Available online at www.sciencedirect.com
ScienceDirect
Data & Knowledge Engineering 62 (2007) 327–351
DATA & KNOWLEDGE ENGINEERING
www.elsevier.com/locate/datak

Towards a context-based multi-type policy approach for Web services composition

Zakaria Maamar [a,*], Djamal Benslimane [b], Philippe Thiran [c], Chirine Ghedira [b], Schahram Dustdar [d], Subramanian Sattanathan [e]

[a] *Zayed University, Dubai, United Arab Emirates*
[b] *Claude Bernard Lyon 1 University, Lyon, France*
[c] *University of Namur, Namur, Belgium*
[d] *Vienna University of Technology, Vienna, Austria*
[e] *Airvana Networks India Private limited, Bangalore, India*

Received 6 April 2006; received in revised form 27 June 2006; accepted 12 August 2006
Available online 29 September 2006

Abstract

The objective of this research work is to look into the role of policies and context in framing the composition of Web services. Context supports the development of adaptable Web services and policies specify the behavior of Web services engaged in composition. In this paper, we present this research work's outcome, namely a context-based multi-type policy for Web services composition in which the binding between Web services occurs at four levels: component level to deal with Web services' definitions and capabilities, composite level to address how Web services are discovered and combined, semantic level to handle the semantic heterogeneity that can arise between Web services, and resource level to focus on the performance of Web services. Because of the composition complex-nature, the proposed approach uses three types of policies: engagement policies assess a Web service participation in a given composition, mediation policies deal with the semantic heterogeneity by tacking into account the mappings and conversions between component Web services, and deployment policies manage the interactions between component Web services and computing resources. Policies manage the transitions between the four levels and context caters the necessary information that tracks the composition progress.

Keywords: Composition; Context; Exception; Policy; Web service

* Corresponding author. Tel.: +971 42082461; fax: +971 42640854.
E-mail address: zakaria.maamar@zu.ac.ae (Z. Maamar).

0169-023X/$ - see front matter
doi:10.1016/j.datak.2006.08.007

1. Introduction

1.1. Preliminaries and motivations

The main motivation of this research is to demonstrate the benefits of integrating *context* and *policies* into a *Web services* composition approach.

Web services offer new opportunities to deploy Business-to-Business scenarios, which tend to crosscut companies' boundaries and to spread over large areas [29]. Web services are independent from specific platforms and computing paradigms, and have the capacity to form composed processes, referred to as composite Web services. Web services composition fulfills user requests that require the participation of several component Web services [7]. The specification of composite Web services is well studied [16], and different composition languages exist such as BPEL [10], WSFL [27], etc.

Policies are defined as "*information which can be used to modify the behavior of a system*" [28]. Our definition goes beyond behavior modification and considers policies as external, dynamically modifiable rules and parameters that are used as input to a system. This permits to the system to adjust to administrative decisions and changes in the execution environment [31]. The intended use of policies is to specify actions to be performed following event detections and context changes. Policies are in general viewed as a set of rules, constraints, or orders that control the behavior of an entity. In the field of Web services, policies are intended to specify different aspects of the behavior of a Web service. This helps a Web service align its capabilities and constraints to user requirements. Different languages for policy specification approaches exist including Ponder [11], SWRL [20], Rei [22], XACML [38], WS-Policy [46], and KAoS [50].

Context "*... is not simply the state of a predefined environment with a fixed set of interaction resources. It is part of a process of interacting with an ever-changing environment composed of reconfigurable, migratory, distributed, and multiscale resources*" [9]. In the field of Web services, context is used to facilitate the development and deployment of context-aware and adaptable Web services. Standard Web services descriptions can then be enriched with context information and new frameworks to support this enrichment can now be developed [25].

Three reasons motivate our adoption of policies in Web services composition. Firstly, policies permit managing Web services at a high level where guidelines that concern composition of Web services are separated from guidelines that concern for example, semantic mediation of Web services and deployment of Web services on top of resources. Secondly, the opportunity of changing policies without altering the specification of a composition. By changing policies, various aspects like Web services participation in a composite Web service and Web services security strategies can be continuously adjusted to accommodate environment variations. Finally, the third reason is the possibility of tracking the policies, whose actions have arisen exceptions and fixing them independently of the specification of a composition.

