Abstract
Dynamic collaborations involve contributed resources across the organisational boundaries that are subjected to different set of policies. The management of such resources for dynamic collaborations including negotiation, validation, instantiation and termination is difficult. Existing approaches for collaborations using Web Services such as WSLA are designed to deal with scenarios involving two parties: a service provider and a service consumer. These approaches do not scale well to multiparty nature of dynamic collaborations. To address this problem, we propose a framework for a language called Web Service Collaborative Context Definition Language for dynamic collaborations. The language itself has been defined using XML Schema and has been implemented in a dynamic collaboration platform.

Keywords: Dynamic Collaborations, Service Level Agreements, Virtual Organisation, Contract.

1. Introduction
The idea of collaboration among organizations has been explored in the area of virtual organizations [1], where this goal is typically achieved through open service discovery, negotiation and execution based on service level agreements (SLAs). However, in contrast to the virtual organization model, there is a growing trend where virtual enterprises, built around transient business interests, require their contributed resources to remain hidden in a limited manner until revealed to a select group of collaborating participants [2][3][4]. Collaborations built around this concept are termed dynamic collaborations.

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, a concept of Software-as-a-Service (SaaS) is introduced for software [9] and Infrastructure-as-a-Service (IaaS) [6] for storage and networking infrastructure. Therefore, it is possible to define and share resources as services in the context of dynamic collaborations.

The management of contributed resources during a dynamic collaboration includes 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. Keller and Ludwig [8] defined a Web Service Level Agreement (WSLA) framework [5] for defining and monitoring SLAs between service providers and service consumers in an electronic commerce scenario. This framework works well in these types 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:

- Complex policies define the interactions, access and use of resources.
- Multiple parties are involved;
- Multiple resources (services) are contributed 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 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. In this paper, we briefly describe the semantics of the framework. The language itself is defined using XML Schema and implemented in an environment [10].
2. WS-CCDL Framework

Figure 1 shows the main concepts and their relationships within WS-CCDL. We next describe these concept definitions briefly. The following definitions assume the existence of some basic types:

- ANYTYPE: the set of all types;
- POLICYTYPE: the set of all policy language types such as EPAL, XACML;
- RESOURCETYPE: the set of all resources expressed as WSDL;

**Definition 1:** Attribute (keyword: Attr) An attribute refers to a name-value pair used to express certain characteristics of the entities involved in the collaborations. Both name and value are expressed as a value taken from ANYTYPE. The set of attributes associated with entities is called ATTRIBUTE. An attribute name within the ATTRIBUTE is unique.

The collaboration is driven by policies. Different organizations participating in the collaboration are operating under their own set of policies. In order to represent policies within our framework, we define policy as follows.

**Definition 2:** Policy (keyword: Pol) A policy refers to an access policy defined for an associated entity. The policy is expressed as POLICYTYPE, and may involve multiple entities. The set of policies associated with an entity is called POLICY.

A collaboration has a number of participants. In general, participants are categorized in this paper as follows: initiator, invited, signatory, and contributing. Contributing participants are defined as resources. In our framework, we define the signatory participants as follows.

**Definition 3:** Participant (keyword: P) A participant refers to a particular named collaborator with a set of attributes. The identification is based on a unique identity, expressed as ANYTYPE. The set of participants associated with a collaboration is called PARTICIPANT. An identity of a participant within the PARTICIPANT is unique. The participant may also have a set of attributes.

A unique characteristic of a dynamic collaboration is that it is formed by combing the resources contributed by participants. In our framework, signatory participants contribute resources towards the collaboration. The resources include contributing participants, software, data, tools and information resources. The resources are formally represented as follows.

**Definition 4:** Resource (keyword: Res) A resource refers to an entity that the participant contributes to a collaboration. The identification of resource is based on a unique identity, expressed as RESOURCETYPE. The set of resources associated with a collaboration is called RESOURCE.

With the emergence of Web Services, it is now possible to define these resources as services and contribute them to the collaboration. The definition of a resource as a service is outside the scope of this paper. However, we have implemented storage and networking infrastructures as services in our prototype implementation of the framework as defined in [6].

Within a collaboration, participants may engage in a number of activities, and each activity may need different set of participants and resources. In our framework, we represent such activities as follows.

**Definition 5:** Activity (keyword: Act) An activity refers to tasks in a collaboration. The identification of activity is based on a unique identity, expressed as ANYTYPE. An activity comprises of resources, policies, and participants. The set of activities within a collaboration is called ACTIVITY.

An initiator – the participant who initiates the collaboration - expresses the need of the collaboration through requirement. In order to capture these initial statements about the collaboration, we define the requirement as follows.

**Definition 6:** Requirement (keyword: Re) The requirement specifies the activities, resources, and policies that need to be satisfied for the collaboration to occur. The set of requirements associated with a collaboration is called REQUIREMENT.

It is also important to note that these requirements are negotiable. 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. We define such contribution as follows.

**Definition 7:** Contribution (keyword: C) A contribution refers to contributions made by a participant towards a collaboration. The contributions by participants include activities, resources, and policies. The
set of all contributions associated with a collaboration is called CONTRIBUTION.

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 in our framework as follows.

**Definition 8:** Agreement (keyword: A) An agreement refers to an acceptance of a participant’s contribution by other participant. The agreement includes contributor, contribution, and acceptor. The set of all agreements within a collaboration is called AGREEMENT.

We have already defined all the concepts except contract shown in Figure 2. We next define the contract using the above defined concepts as follows.

