Available online at www.sciencedirect.com
SCIENCE DIRECT®
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 146 (2006) 43–57
www.elsevier.com/locate/entcs

Ontologies for Specifying and Reconciling Contexts of Web Services

S. Sattanathan[1]

National Institute of Technology Karnataka, Surathkal, India

N. C. Narendra[2]

IBM Software Labs India, Bangalore, India

Z. Maamar[3]

Zayed University, Dubai, U.A.E

Abstract

This paper presents an ontology-based approach for the specification (using OWL-$\mathcal{C}$ as a definition language) and reconciliation (using $\mathcal{C}$on$\mathcal{W}$e$\mathcal{S}$ as a mediation tool) of contexts of Web services. Web services are independent components that can be triggered and composed for the satisfaction of user needs (e.g., hotel booking). Because Web services originate from different providers, their composition faces the obstacle of the context heterogeneity featuring these Web services. An unawareness of this context heterogeneity during Web services composition and execution results in a lack of the quality and relevancy of information that permits tracking the composition, monitoring the execution, and handling exceptions.

Keywords: Web services, Ontology, Context, Reconciliation.

1 Introduction & Motivation

Web services constitute a new approach for achieving Business-to-Business integration [15]. One of the strengths of Web services (also called services in

[1] Email: ss_nitk@yahoo.co.in
[2] Email: narendra@in.ibm.com
[3] Email: zakaria.maamar@zu.ac.ae

doi:10.1016/j.entcs.2005.11.006

this paper) is their capacity to be composed into high-level business processes known as composite services. Composition primarily addresses the situation of a user request that cannot be satisfied by any available service, whereas a composite service obtained by integrating available services might be used.

Several efforts are put in the development of standards for Web services (e.g., WSDL, SOAP, BPEL [13]) in terms of specification, discovery, selection, composition, just to cite a few. In [11], we highlighted the importance of dealing with the composition of Web services at three connected levels. The lower level is about the messages that Web services of a composite service exchange during interaction. The mid level is about the semantics of the content that these messages convey. The need for a common semantics is intensified when Web services, which originate from different providers, take part in the same composition. Finally, the higher level is about the context in which the composition of Web services takes place. By developing context-aware Web services it would be possible, for example, to consider the aspects of the environment surrounding Web services. These aspects are multiple and can be related to users (e.g., stationary user, mobile user), their level of expertise (e.g., expert, novice), computing resources (e.g., fixed device, handheld device), time of day (e.g., afternoon, morning), and physical locations (e.g., office, cafeteria).

Associating Web services with context is a response to the challenges that hinder the smooth automation of composition. In [11], we discussed some of these challenges such as which businesses have the capacity to provision Web services, when and where the provisioning of Web services occurs, and how Web services from independent providers coordinate their activities so that conflicts are avoided. Since Web services belong to different providers, their context definition is definitely different in terms of structure, number of arguments, name of arguments, meaning of arguments, etc. Ignoring the problem of context heterogeneity has side-effects on the normal progress of the composition of Web services. These side-effects are various such as adopting the wrong strategy for selecting a component Web service (e.g., favoring execution-cost criterion over reliability criterion, instead of the opposite), delaying the triggering of some urgent component Web services, or wrongly assessing the exact status of a Web service. Addressing the context heterogeneity of Web services is a two-step process. The first step consists of specifying contexts using a dedicated language. This language is OWL-$\mathcal{C}$ (**O**ntology **W**eb **L**anguage for $\mathcal{C}$ontext ontologies) and is detailed in terms of concepts, formalism and utilization in [11]. The second step consists of fixing the heterogeneity of contexts using mediation mechanisms. These mechanisms are supported by a prototype called $\mathcal{C}$on$\mathcal{W}$e$\mathcal{S}$ ($\mathcal{C}$ontext-based semantic $\mathcal{W}$eb $\mathcal{S}$ervices).

The rest of this paper is organized as follows. Section 2 presents some

basic definitions that make readers familiar with the concepts used in the paper. Section 3 overviews the $\mathcal{C}on\mathcal{W}e\mathcal{S}$ framework in terms of architecture and implementation. Section 4 is about related work and how it has impacted the design of our context ontology. Section 5 introduces our ongoing work on the security of Web services. Finally, we draw our conclusions in Section 6.

2 Background

`Web service` - is an application that other applications and humans can discover and invoke, and presents the following properties [2]: independent as much as possible from specific platforms and computing paradigms; primarily developed for inter-organizational situations; and easily composable so that developing complex adapters for the needs of composition is not required.