1.2. Contributions

While context and policies are separately used for different needs of Web services, the objective of this research is to look into their role in framing the composition process of Web services. Therefore, a multi-level approach for composing Web services is proposed. This approach does not only make Web services bind to each other, but emphasizes the cornerstone of refining this binding at four levels: *component* level ($\mathscr{W}$-level) to deal with Web services' definitions and capabilities, *composite* level ($\mathscr{C}$-level) to address how Web services are discovered and combined, *semantic* level ($\mathscr{S}$-level) to handle the semantic heterogeneity that can arise between Web services, and *resource* level ($\mathscr{R}$-level) to focus on the performance of Web services.

The role of policies and context to support the composition of Web services is depicted as follows:

- Policies manage the transitions between the four levels. Going from one level to another direct-level requires policy activation. Because of the complex, dynamic, and distributed nature of composition, several types of policies are required. Each type of policy addresses a specific aspect of the composition. We put forward three types of policies: *engagement* policies assess a Web service participation in a given composi-

tion, *mediation* policies deal with semantic heterogeneity by tacking into account the mappings and conversions between component Web services, and *deployment* policies manage the interactions between component Web services and computing resources.
- Context provides useful information concerning the environment wherein the composition of Web services occurs. Context caters the necessary information that enables tracking the whole composition process by enabling, for instance, triggering the appropriate policies and regulating the interactions between Web services according to the current state of the environment. Because of the four levels of the approach, four types of context are defined, namely $\mathscr{W}/\mathscr{C}/\mathscr{S}/\mathscr{R}$-context standing for context of Web service, context of composite Web service, context of ontology, and context of resource. Each context type is concerned with the specificities of each level.

1.3. Paper organization

Section 1 motivates and states the contributions of the research project discussed in this paper. Section 2 talks about related work in terms of context for Web services and policies for Web services. Section 3 presents our context-based multi-type policy approach for Web services composition. Section 4 describes the internal structure of the different types of contexts. Section 5 discusses the impact of policies on Web services and specifies the policies for the behavior of Web services. Section 6 is about exception handling in the context-based multi-type policy approach. Section 7 presents a prototype of the approach. Section 8 concludes the paper with some directions for future work. It should be noted that the way component Web services are discovered, selected, and appended into a composite Web service is outside this paper's scope.

2. Related work

Context and policies are separately explored for the needs of Web services. This related work is seen along two perspectives: context for Web services and policies for Web services.

2.1. Context for Web services

In literature the first time "context-aware" appeared was in [45]. Schilit and Theimer describe context as location, identities of nearby people, objects and changes in those objects. Such enumerations of context examples were often used in the beginning of context-aware systems research. Ryan et al. referred to context as the user's location, environment, identity, and time [43]. Dey defines context as the user's emotional state, focus of attention, location and orientation, date and time, as well as objects and people in the user's environment [15]. Another common way of defining context was the use of synonyms. Hull et al. describe context as the aspects of the current situation [21]. These kinds of definitions are often too wide. One of the most accurate definitions is given by Dey and Abowd [13]. They refer to context as "*any information that can be used to characterize the situation of entities* (*i.e.*, *whether a person, place or object*) *that are considered relevant to the interaction between a user and an application*, *including the user and the application themselves*". One popular way to classify context instances is the distinction of different context dimensions. Prekop and Burnett in [42] and Gustavensen in [18], call these dimensions external and internal, and Hofer et al. [19] refer to physical and logical context. The external (physical) dimension refers to context that can be measured by hardware sensors, i.e., location, light, sound, movement, touch, temperature or air pressure, whereas the internal (logical) dimension is mostly specified by the user or captured by monitoring user interactions, i.e., the user's goals, tasks, work context, business processes, the user's emotional state. Most context-aware systems make use of external context factors as they provide useful data, such as location information.

