType"/>
 <xs:complexType name="policyType">
     <xs:sequence>
         <xs:element name="PolicySet" type="xacml:PolicySetType"/>
     </xs:sequence>
 </xs:complexType>
```

Different Digital Right Management and access policy expression languages such as XACML [24][25] can be used to describe policies.

One of the key elements specified in the requirements is resources. The requirement of resources is fulfilled through contributions from the participating parties. The contribution can be done at a specific activity level or at the collaboration level. Also, the contributions need to satisfy the corresponding policies specified in the requirement. Therefore, the contribution of the resources by participants depends on the resource needs specified in the requirements as well as the corresponding policies. We define such contribution as follows.

```
<xs:element name="contributions" type="ccdl:contributionsType"/>
 <xs:complexType name="contributionsType">
     <xs:sequence>
         <xs:element ref="ccdl:contribution" minOccurs="0" maxOccurs="unbounded"/>
     </xs:sequence>
 </xs:complexType>
 <xs:element name="contribution" type="ccdl:contributionType" id="string"/>
 <xs:complexType name="contributionType">
     <xs:sequence>
         <xs:element ref="ccdl:contributor"/>
         <xs:element ref="ccdl:activitityName" minOccurs="0"
                     maxOccurs="unbounded"/>
         <xs:element ref="ccdl:resourceName" minOccurs="0"
                     maxOccurs="unbounded"/>
         <xs:element ref="ccdl:policies" minOccurs="0" maxOccurs="unbounded"/>
     </xs:sequence>
 </xs:complexType>
```

As mentioned earlier, in the Web Services and SOA environment, such resources can be contributed as services, i.e., Resource-as-a-Service (RaaS).

The contributed resources are negotiated among participants. One of the important requirements of the dynamic collaboration is that all participants must agree with each others' contributions. In the Web Services environment, this means agreement with the contributed services. We define agreement using a digital signature as shown below.