In [10], the Web services instantiation principle was put forward. Adhering to this principle, a Web service is instantiated each time it is invited for participating in a new composition. Prior to any invitation acceptance and instantiation, several elements of the Web service are checked. These elements constitute a part of the context of the Web service and are discussed in Section 3.1. The Web services instantiation principle offers the possibility of organizing a Web service along three temporal categories (Fig. 1): Web service instances already deployed, Web service instances currently deployed, and Web service instances to be deployed upon invitation acceptance.

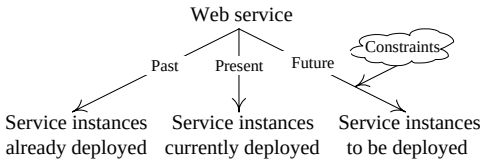

Fig. 1. Organization of a Web service

`Context` - is any information that is judged relevant to the interactions between a user and an environment [[4]. This information refers to the circumstances, objects, or conditions that surround the user. From a Web services perspective, we defined context as a set of common meta-data about the current execution status of a Web service and its capability of collaborating with peers, possibly enacted by distinct providers [11].

`Ontology` - is "*a logical theory accounting for the intended meaning of a formal vocabulary, i.e., its ontological commitment to a particular conceptualization of the world*" [5].

3 $\mathcal{C}$on$\mathcal{W}$e$\mathcal{S}$ framework

3.1 Architecture

Fig. 2 conceptualizes the approach of dealing with the heterogeneity of the contexts of Web services. The approach is built upon the fact that once Web service instances are created, they bind to appropriate ontologies. Binding means, here, that a Web service instance complies with a specific ontology for the needs of manipulating and adapting data when this service instance interacts with peers. The creation of Web service instances is subject to accepting the invitations of participation that originate from composite services to Web services (see [10] for more details). Once an invitation is accepted (could be rejected, too), the composite service informs the multiple component Web services about the ontology that their respective Web service instances should adopt during their data-manipulation operations. The ontology is related to the application domain (e.g., travel) in which the composition of Web services occurs. We assume that ontologies exist in a repository (Fig. 2). In the rest of this paper, we also assume that the specification of the component Web services is based on service chart diagrams [9]. To keep the paper self-contained, the specification of composite Web services is not featured.

Before the instantiation happens, several elements related to the Web service are checked. First of all, the number of Web service instances currently running *vs.* the maximum number of Web service instances that can be simultaneously run (i.e., instances of the same Web service). Each service instance has its execution load parameters (e.g., memory use, data-transfer volume), which differentiate it from other peers of the same Web service. Second, the execution status of each Web service instance that is part of a composite service. Third, the execution progress of the preceding Web service instances per Web service instance to be deployed in the future. This execution progress is required in case of data or control dependency between the service instances. Finally, the time that the composite service would like having a Web service instance made available for invocation *vs.* the time it would be possible for the Web service to have a Web service instance made available for invocation.

For management purposes of context, two operations, known as consolidation and reconciliation, are deemed appropriate. In Fig. 2, numbers between brackets represent the chronology of executing both operations.

- *Consolidation* at the level of Web services: when a Web service accepts an invitation of participation in a composite service (1), a Web service instance along with an $\mathcal{I}$-context is created (2). The transfer of details from the $\mathcal{I}$-contexts of the same Web service instances to the $\mathcal{W}$-context of their associated Web service is featured by a consolidation of these details

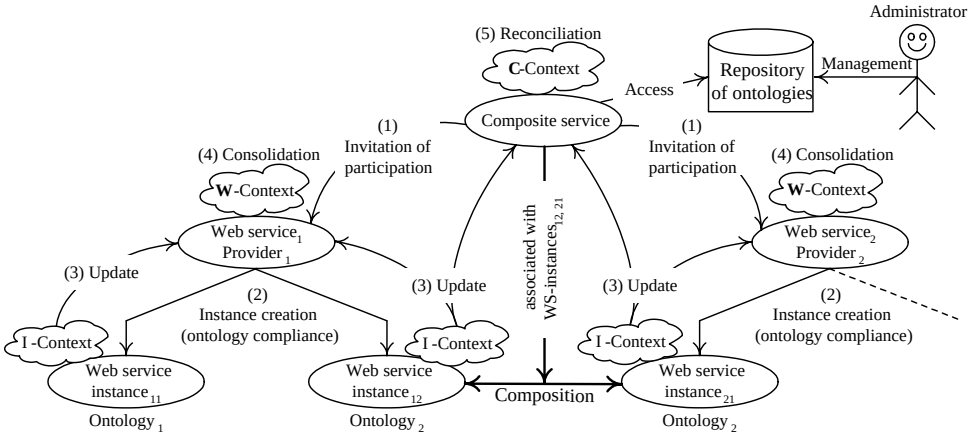

Fig. 2. Overall conceptual architecture of $\mathcal{C}$on$\mathcal{W}$e$\mathcal{S}$ framework