Furthermore, external attributes are easy to sense by using off-the-shelf sensing technologies. Virtually all systems surveyed in [4] deal with physical context information. When dealing with context, three entities can be distinguished [14]: places (rooms, buildings, etc.), people (individuals, groups), and things (physical objects, software services (e.g., Web services)). Each of these entities may be described by various attributes which can be classified into four categories: identity (each entity has a unique identifier), location (entity's position,

co-location, proximity, etc.), status (or activity, meaning the intrinsic properties of an entity, e.g., temperature and lightning for a room, processes running currently on a device, etc.) and time (used for timestamps to accurately define situation, ordering events, etc.). To conclude this brief summary context has been used in many research communities. Its appearance in Web services research is rather recent but has a grounding as can be seen by related work [4] in software engineering.

Making Web services context-aware is not straightforward; many issues are to be addressed (adapted from [44]): how is context structured, how does a Web service bind to context, how are changes detected and assessed for context update purposes, and what is the overload on a Web service of taking context into account? Responses to some of these questions have guided Maamar et al. to organize the context of a Web service along three inter-connected perspectives [32]. The participation perspective is about overseeing the multiple composition scenarios in which a Web service concurrently takes part. The execution perspective is about looking for the computing resources upon which a Web service operates, and monitoring the capabilities of these computing resources so that the Web service's requirements are constantly satisfied. Finally, the preference perspective is about ensuring that user preferences (e.g., execution time at 2 pm) are integrated into the specification of a composite Web service.

2.2. Policies for Web services

Although Web services are a very active area of R&D, to our knowledge, few projects have aimed at integrating policies into Web services. There is also a lack of standardization efforts, which is somehow balanced with two approaches: semantic Web and Boolean combinations of policy predicates.

Kagal et al. claim that policies should be part of the representation of Web services, and in particular of semantic Web services [23]. Policies provide the specification of who can use a service under what conditions, how information should be provided to the service, and how provided information will be used later. In [5], Baresi et al. proposed a policy-based approach to monitor Web services' functional (e.g., constraints on exchanged data) and non-functional (e.g., security, reliability) requirements. In this approach Baresi et al. report on the different types of policies that can be defined along the life cycle of a Web service [40]. These types are service policies, server policies, supported policies, and requested policies.

In [33], Maamar et al. developed consistency, feasibility, and inspection policies to conduct Web services personalization. For illustration purposes, we only overview the consistency policy, which guarantees that a Web service still exactly does what it is supposed to do prior to its personalization. Personalization may alter the initial specification of a Web service when it comes to the list of events that trigger this Web service. Time- and location-related parameters are new events that need to be added to the list of regular events of the Web service.

Desai et al. have recognized the importance of separation of concerns from a software engineering perspective [12]. Indeed, they split a business process into two parts: protocol and policy. The protocol part is a modular, public specification of an interaction among different roles that collaborate towards achieving a desired goal. The policy part is a private description of a participant's business logic that controls how this participant takes part in a protocol.

3. The context-based multi-type policy approach

3.1. Approach presentation

In Fig. 1,[1] we illustrate the context-based multi-type policy approach to support Web services composition. The underlying idea in this approach is that context and policies are combined to handle the specification and execution of a composite Web service. Four levels make up this approach. The component level is concerned with the definition of Web services and their announcements to third parties. These parties correspond to composite Web services that populate the composition level. This level is concerned with the way component Web

[1] The figure reads as follows: bottom-up during normal progress of composition and top-down during exception in composition.