```
<xs:element name="agreements" type="ccdl:agreementType"/>
 <xs:complexType name="agreementTypeType">
      <xs:sequence>
          <xs:element ref="ds:Signature" maxOccurs="unbounded"/>
      </xs:sequence>
 </xs:complexType>
```

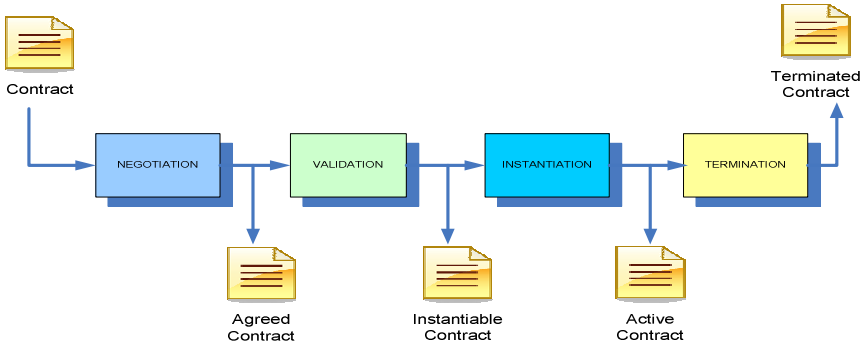


Fig. 3. Different phases of the contract in a runtime environment

4 WS-CCDL Runtime Framework

In the above section, we define a contract language along with its associated elements. The contract goes through the different phases in its lifecycle as shown in Figure 3. This section describes the four different phases of the contract namely negotiation phase, validation phase, instantiation phase and termination phase.

Negotiation Phase

During this phase, the participants negotiate the terms and conditions of the collaboration, including contributed resources and policies attached to them. Each party may contribute a set of resources along with associated access policies. A party in the collaboration may, however, disagree with the terms and conditions of the collaboration or some particular contributed resources or access policies. In such scenarios, participants may need to re-negotiate the content in the contract. This is facilitated using contract negotiation algorithms.

A set of Web Services standards for negotiations have been proposed for this purpose: WS-AgreementNegotiation [7], WS-Negotiation [8], and WS-Agreement [9]. WS-Negotiation is a domain independent language for generating agreements between a service provider and a service consumer. WS-Negotiation only considers two parties: a provider and a consumer. WS-Negotiation supports a simple one to one negotiation protocol. Though the objective of WS-Agreement is to have a language and a protocol that creates agreements, publicizes a service offer and provides a monitoring service, it also specifies a very simple negotiation protocol. As this protocol does not allow offer refinement, WS-AgreementNegotiation was defined so as to overcome this shortcoming by allowing negotiation and re-negotiation of agreements between two parties.

As mentioned earlier, these work well in two party scenarios but are not suited for use in multi-party, multi-service negotiation in the context of dynamic collaborations. We have defined protocols for dynamic collaboration in [12]. The unique characteristic of such protocols is that all parties must agree with each others' contributions. The negotiation phase ends when the contract meets this characteristic and results in an *agreed* contract. An agreed contract is such all parties have seen the final contributions from all other parties and all have agreed with it.

Validation Phase

During this phase, the agreed contract is validated against two criteria first of all, and if both are successful, a determination is then made whether the collaboration is instantiable or not.

The first criterion is to check whether the resources required for the collaboration as well as activities are met by the contributions made by the participants. Contributed resources can be categorized into the following two broad categories:

- Collaboration specific – includes resources contributed to the whole collaboration and not tied to a specific activity.
- Activity specific – includes resources contributed for specific activities.

That means resources specified for the collaboration can be used for a particular activity if the resources contributed for an activity do not satisfy the requirements specified for that activity. We also need to ensure that the resource satisfaction for a contract meet the policy conditions. For example, if a printer is available for a week and the collaboration requires the printer for one month, then the contributed resource can not meet the required resource. We define this check as follows. The contract is said to be *resource satisfied* if the contributed resources meet all the resource requirements of the contract under the given policy.