before this $\mathcal{W}$-context is updated (3,4). Once the consolidation is over, a Web service determines for each of its Web service instances the following: execution status, actions it has performed, and expected completion-time of execution.

- *Reconciliation* at the level of composite services (how context heterogeneity is handled): since the component Web services of a composite service have multiple providers, the definition of the $\mathcal{W}$-contexts and $\mathcal{I}$-contexts varies like number and name of arguments. The transfer of details from the $\mathcal{I}$-contexts of Web service instances to the $\mathcal{C}$-context of a composite service is featured by a reconciliation of these details before the $\mathcal{C}$-context is updated (3,5). For example it occurs that the $\mathcal{I}$-context of a Web service instance of a composite service contains "location of execution" argument, whereas the $\mathcal{I}$-context of another Web service instance of this composite service contains "site of execution" argument. During reconciliation, "execution site" and "execution location" have both to be considered the same. To ensure that the composite service recognizes the differences between the arguments of contexts, it refers to ontology of contexts that will be specified using OWL-$\mathcal{C}$. "Execution location" and "execution site" mean here the computing platform on which Web services operate.

3.2 OWL-$\mathcal{C}$ foundations

Fig. 2 consists of three levels of abstraction, where each level identifies a type of service. Contexts of Web service instances have the fine-grained content, whereas contexts of composite services have the coarse-grained content. The content of $\mathcal{I}$-context updates first, the content of $\mathcal{W}$-context after consolida-

tion and second, the content of $\mathcal{C}$-context after reconciliation, respectively. Since OWL-$\mathcal{C}$ takes advantage of the research findings of the Semantic Web community and its specification language OWL-S (Ontology Web Language-based Web Service Ontology, formerly DAML-S), the relation between both languages is depicted in Fig. 3, with OWL-$\mathcal{C}$ shown in yellow color. Context of a Web service is specialized into two types: $\mathcal{W}$-context (focus of this paper) and $\mathcal{WS}$ec-context (part of our future work as discussed in Section 5.1).

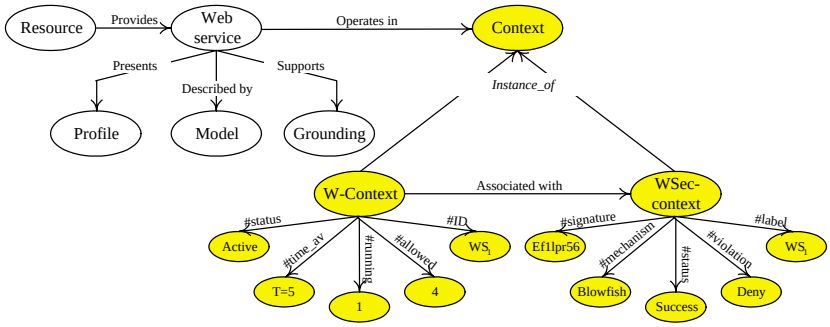

Fig. 3. Extensions to OWL-S ontology

Context specification using OWL-$\mathcal{C}$ includes two parts. The first part is about the arguments of context. The second part is about the capabilities of context. In the first part, context is considered as an extra argument that belongs to the structure of a Web service. Web services regularly post their arguments (e.g., identifier, execution cost, response time) on the external environment, after defining these arguments using WSDL for example. Because context is a multi-argument structure, OWL-$\mathcal{C}$ assists in the stipulated semantics of these arguments. As a result various parties can agree now on a common representation of the content of the context of Web services. With regard to the capabilities of context, a service that has a context needs to be embedded with awareness mechanisms. These mechanisms permit gathering contextual raw data from sensors, and detecting any change of the environment. A change needs to be assessed by the Web service through an assessment module, so that the Web service takes appropriate actions through a deployment module.