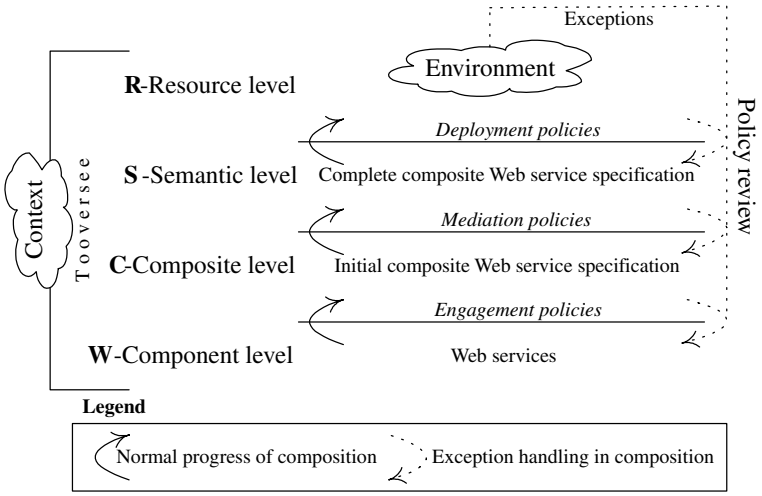


Fig. 1. Representation of the context-based multi-type policy approach.

services are discovered (based on their capabilities) and assembled (according to particular collaboration patterns). However, various obstacles hinder collaboration (for example, the different meanings of the exchanged data between component Web services). Addressing the meaning-disparity obstacle takes to place at the semantic level where agreements between component Web services on the ontologies to comply with and the mapping and conversion mechanisms to perform, are to be reached. Finally, the resource level focusses on the performance of a composite Web service once the semantic issues are resolved between component Web services. This performance is subject to the availability of resources and the commitments of these latter to other composite Web services.

In Fig. 1, policies support transitions between the four levels. Transiting from one level to a next higher-direct-level is subject to policy execution. This execution uses the information that context caters on different elements like the number of Web services that are currently running, the composite Web services that are under preparation, and the ontologies that are selected. Three types of policies are shown in Fig. 1: engagement, mediation, and deployment. The use of these policies allows the separate handling of the specification of composition, the functionalities of Web services, the mediation mechanisms, the execution mechanisms, and the exception handling mechanisms.

While policies support transitions between levels (i.e., how a composite Web service specification progresses), context oversees first, the actions that occur within and across these levels and second, the elements like Web service, composite Web service, ontology, or resource, that are involved by these actions. Prior to triggering any policy and executing any action, the information that context provides is validated. Table 1 summarizes the actions that are performed during level transitions. These actions are categorized per level type and presented as follows:

- At the $\mathscr{W}$-Component level, the actions concern searching for the component Web services through, for example, UDDI facilities (security- and semantic-based search could be considered as reported in [1,41]), and seeking the acceptance of these component Web services to take part in composition. Details on par-

Table 1
Actions, contexts, and policies

Transitions	Actions	Contexts	Policies
$\mathscr{W}$-Component → $\mathscr{C}$-Composite	Search and invite Web services	$\mathscr{W}/\mathscr{C}$-contexts	Engagement
$\mathscr{C}$-Composite → $\mathscr{S}$-Semantic	Connect Web services, and detect and resolve semantic heterogeneities	$\mathscr{C}/\mathscr{S}$-contexts	Mediation
$\mathscr{S}$-Semantic → $\mathscr{R}$-Resource	Search for resources and satisfy execution requirements	$\mathscr{S}/\mathscr{R}$-contexts	Deployment

ticipation acceptance are given in Section 3.2. Roughly, the $\mathscr{W}$-context at the component level tracks the status of each component Web service per composition and the policies this component Web service gets involved in.