The second criterion is to check whether different set of policies specified within the contract conflict with each other or not. Policies are specified at the level of requirements, participants, resources, activities and contributions. Some of these policies may conflict with each other. For example, a policy may state that participants can access all information related to the collaboration at the requirement level, but the policy specified at the contribution by a participant may prevent certain participants accessing some information. The policies need to be checked and any conflicts arises need to be resolved. The final set of policy must satisfy the

requirements for the collaboration and activities within it. We define this process formally as follows. The contract is said to be *policy satisfied* if the policies expressed in the contract are not conflicting each other.

After the successful checking of the two criteria above, we need to check whether the contract is instantiable or not. We define the contract is *instantiable* if it is agreed by all participant and satisfies both policy and resource requirements. It is important to note that the above definition does not cover some of the technical aspects such as all resources specified must have a valid address and they are online. These aspects are dynamic and included in the monitoring of contract, which is outside the scope of this paper.

Instantiation Phase

Once the contract has been validated, it is then interpreted by an engine (the instantiation engine) that results in an instantiation of the collaboration. The instantiation engine extracts the information from the contract and sends to relevant services. For example, the network specific requirements are extracted and sent to the network service provider. Once the collaboration is established (and has not been terminated), the contract is said to be *active*. The contract is *active* if it has been successfully instantiated and has not been terminated.

Termination Phase

The last phase of the runtime environment is called termination, where the collaboration is terminated as per contract. We have previously defined one such termination protocol using WS-BusinessActivity [12]. Other distributed termination protocols can be used. There are a lot of activities that need to be performed at the time of termination such as logging, archiving and destroying. The agreed policies for termination determine what actions need to be taken for which piece of information. The contract is said to be terminated if all parties agreed to terminate the instantiated collaboration and the termination protocol is run successfully.

We discussed the four different phases that the contract goes through during a runtime environment. The discussion of these phases above also raises a number of research questions such as: determination of both resource and policy satisfaction, the definition of policies for a party to join an already existing collaboration, and how exceptions are handled, such as a partner failing to comply with an agreement. We are actively perusing these issues as part of the future works.

5 Connectivity Service

This section describes a prototype contract based connectivity service that we have implemented and deployed in order to demonstrate the usage of the proposed contract language in dynamic collaborations. In our prototype system, we have made an assumption that this contract service will be provided by a trusted third party, whose sole business is to provide secured, managed network connectivity to clients for the purpose of collaboration. One of the main features of this service is that the connectivity is abstracted to the user level so that any number of users can connect to

each other by using computer connected to the Internet and can use any collaborative applications. The connectivity between collaborators is established when they agree to the terms and conditions specified in the contract.