The structure of context consists of multiple arguments, whose number depends on the type of context. In what follows, we only list the arguments of each context type (more details are given in [11]).

- Arguments of $\mathcal{I}$-context of a Web service instance: label, status, previous service instances, next service instances, regular actions, beginning-time, ending-time (expected & effective), reasons of failure, corrective actions, and date.
- Arguments of $\mathcal{W}$-context of a Web service: label, number of service in-

stances allowed, number of service instances running, next service instance availability, status per service instance per composite service, and date.

- Arguments of $\mathcal{C}$-context of a composite service: label, previous Web service instances, current Web service instances, next Web service instances, beginning time, status per Web service instance, and date.

3.3 $\mathcal{C}onW\mathcal{S}$ prototype

The use of OWL-$\mathcal{C}$ is backed by an automatic tool: $\mathcal{C}onW\mathcal{S}$. $\mathcal{C}onW\mathcal{S}$ supports context definition (i.e., representation), context consolidation at the Web service level, and context reconciliation at the composite service level. In $\mathcal{C}onW\mathcal{S}$, OWL-$\mathcal{C}$ statements are represented as a triple structure consisting of subject, predicate, and object. Fig. 4 illustrates a typical OWL-$\mathcal{C}$ expression for $\mathcal{W}$-context. Subject is the source from which the arc leaves. Predicate is the property that labels the arc. Finally, object is the resource or literal pointed by the arc. In Fig. 4, $\mathcal{W}$-context is the subject (i.e., resource), {status, time_av, running, allowed, ID} are the predicates, and {Active, T=5, 1, 4, WS_1} are the objects.

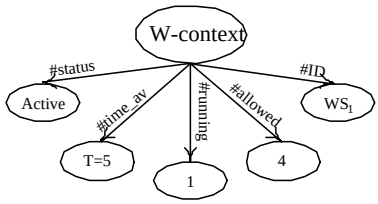

Fig. 4. Triple structure of $\mathcal{W}$-context

$\mathcal{C}onW\mathcal{S}$ is developed using the following tools and languages: Eclipse SWT (Standard Widget Toolkit) for getting the look-and-feel front-end environment, Jena for defining and processing OWL-$\mathcal{C}$ based context-ontologies, Core Java for integrating the aforementioned APIs, and Mindswap's OWL-S API for processing OWL-S based ontologies for getting service related information. For illustration purposes, the running example used in the paper is about a BookService, which delivers books to customers after receiving orders via a Web site. This is a composite service with two component Web services: BookFinder (finds a book supplier) and BookPayment (accepts payments for the book from the customer).

Context representation

Table 1 is the OWL-$\mathcal{C}$ representation of the $\mathcal{C}$-context of BookService. Table 2 is the OWL-$\mathcal{C}$ representation of the $\mathcal{W}$-context of BookFinder. Finally, Table 3 is the OWL-$\mathcal{C}$ representation of the $\mathcal{I}$-context of an instance of BookFinder.

Table 1
$\mathcal{C}$-context of BookService

```
<?xml version="1.0" encoding="WINDOWS-1252"?>
<rdf:RDF
    xmlns:Ccontext="http://www.nitk.ac.in/OWLC/Context/Ccontext#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description rdf:about="http://defaultURI/Ccontext#">
        <Ccontext:Date>17/12/2004</Ccontext:Date>
        <Ccontext:NextWebService>BookPayment</Ccontext:NextWebService>
        <Ccontext:Status>Active</Ccontext:Status>
        <Ccontext:BeginTime>20:30:45</Ccontext:BeginTime>
        <Ccontext:Label>BookService</Ccontext:Label>
        <Ccontext:CurrentWebService>BookFinder</Ccontext:CurrentWebService>
        <Ccontext:PreviousWebService>Nil</Ccontext:PreviousWebService>
    </rdf:Description>
</rdf:RDF>
```

Table 2
$\mathcal{W}$-context of BookFinder

```
<?xml version="1.0" encoding="WINDOWS-1252"?>
<rdf:RDF
    xmlns:Ccontext="http://www.nitk.ac.in/OWLC/Context/Wcontext#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description rdf:about="http://www.sp.com/Wcontext#">
        <Wcontext:Status>Active</Wcontext:Status>
        <Wcontext:Label>Book_Finder</Wcontext:Label>
        <Wcontext:InstanceRunning>2</Wcontext:InstanceRunning>
        <Wcontext:NextInstanceAvilability>true</Wcontext:NextInstanceAvilability>
        <Wcontext:InstanceAllowed>3</Wcontext:InstanceAllowed>
    </rdf:Description>
</rdf:RDF>
```