- At the $\mathscr{C}$-Composite level, the actions concern connecting the component Web services together according to a specific specification and detecting the semantic heterogeneities that could arise between these component Web services. Roughly, the $\mathscr{C}$-context at the composition level tracks the status of a composite Web service specification in terms of the component Web services that have completed their execution, are under execution, and will be called for execution.
- At the $\mathscr{S}$-Semantic level, the actions concern identifying the type of mediation (semantic, structural, and syntactic as reported in [26]) that needs to be performed between the input and output arguments of the component Web services, and ensuring the success of this mediation. Roughly, the $\mathscr{S}$-context at the semantic level tracks the ontology that annotates these input and output arguments and the mechanisms that are triggered per mediation type. We specialize these mechanisms into mapping functions for input and output arguments and conversion functions for values of input and output arguments.
- At the $\mathscr{R}$-Resource level, the actions concern getting Web services ready for execution after searching for the appropriate resources and satisfying the security requirements of both, Web services, and resources. In [34], we pointed out that neither the Web services should misuse the resources (e.g., resource blocked for excessive periods) nor the resources should alter the integrity of the Web services (e.g., Web service's sensitive information disclosed to non-authorized parties). Roughly, the $\mathscr{R}$-context at the resource level tracks the current status of a resource such as load and the security mechanisms that are triggered for the benefit of Web services and resources.

Deviations from a composite Web service specification like execution delays, throw in exceptions, which leads to reviewing the different actions of the policies that were carried out within and across the multiple levels. The review process occurs in a descending way commencing with resource, semantic, composite, and then, component levels (Fig. 1). Transiting to a lower level means that the direct upper-level is not faulty and extra investigations of the exception are deemed appropriate at this lower level. To keep track of the actions of policies that happened, are happening, and might happen, hence corrective measures (e.g., compensation policies) are scheduled, and undertaken as part of the review process, context monitors the whole composition of Web services.

3.2. Description of $\mathscr{W}/\mathscr{C}/\mathscr{S}/\mathscr{R}$-levels

3.2.1. The $\mathscr{W}$-Component level

It shows the Web services that participate in a composite Web service. This participation is upon approval and happens in accordance with the Web service *instances* principle that was introduced in [34]. This principle emphasizes the simultaneous participation of a Web service in several compositions. A benefit of this principle is the temporal organization of a Web service along three categories (Fig. 2): Web service instances already deployed (past), Web service instances currently deployed (present), and Web service instances to be potentially deployed (future).

Before a Web service accepts the invitation of participation in a composite Web service, it consults its $\mathscr{W}$-context so it can verify the number of current participations *vs.* the maximum number of participations, the expected completion time of its other participations, and the features of the newly received invitation of par-

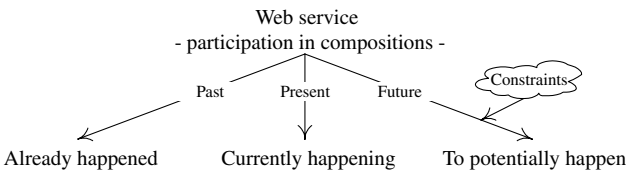


Fig. 2. Organization of a Web service.

ticipation like when the new Web service instance is required. It happens that a Web service refuses an invitation of participation in a composite Web service for various reasons: unavailability due to some expected maintenance work or risk of overload.

3.2.2. The $\mathscr{C}$-Composite level

It is about the specification of a composite Web service in term of business logic. Fig. 3 shows a sample of a specification that uses a combination of service chart diagrams and state chart diagrams [30]. The component Web services of this specification are: trip (`TP`), weather (`WE`), location (`LO`), taxi (`TA`), bus schedule (`BS`), and traffic (`TC`). The respective service chart diagrams of these components are connected through transitions. Some of these transitions have constraints to satisfy like [confirmed(bad weather)]. Other specification languages of composite Web services exist, most promoted BPEL [10]. A composite Web service consults its $\mathscr{C}$-context so it can follow up the execution progress of its specification. This enables, for example, identifying the next Web services to be subject to invitation of participation and the state of each component Web service of the composite Web service (e.g., initiated, running, suspended, resumed).

3.2.3. The $\mathscr{S}$-Semantic level

It is deemed appropriate since the standard Web services protocol-stack (SOAP, WSDL, UDDI) does not meet the requirements of a successful semantic exchange between independent Web services. The semantic level illustrates what we here refer to as *global ontology* that a composite Web service binds to. This ontology tackles the semantic heterogeneities that exist between the component Web services in terms of input and output arguments [8]. In addition, we consider a second type of ontology, which we here refer to as *local ontology*. This latter associates a semantic definition with the respective values of the input and output arguments of Web services [47]. For example, `3(currency=euro)` is a semantic value that defines `Euros` as the currency of the simple value 3. Here, `currency` and `euro` are part of the local ontology. When both types of ontologies are set, this is reported at the $\mathscr{S}$-context level. The objective is to let ontologies (acting like information sources) be aware of the compositions in which they will take part and the mediation mechanisms that will be loaded whether at the composition or component levels.