Figure 4 shows the overall architecture for the implemented prototype connectivity service driven by contract. Our architecture consists of two services: the contract service itself, and a VPN service. The contract service provides the management of the contract and includes a registration and discovery service for the use of its clients. The VPN service is used by the contract service provider to provide connectivity once the contract is agreed. All these services as well as users (from different administrative domain such as home office, university and government office) are connected to the Internet. Initially, all of them communicate with each other using an open Internet connection. When all users agreed with the contract and the contract is deployed into the Internet, a dynamic Intranet is created (the intranet is referred to as dynamic because the connectivity is established at the user level rather than the machines used by the users). We next describe the major implemented components of the architecture and the interactions among them.

Initially, each collaborator resides within its own administrative domain and may be unknown to the other collaborators. In order to initiate the collaboration, each collaborator needs a contract service. All collaborators first subscribe to the contract service so that they can either initiate or be invited to join into a dynamic collaboration. The collaborators use a service to register themselves and make them available for collaborations. Once the registration process is completed, the collaborator then downloads and installs a client application which then enables them to process the contract through each of the negotiation, validation, instantiation and termination phases described above.

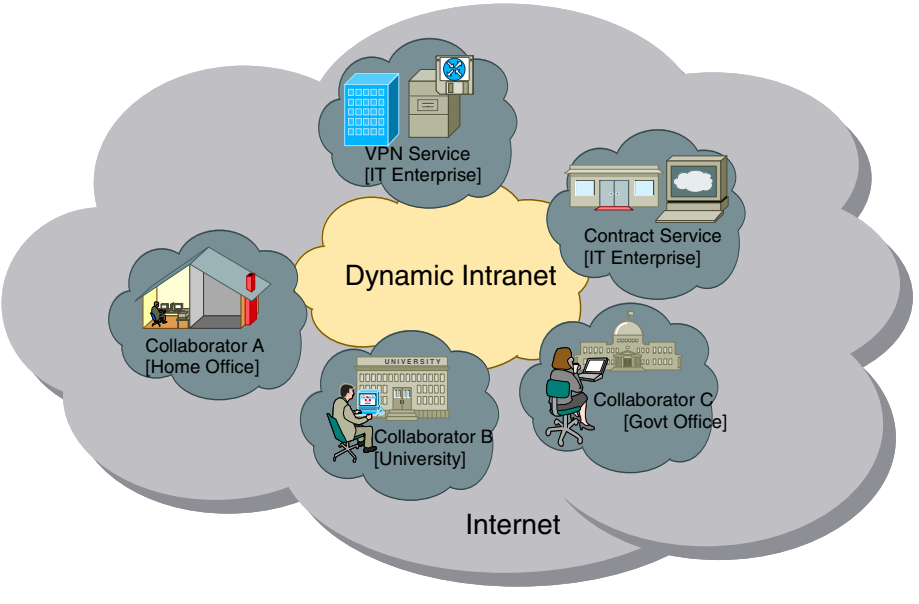
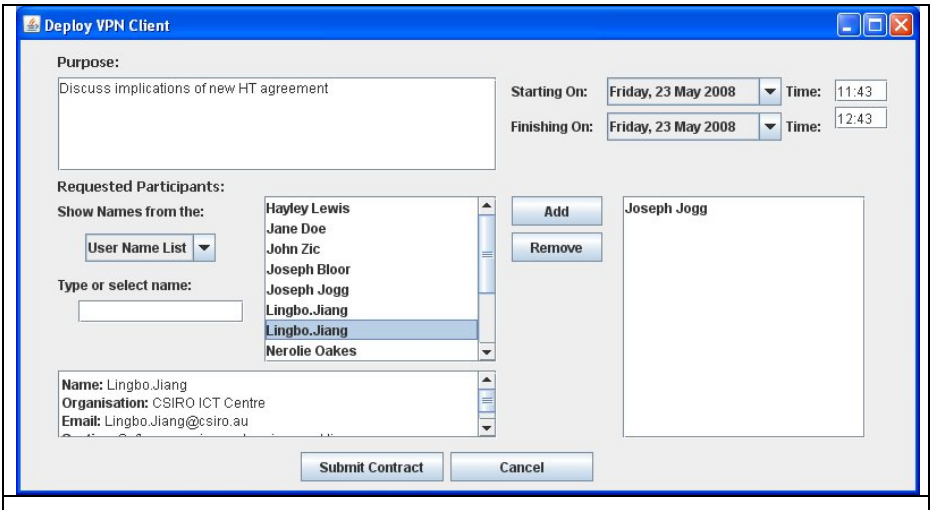
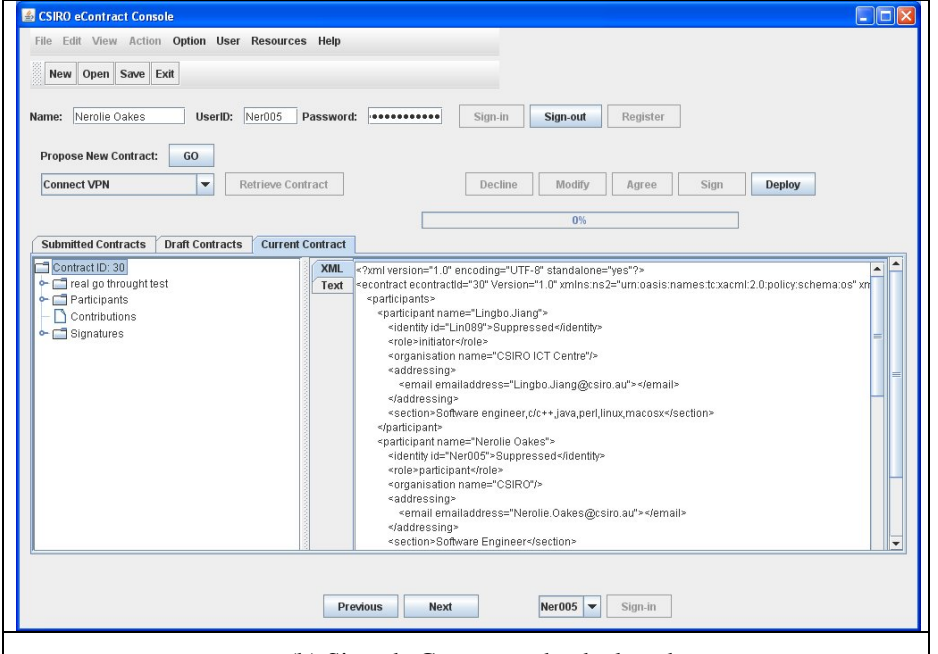


Fig. 4. Overall architecture of eContract service for dynamic collaborations



(a) inviting collaboration participants



(b) Signed eContract to be deployed

Fig. 5. The screenshots of the eContract prototype system

We next describe the process of creating a dynamic collaboration between three parties A, B and C. Suppose A wishes to set up a dynamic collaboration with B and C and use a document sharing application called Virtual Terminal (VT). First, all three

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<econtract econtractId="26" Version="1.0" xmlns:ns2="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
xmlns="urn:ccdl:v1" xmlns:ns4="http://www.w3.org/2000/09/xmldsig#"
xmlns:ns3="http://www.w3.org/2004/12/addressing">
   <participants>
      <participant name="Nerolie Oakes">
         <identity id="Ner005">Suppressed</identity>
         <role>initiator</role>
         <organisation name="CSIRO"/>
         <addressing>
            <email emailaddress="Nerolie.Oakes@csiro.au"></email>
         </addressing>
         <section>Software Engineer</section>
      </participant>
      <participant name="Lingbo.Jiang">
         <identity id="Lin089">Suppressed</identity>
         <role>participant</role>
         <organisation name="CSIRO ICT Centre"/>
         <addressing>
            <email emailaddress="Lingbo.Jiang@csiro.au"></email>
         </addressing>
         <section>Software engineer,c/c++,java,perl,linux,macosx</section>
      </participant>
   </participants>

<requirements>
      <collaborationContext time="2008-05-16T10:59:58+10:00" name="To go through the routine"
date="2008-05-16T09:59:58+10:00"/>
      <resources>
         <resource>
            <resid id="http://www.xmlspy.com">0002</resid>
            <resname>VT Client</resname>
          </resource>
       </resources>
  <requirements>

<contributions>
        <contribution>
            <id>Lin089</resid>
            <resname>VT Client</resname>
        </contribution>
        <contribution>
            <id>Ner005</resid>
            <resname>VT Client</resname>
        </contribution>
 </contributions>

<agreements>
    <ns4:Signature>
       <ns4:SignedInfo>
          <ns4:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315"/>
          <ns4:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
          <ns4:Reference URI="">
             <ns4:Transforms>
                <ns4:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
             </ns4:Transforms>
             <ns4:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
```