Context Consolidation

Consolidation happens at the level of Web services, and means the combination of details that stem from the Web service instances level to the Web service level. Once the consolidation is completed, a Web service is able to determine for each of its Web service instances the following: execution status, the actions it has performed, and the expected completion execution-time so that the Web service can commit additional Web service instances as per other composite services' requests. Fig. 5-(a) presents the initial values of the $\mathcal{W}$-context parameters of BookFinder Web service. In this figure, the focus is on "InstanceRunning" parameter (highlighted in green color). After the acceptance of two service requests, the consolidated version of $\mathcal{W}$-context

Table 3
$\mathcal{I}$-context of an instance of BookFinder

```
<?xml version="1.0" encoding="WINDOWS-1252"?>
<rdf:RDF
    xmlns:Ccontext="http://www.nitk.ac.in/OWLC/Context/Icontext#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description rdf:about="http://www.sp.com/Icontext#">
        <Icontext:NextServiceInstance>Nil</Icontext:NextServiceInstance>
        <Icontext:Status>Active</Icontext:Status>
        <Icontext:RegularAction>Finding a Book</Icontext:RegularAction>
        <Icontext:PreviousServiceInstance>Nil</Icontext:PreviousServiceInstance>
        <Icontext:Label>Book_Finder_Service_Instance_1</Icontext:Label>
        <Icontext:ReasonsOfFailure>Nil</Icontext:ReasonsOfFailure>
        <Icontext:CorrectiveAction>Nil</Icontext:CorrectiveAction>
    </rdf:Description>
</rdf:RDF>
```

shows that two Web service instances are running (Fig. 5-(b)). When one of these service instances completes its execution with success, the number of the current running instances drops to 1 (Fig. 5-(c)).

(a)

W-Context Parameters	Values
Label	Book_Finder
InstanceAllowed	3
InstanceRunning	0
NextInstanceAvilability	true
Status	Active

(b)

W-Context Parameters	Values
Label	Book_Finder
InstanceAllowed	3
InstanceRunning	2
NextInstanceAvilability	true
Status	Active

(c)

W-Context Parameters	Values
Label	Book_Finder
InstanceAllowed	3
InstanceRunning	1
NextInstanceAvilability	true
Status	Active

Fig. 5. Representation of consolidation of contexts

Context Reconciliation

Reconciliation happens at the level of a composite service, since the component Web services of this composite service have multiple providers, and the definition of their respective $\mathcal{W}$-contexts varies in terms of structure and content. The transfer of details from the $\mathcal{I}$-contexts of the Web service instances to the $\mathcal{C}$-context of a composite service is featured by a reconciliation of these details before the $\mathcal{C}$-context is updated.

Fig. 6 summarizes the reconciliation as supported by $\mathcal{C}$on$\mathcal{W}$e$\mathcal{S}$. Fig. 6-(a) and -(b) show the initial status of the $\mathcal{W}$-context of BookFinder as well as the initial status of the $\mathcal{C}$-context of BookService after BookFinder accepts the invitation of participation of BookService. PreviousWebService, CurrentWebService, and NextWebService parameters are significant in $\mathcal{C}$on$\mathcal{W}$e$\mathcal{S}$. It can be seen for instance that BookService will execute two component Web services sequentially namely BookFinder and BookPayment. As shown in Fig. 6-(b),

(a)

W-Context Parameters	Values
Label	Book_Finder
InstanceAllowed	3
InstanceRunning	0
NextInstanceAvilability	true
Status	Active

(b)

C-Context Parameters	Values
Label	Book Service
Status	Active
PreviousWebService	Nil
CurrentWebService	Book Finder
NextWebService	Book Payment

(c)

I-Context Parameters	Values
Label	Book_Finder_Service_Instance_1
Status	Active
PreviousServiceInstance	Nil
NextServiceInstance	Book_Payment_Instance_1
RegularAction	Finding a Book
ReasonsOfFailure	Nil
CorrectiveAction	Nil

(d)

I-Context Parameters	Values
Label	Book_Payment_Instance_1
Status	Waiting
PreviousServiceInstance	Book_Finder_Instance1
NextServiceInstance	Nil
RegularAction	Getting Payment for Book
ReasonsOfFailure	Nil
CorrectiveAction	Nil

(e)