**Definition 9:** Contract (keyword: Con) A contract is a collaborative context that specifies not only the requirements for the collaboration but also captures the contributions made by the participants. The contract includes collaboration requirements, contributions by participants, agreements among participant and the list of participants. The set of contracts is called CONTRACT.

\[
\begin{align*}
\text{contract} \in \text{CONTRACT} \\
\text{Re} = \{\text{Re} \in \text{REQUIREMENT}\} \\
\text{C} = \{\text{C} \in \text{CONTRIBUTION}\} \\
\text{A} = \{\text{A} \in \text{AGREEMENT}\} \\
\text{P} = \{\text{P} \in \text{PARTICIPANT}\} \\
\text{attr} = \{\text{attr} \in \text{ATTRIBUTE}\}
\end{align*}
\]

The above defined framework provides a template for defining a language for describing contract for dynamic collaboration. We have defined one such language using XML schema. The high-level elements of the language are shown in a snapshot of XML-schema below.

As the paper focuses on the semantics of the framework, the detailed discussion on the language and its implementation in a collaborative context is presented in [cite].

**Figure 2** Different phases of the contract in a runtime environment.

### 3. WS-CCDL Runtime Framework

The contract goes through the different phases in its lifecycle as shown in Figure 2.

**Negotiation Phase:**

The negotiation phase involves negotiation among participants about the terms and conditions of the collaboration including contributed resources and policies attached to them. A set of Web Services standards for negotiations have been proposed for this purpose: WS-AgreementNegotiation, WS-Negotiation, and WS-Agreement. Web Services proposals for negotiation works well in electronic commerce scenario between two parties, but they are not applicable/suitable in multi-party, multi-services negotiation in the context of dynamic collaborations. We have defined protocols for dynamic collaboration in [7]. The unique characteristic of such protocol 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, which we define as follows.

**Definition 10:** Agreed Contract (keyword: Agreed-Con) The Contract is said to be agreed if all parties have seen the final contributions from other parties and have agreed with it.

\[
\begin{align*}
\text{agreedcontract} \in \text{AGREEDCONTRACT} \\
\text{Condition} = \{\text{Con} \in \text{CONTRACT}\} \\
\text{Condition} = \{\text{Con} \in \text{PARTICIPANT}, \text{Con} \in \text{CONTRIBUTION}, \text{Con} \in \text{AGREEMENT}\}
\end{align*}
\]

**Validation Phase**

The next phase is called validation phase where the contract is validated against two criteria and determines 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. We also need to ensure that the resource satisfaction for a
contract meet the policy conditions. We define these checks as follows.

**Definition 11:** Resource Satisfied The contract is said to be Resource satisfied if the contributed resources meet all the resource requirements of the contract under the given policy. Let

\[
\begin{align*}
Ctnt.Rsrc &= \text{CONTRIBUTION.RESOURCE} \\
Rq.Rsrc &= \text{REQUIREMENT.RESOURCE} \\
Ctnt.Act.Rsrc &= \text{CONTRIBUTION.ACTIVITY.RESOURCE} \\
Rq.Act.Rsrc &= \text{REQUIREMENT.ACTIVITY.RESOURCE}
\end{align*}
\]

Then ResourceSatisfied(C)=

\[
\forall C \in \text{CONTRACT} : \left\{ \begin{array}{ccc}
Ctnt.Rsrc \supseteq C.Rq.Rsrc \\
\end{array} \right. 
\]

**Definition 12:** Policy Satisfied The contract is said to be policy satisfied if the policies expressed in the contract are not conflicting each other. Let

\[
\begin{align*}
Ctnt.Pol &= \text{CONTRIBUTION.POLICY} \\
Ctnt.APol &= \text{CONTRIBUTION.ACTIVITY.POLICY} \\
Rq.APol &= \text{REQUIREMENT.ACTIVITY.POLICY} \\
P.Pol &= \text{PARTICIPANT.POLICY} \\
\forall C \in \text{CONTRACT} : \text{PolicySatisfied}(C) =
\end{align*}
\]

\[
\begin{align*}
Ctnt.Pol Rq.APol \& Ctnt.APol = \text{null}
\end{align*}
\]

**Instantiation Phase:**

This phase involves interpretation of contract. The interpretation of contract is performed by an instantiation engine and results in an instantiation of the collaboration. Instantiability of the contract is defined as follows.

**Definition 13:** Instantiable The contract is said to be instantiable if it is agreed by all participants, and is both policy and resource satisfied.

\[
\forall C \in \text{CONTRACT} : \text{Instantiable}(C) =
\]

\[
\begin{align*}
Ctnt.AGREEDCONTRACT \& \text{PolicySatisfied}(C) \& \text{ResourceSatisfied}(C)
\end{align*}
\]

**Termination Phase**

The last phase of the runtime environment is called termination, where the collaboration is terminated as per contract or in an agreed manner. We define one such termination protocol using WS-Business/Activity in [7]. Other distributed termination protocols can be used.

4. Discussions and Future Works

The paper presented a framework for a 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 framework is called Web Service Collaborative Context Definition Language (WSCCDL). We have also devised a runtime framework for contracts. A language for based on the framework has been defined using XML Schema. The framework is also prototyped for an instance of a collaborative environment to connect network resources (see [10]). The prototype system provided us evidence that it is feasible to develop a contract driven dynamic collaboration using our proposed framework.

It is important here to emphasize that the framework in its current state only captures the necessary elements for negotiation, validation and instantiation. At present, our framework captures obligations at the coarse level such as resources are available for the agreed duration. We plan to extend it to capture obligations at greater details as in WSLA. Similarly, we plan to extend the runtime framework by introducing a phase for monitoring those obligations. Further work will also include the detailed analysis of the different phases of the contract and the validation process.

5. References