Fig. 6. An eContract signed by two participants in our prototype system

```
            <ns4:DigestValue>+llp4pVrVBr3h/RnIAdIXTDas+Y=</ns4:DigestValue>
          </ns4:Reference>
        </ns4:SignedInfo>
<ns4:SignatureValue>DZn8HWsXBtc5rUM9Os5SHWpOQ3/gcqGfO1FPtxwk+g8FnU7n3xztLsLubMu
St5BLbP…….</ns4:SignatureValue>
        <ns4:KeyInfo>
          <ns4:X509Data>

<ns4:X509SubjectName>CN=Ner005,OU=Unknown,O=Unknown,L=Unknown,ST=Unknown,C=Unkn
own</ns4:X509SubjectName>

<ns4:X509Certificate>WCZBfPsCl+7r2//Z76DiEFCrLZgDn0GYNPFBBZr2aY4V2MTSAy3xi.....</ns
4:X509Certificate>
          </ns4:X509Data>
        </ns4:KeyInfo>
    </ns4:Signature>
    <ns4:Signature>
        <ns4:SignedInfo>
          <ns4:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315"/>
          <ns4:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
          <ns4:Reference URI="">
            <ns4:Transforms>
              <ns4:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
            </ns4:Transforms>
            <ns4:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <ns4:DigestValue>+llp4pVrVBr3h/RnIAdIXTDas+Y=</ns4:DigestValue>
          </ns4:Reference>
        </ns4:SignedInfo>

<ns4:SignatureValue>FlGysAmLkxw/oAFk6I2Y0yig/c3wctle+9pr8xvVpMAZv35DfGQ4nXB…….
</ns4:SignatureValue>
        <ns4:KeyInfo>
          <ns4:X509Data>

<ns4:X509SubjectName>CN=Ner005,OU=Unknown,O=Unknown,L=Unknown,ST=Unknown,C=Unkn
own</ns4:X509SubjectName>

<ns4:X509Certificate>MIICTTCCAbagAwIBAgIESCkMWjANBgkqhkiG9w0BAQUFADBrMRAwDg
Y……</ns4:X509Certificate>
          </ns4:X509Data>
        </ns4:KeyInfo>
    </ns4:Signature>
  </agreements>
</econtract>
```