I-Context Parameters	Values
Label	Book_Payment_Instance_1
Status	Active
PreviousServiceInstance	Book_Finder_Instance1
NextServiceInstance	Nil
RegularAction	Getting Payment for Book
ReasonsOfFailure	Nil
CorrectiveAction	Nil

(f)

C-Context Parameters	Values
Label	Book Service
Status	Active
PreviousWebService	Book Finder
CurrentWebService	Book Payment
NextWebService	Nil

Fig. 6. Representation of Reconciliation of contexts

BookFinder is under execution whereas BookPayment is expected to be initiated upon completion of this execution. Fig. 6-(d) presents the $\mathcal{I}$-context of an instance of BookPayment service. The instance has a waiting status (highlighted in green), i.e., waiting for the completion of its previous Web service instance namely Book_Finder_Service_Instance_1. Once the execution of this instance is over (Fig. 6-(c)), Book_Payment_Service_Instance_1 will be changed to active (Fig. 6-(e)). The relevant parameters of the $\mathcal{C}$-context of BookService are also updated following the successful execution of the Web service instances (Fig. 6-(f)).

4 Related Work

While the concepts of Web services, composition, ontology, and context are independently studied by academia and industry (except for Web services composition and ontology), our work aims at their combination. Backing the idea of having a specification language for context, Hegering et al. note that automating contextualization and using contextual information across organizational boundaries can only be done if all participants agree on how to interpret context [6]. Thus, the management information model has to be extended with a context description language, which formalizes context. A starting strategy to uniform context is to build context categories, such as device-specific, environment-specific, and user-specific context.

While we strengthened the importance of reconciling contextual information using ontologies, Keidl and Kemper do not see any motive to that reconciliation [7]. For both authors, context encompasses all the information about the client of a Web service that may be utilized by the Web service to adjust execution and output delivery, so that the client can benefit from a customized and personalized behavior [7]. Plus, both authors differentiate between the parameters of context and the parameters of a Web service. We claim that there is no need to exchange contextual information if the recipient Web service does not understand this information and thus, cannot adapt its behavior according to the context of other Web services. A common understanding of the information exchanged is required, backing thuis our context reconciliation efforts.

The importance of formal modeling of context is stressed by Shehzad et al. [16]. The authors consider context-awareness as an important ingredient of most ubiquitous applications today. The collaboration of these applications calls for a shared understanding of context in a way that contextual information is effectively communicated among them. The shared understanding could happen only if a formal model of context exists. This model has a number of advantages such as the possibility of storing context for a long term since its meaning will remain the same for future uses, and the widespread use of context by various applications.

One of the relevant uses of context is during Web services selection. Verheecke et al. argue that another limitation encountered in the field of Web services is that Web services can only be selected based on the functionality they offer [18]. WSDL-based Web services documentation does not support the explicit specification of the non-functional requirements such as constraint-based on QoS, access rights, and management statements. While we back the statements of Verheecke et al., we consider that context is suitable for hosting non-functional requirements that are dynamic by nature.

Some research on using UML for modeling context-based security parameters is presented in [1,17]. These papers describe extensions to the existing UML language in order to model contextual information for web services pertaining to constructs for safeguarding context security. We are currently investigating how to incorporate these ideas into our framework.

5 Ongoing work

5.1 Policies for Web services security

With Web services relying on the insecure Internet for mission-critical transactions, security is a major concern. In previous works [10], we argued that

because Web services require resources on which they execute, it is important to ensure that neither the services misuse the resources nor the resources alter the integrity of the services. Currently, a range of XML-based security techniques exist in order to protect Web services when it comes to authentication, role-based access control, messaging, and data integrity. It should be noted, however, that these techniques are statically determined at design-time and cannot be adjusted during the execution life-cycle of Web services without going through an extensive programming exercise.

In order to develop security strategies for Web services, we are in the process of developing a dynamic approach that takes advantage of context ontologies. The approach puts forward a new type of context called security (to be denoted by $\mathcal{CS}/\mathcal{WS}/\mathcal{IS}$ec-context per type of service) and security policies, to be both part of the $\mathcal{C}$on$\mathcal{W}$e$\mathcal{S}$ framework. The primary use of policies is to take actions according to the occurring events and detected changes that affect the security of Web services. Policies are to be defined according to the present context of a service whether Web, composite, or instance. Some of the elements that could be identified through the use of a security context are multiple such as the identification of the security violations that have happened and the corrective actions that have been taken in case of any attempt of misusing a resource. In [8], Kouadri Mostéfaoui and Brézillon considered context-based security to adapt the security strategy depending on a set of relevant information collected from the dynamic environment. While there is no disagreement on this definition, we deem appropriate to expand it. The objective is to consider context-based security as a means for tracking all the concerns and threats that affect the security in a certain context, which permits deploying appropriate security mechanisms while relying on previous security contexts.

Fig. 7 presents the way the connection between security policies, service contexts, and security contexts happens. In this figure, the configuration of security policies is tuned based on the information that contexts of services cater. Once these contexts are updated after consolidation and reconciliation [4] operations, the security contexts are notified about the integrity of this content. If this content has been subject to any alteration, the security policies are adjusted to cope with this alteration. We are currently investigating the use of the Ponder policy specification language [3] for specifying/updating the policies. Ponder is an object-oriented policy language for the management of distributed systems and networks. It provides a unified framework for specifying policies for security management in large-scale distributed systems,

[4] The efficiency of the security policies depends on how the context heterogeneity is dealt with, since wrong information leads to inappropriate policy.

thereby making it suitable for web services security. A policy language such as Ponder also helps separate the specification of policy from its enforcement, since policies in Ponder are specified declaratively.

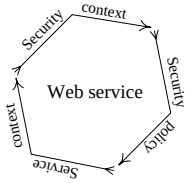

Fig. 7. Connection between context and policies

The distinction between service context ($\mathcal{C}/\mathcal{W}/\mathcal{I}$-context) and security context ($\mathcal{CS}/\mathcal{WS}/\mathcal{IS}$ec-context) permits the management of the aspects that each type of context is concerned with, in a better way (Fig. 7). A service context focuses on the changes that apply to a service (whether composite, Web, or instance) such as availability and commitment, whereas the security context focuses on the strategy of securing the interactions of services during data-context exchange. Various interactions occur between the two categories of context whether at the instance, Web, or composite level. Each service dynamically determines its security mechanisms based on the guidelines it receives from its respective $\mathcal{I}/\mathcal{W}/\mathcal{C}$-context. Initially, the threats that jeopardize the integrity content of $\mathcal{I}/\mathcal{W}/\mathcal{C}$-contexts are sensed and detected by verifying for example this content after submission from a service instance to a composite service. If there is any alteration, this means that the security mechanisms have to be reviewed through policies and announced at the level of /$\mathcal{IS}/\mathcal{WS}/\mathcal{CS}$ec-contexts.

5.2 Trust and reputation management of Web services

It is agreed that uncertainty has a great impact on the reliability of the information that Web services exchange, when mid-stream adaptation needs to be implemented. For example, if a Web service instance suddenly announces that it cannot perform its job, then either the Web service provider will have to instantiate a replacement service instance or the composite service provider will have to look for another Web service provider. This could impact the rest of the service instances that are dependent on the execution of the failed instance. In a similar situation, trust information needs to be used to decide whether to select a replacement service instance or start looking for a new Web service provider. In [14], the author defines trust as the characteristic that makes an entity willing to rely upon a second entity to execute a set of actions and make a set of assertions (usually dealing with identity) about a set

of subjects or scopes. Trust depends on the ability to bind unique attributes or credentials to a unique entity or user. When choosing a replacement instance or another Web service provider, the composite service would need to select the most trustworthy or most reputed Web service. Of course, trust information would be useful even during initial composition. A less trustworthy provider would be chosen, based on other considerations such as time, cost, etc. When an exception occurs, a more trustworthy provider would then be chosen, so that the exception does not occur again. We are currently working on enhancing our context ontologies so that trust and reputation information are incorporated [12].

6 Conclusion

In this paper an ontology-based approach for the specification and reconciliation of contexts of Web services has been presented. Because multiple providers supply Web services for potential compositions, a reconciliation of their respective contexts was deemed appropriate. Besides the multiple origins of Web services, disparities between contexts at the granularity level also exist as the three types of context ($\mathcal{I}/\mathcal{W}/\mathcal{C}$-context) have shown. The importance of having a language, e.g., OWL-$\mathcal{C}$, for context specification and management was also stressed. While it is argued that semantically described services will enable better service discovery, allow easier interoperability, and composition of services, we argue that semantically described context of services will enable better tracking and promote easier interoperability of Web services.