Fig. 6. (*continued*)

collaborators must have previously and successfully registered themselves with the contract service, and download and install the client application. Collaborator A runs the client application and initializes contract by first discovering, and then adding the collaborators B and C as participants as shown in Figure 4 (a).

As an initiator, collaborator A also specifies the resources required for the collaboration such as the VT application, as well as any of A's contributed resources in the contract. Once the requirement and contribution are specified, the contract is

submitted to the contract service which will then go through the different phases as discussed earlier. During the negotiation phase, collaborators B and C will be informed that they are invited to participate in the collaboration and can negotiate the resources with A. They can decide to accept or decline the invitation. If they decide to join the collaboration, each can negotiate the content in the eContract with all the other participants following a negotiation protocol [12]. Once all participants agree upon and sign the eContract as shown in Figure 4(b), the eContract becomes agreed contract.

The agreed contract then goes through the validation phase, where resources and policies are checked. It should be noted that we have not yet implemented the validation part in our prototype system and for the purposes of this prototype, we assume that all agreed contracts are valid. The validated contract is then executed during the insanitation phase. As part of the execution, the contract service extracts the necessary information and sends it to the VPN service. The VPN service then establishes a dynamic intranet whose behavior is determined by the defined and agreed upon contract between the participants. The VPN service then automatically sends and installs a VPN driver into each of the participants' machines, and if required, dispatches the VT application as well. The collaborators are then ready to use VT to share documents and files. Figure 5 shows an example contract generated by our implemented system. Here, the aim is to establish network connectivity, allowing the collaboration using the VT between the participants to proceed. When the collaboration is completed and all parties agreed to terminate the contract, a termination protocol is executed resulting in the contract being terminated.

In our prototype, there are three major components that are implemented as follows. We use Sun Glassfish v2.0 to implement the contract Web Service with the support of MySQL v5.122 as a backend database. The VPN server was implemented using OpenVPN v2.1. The client application is implemented using Sun's JDK 1.6, including Swing and JAXB.

Through the implementation of above discussed connectivity service, we have shown how one can use our contract language as a template to (a) capture the requirements of the collaboration, (b) contribute resources as services in the collaboration, (c) negotiate resources for the collaboration, (d) capture agreements between collaborators, and (e) instantiate and terminate the collaboration.

6 Conclusions and Future Work

The paper presented a contract language for defining a context for a dynamic collaboration between partners. The language is used to generate the templates that can be used to automatically configure the collaboration. The templates, which we refer as electronic contracts, are also used to negotiate resources (that can be expressed as Web Services) and policies. Hence, the language is called Web Service Collaborative Context Definition Language (WS-CCDL). The language has been defined using XML Schema. We have also developed a service-oriented prototype system for an instance of a collaborative environment to provide connectivity service to the collaborating partners. The prototype system provided us evidence that it is feasible to develop a contract driven dynamic collaboration using our proposed

language. We have finally devised a runtime framework for the contracts consisting four phases: negotiation, validation, instantiation and termination.

The proposed language in its current form only captures some core elements for negotiation, validation and instantiation. We plan to extend the language and propose a Quality of Service (QoS) model for it. The QoS model is expected to captures obligations in terms of security, privacy, trust, performance and availability. We then plan to propose a mechanism for monitoring those obligations. With regard to validation of contract, we are also working on defining formal models as well as algorithms for checking resource and policy satisfactions so that all participants' requirements needed for the completion of tasks within the collaboration are met. Finally, we are also looking at the specification of different termination protocols within the contract similar to that of negotiation and agreement protocols.

References

[1] Mowshowitz, A.: Virtual Organization: A vision of management in the information age. The Information Society 10(4), 267–288 (1994)

[2] Foster, I., Kesselman, C., Tuecke, S.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of High Performance Computing Applications 15(3), 200–222 (2001)

[3] Globus, http://www.globus.org/grid_software/monitoring/

[4] Yamazaki, Y.: Dynamic Collaboration: the model of new business that quickly responds to changes in the market through The integrated IT/Network Solutions provided by NEC. NEC Journal of Advanced Technology 1(1), 9–16 (2004)

[5] Handley, H.A.H., Wentz, L., Levis, A.H.: Continuity in Dynamic Coalition Operations. In: Proc. 7th Int'l Command and Control Research and Technology Symposium, Monterey, CA (June 2002)

[6] Chan, J., Rogers, G., Agahari, D., Moreland, D., Zic, J.: Enterprise Collaborative Contexts and their Provisioning for Secure Managed Extranets. In: Proc. of IEEE WETICE 2006, pp. 313–318 (2006)