Acknowledgement. The first author is supported by the Center for Advanced Studies (CAS) program of IBM Software Labs India. The first author would also like to thank Prof. K. C. Shet of NITK, for supporting his doctoral work. The second author wishes to thank his manager, K. Muralidharan, for his support. IBM is a trademark of International Business Machines Corporation in the United States, other countries, or both. All the authors wish to thank the anonymous referees for their comments, which have significantly improved the quality of the paper.

References

[1] Basin, D. and J. D. T. Lodderstedt, *Model-driven Security: From UML Models to Access Control Infrastructures*, Technical Report 414, ETH, Zurich, Switzerland (2003).

[2] Benatallah, B., Q. Z. Sheng and M. Dumas, *The Self-Serv Environment for Web Services Composition*, IEEE Internet Computing **7** (2003).

[3] Damianou, N., N. Dulay, E. Lupu and M. Sloman, *The Ponder Specification Language*, in: *Proceedings of The Workshop on Policies for Distributed Systems and Networks (Policy'2001)*, Bristol, UK, 2001.

[4] Dey, A. K., G. D. Abowd and D. Salber, *A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications*, Human-Computer Interaction Journal **16** (2001).

[5] Gruber, T., *What is an Ontology? (white paper)*, http://www-ksl.stanford.edu/kst/what-is-an-ontology.html (2000).

[6] Hegering, H. G., A. Kpper, C. Linnhoff-Popien and H. Reiser, *Management Challenges of Context-Aware Services in Ubiquitous Environments*, in: *Proceedings of The 14th IFIP/IEEE Workshop on Distributed Systems: Operations and Management (DSOM'2003)*, Heidelberg, Germany, 2003.

[7] Keidl, M. and A. Kemper, *Towards Context-Aware Adaptable Web Services*, in: *Proceedings of The 12th International World Wide Web Conference (WWW'2003)*, Budapest, Hungary, 2003.

[8] Kouadri Mostéfaoui, G. and P. Brézillon, *Modeling Context-Based Security Policies with Contextual Graphs*, in: *Proceedings of The Workshop on Context Modeling and Reasoning (CoMoRea'2004) held in conjunction with The 2nd IEEE International Conference on Pervasive Computing and Communication (PerCom'2004)*, Orlando, Florida, USA, 2004.

[9] Maamar, Z., B. Benatallah and W. Mansoor, *Service Chart Diagrams - Description & Application*, in: *Proceedings of The Alternate Track of The 12th International World Wide Web Conference (WWW'2003)*, Budapest, Hungary, 2003.

[10] Maamar, Z., S. Kouadri Mostéfaoui and H. Yahyaoui, *Towards an Agent-based and Context-oriented Approach for Web Services Composition*, IEEE Transactions on Knowledge and Data Engineering **17** (2005).

[11] Maamar, Z., N. C. Narendra and W. J. van den Heuvel, *Towards an Ontology-based Approach for Specifying Contexts of Web Services*, in: *Proceedings of The Montreal Conference on e-Technologies (MCETECH'2005)*, Montreal, Canada, 2005.

[12] Maximilen, E. M. and M. P. Singh, *Conceptual Model of Web Service Reputation*, ACM SIGMOD Record **31** (December 2002).

[13] Milanovic, N. and M. Malek, *Current Solutions for Web Service Composition*, IEEE Internet Computing **8** (November/December 2004).

[14] Mysore, S., *Securing Web Services - Concepts, Standards, and Requirements (white paper)*, https://sdc.sun.com/kiosk/ViewPDF?pdf_id=L7WDTT4FYC (2003).

[15] Papazoglou, M. and D. Georgakopoulos, *Special Issue on Service-Oriented Computing*, CACM **46** (2003).

[16] Shehzad, A., H. Q. Ngo, K. Anh Pham and S. Y. Lee, *Formal Modeling in Context Aware Systems*, in: *Proceedings of The 1st International Workshop on Modeling and Retrieval of Context (MRC'2004)*, Ulm, Germany, 2004.

[17] Sheng, Q. Z. and B. Benatallah, *ContextUML: A UML-based Modeling Language for Model-Driven Development of Context-Aware Web Services*, in: *Proceedings of The 4th International Conference on Mobile Business!(mBusiness'2005)*, Sydney, Australia, 2005.

[18] Verheecke, B., M. A. Cibran and V. Jonckers, *AOP for Dynamic Configuration and Management of Web Services*, in: *Proceedings of The International European Conference on Web Services (ICWS'2003)*, Erfurt, Germany, 2003.