[7] Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M.: Web Services Agreement Negotiation Specification (WS-AgreementNegotiation), version 1, http://forge.ogf.org/sf/go/doc6092?nav=1

[8] Hung, P.C.K., Li, H., Jeng, J.J.: WS-Negotiation: An overview of research issues. In: Proc. of the 37th Hawaii International Conference on System Sciences (2004)

[9] Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Heiko, L.: WS-Agreement Specification (2005), http://www.gridforum.rg/Meetings/GGF11/Documents/draft-ggf-graap-agreement.pdf2005

[10] Ludwig, H., Keller, A., Dan, A., King, R.P., Franck, R.: Web Service Level Agreement (WSLA) Language Specification (2003), http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf

[11] Nepal, S., Chan, J., Chen, S., Moreland, D., Zic, J.: An Infrastructure Virtualisation SOA for VNO-based Business Models. In: IEEE International Conference on Services Computing (SCC 2007), July 2007, pp. 41–51 (2007)

[12] Nepal, S., Zic, J., Chan, J.: A distributed Approach for Negotiating Resource Contributions in Dynamic Collaboration. In: The Eight International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT 2007), December 3-6, 2007, pp. 82–86 (2007)
[13] Chen, S., Nepal, S., Chan, J., Moreland, D., Zic, J.: Virtual Storage Services for Dynamic Coalitions. In: Proceedings of IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises, WETICE (2007)
[14] Khurana, H., Gligor, V.D.: A Model for Access Negotiations in Dynamic Collaborations. In: Proc. of the 13th IEEE WETICE, 2004, pp. 205–210 (2004)
[15] Freudenthal, E., Pesin, T., Keenan, E., Port, L., Karamcheti, V.: dRBAC: Distributed Role-Based Access Control for Dynamic Collaboration Environments. In: Proc. of the ICDCS 2002, pp. 411–420 (2002)
[16] Patz, G., Condell, M., Krishnan, R., Sanchez, L.: Multidimensional Security Policy Management for Dynamic Collaborations. In: DARPA Information Survivability Conference and Exposition (2001)
[17] Keller, A., Ludwig, H.: Defining and Monitoring Service-Level Agreements for Dynamic e-Business. In: 16th System Administration Conference, pp. 189–204 (2002)
[18] Department of Education, Science and Training, Australia. An Australian e-Research Strategy and Implementation Framework. Report, 4/2006
[19] Ma, D.: The Business Model of "Software-as-a-Service". In: SCC 2007, pp. 701–702 (2007)
[20] Microsoft Enterprise Collaboration, http://download.microsoft.com/download/c/6/0/c6003d74-2f58-4868-a8ff-172576303864/CollaborationBizOverview.pdf
[21] Baltic eHealth, http://www.ehealthconference.info/StockholmConferenceBrochure.pdf
[22] Kentucky eHealth, http://ehealth.ky.gov/NR/rdonlyres/DE96BBFC-6AE5-4A80-BA62-44B1D233514C/0/PrivacySecurityFinalReport.pdf
[23] ECOSPACE, http://www.ip-ecospace.org/
[24] XACML, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml
[25] EPAL, http://www.zurich.ibm.com/security/enterprise-privacy/epal/
[26] SecPAL, http://research.microsoft.com/projects/secpal/
[27] Kavantzas, N., Burdett, D., Ritzinger, G.: Web Services Choreography Description Language, http://www.w3.org/TR/2004/WD-ws-cdl-10-20040427/
[28] Nepal, S., Zic, J., Chen, S.: WSLA+: Web Service Level Agreement Language for Collaborations. In: IEEE International Conference on Service Computing (SCC), Hawaii, USA, July 8-11 (2008) (to appear)
[29] Nepal, S., Zic, J., Chen, S.: WS-CCDL: A Framework for Web Service Collaborative Context Definition Language for Dynamic Collaborations. In: IEEE International Conference on Web Services (ICWS), Beijing, China, September 23-26 (2008)