3.2.4. The $\mathscr{R}$-Resource level

It represents the computing facilities upon which Web services operate. Scheduling the execution requests of Web services is prioritized when sufficient resources are unavailable to satisfy all these requests at once. A Web service requires resources for different operations like self-assessment of current state before accepting/rejecting participations in compositions, satisfying user needs upon request, and data exchange with pre- and post-Web services. Before a resource accepts supporting the execution of an additional Web service, it consults its $\mathscr{R}$-context so it can verify the number of Web services currently executed *vs.* the maximum number of Web services under execution, the approximate completion time of the ongoing executions of Web services, and the features of the newly received request like requested time of execution. Like with Web services, similar considerations apply to resources when it comes to turning down execution requests of Web services.

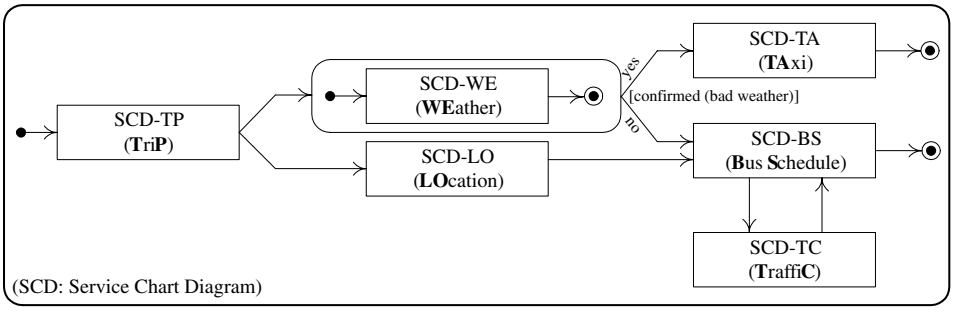


Fig. 3. Sample of a composite Web service specification.

3.3. Description of $\mathscr{W}/\mathscr{C}/\mathscr{S}/\mathscr{R}$-contexts

To comply with the four levels of the multi-type policy approach, we defined four types of context: $\mathscr{W}$-context, $\mathscr{C}$-context, $\mathscr{S}$-context, and $\mathscr{R}$-context (Fig. 1). The specialization of context into sub-contexts is backed by our previous research on Web services personalization [33].

The **$\mathscr{W}$-context** of a Web service returns information on the participations of the Web service in different compositions. These participations are made possible because of the Web services instantiation principle (Fig. 2).

The **$\mathscr{C}$-context** of a composite Web service is built upon the $\mathscr{W}$-contexts of its component Web services and permits overseeing the progress of a composition.

The **$\mathscr{S}$-context** of ontology highlights the composite Web services or component Web services that adopt this ontology first, in the mapping operations between input and output arguments and second, in the conversion operations between the values of input and output arguments.

The **$\mathscr{R}$-context** of a resource oversees the execution of the Web services that operate on top of this resource prior it accepts additional Web services for execution. A resource has computation capabilities that continuously change depending on the number[2] of Web services that are now under execution and the execution duration per Web service.

3.4. Description of policies

Policies permit during a normal progress scenario to move a composition from one level to a higher-direct-level. Fig. 1 illustrates three types of policies: engagement, mediation, and deployment. Policies's outcomes are also subject to review in case of exceptions, i.e., abnormal progress. In the following we explain the intended role of each type of policy.

Engagement policies frame the participation of component Web services in composite Web services. This framing highlights the opportunity of a Web service to take part in several compositions at a time (Fig. 2). Since Web services are context-aware, they assess their participations and use the engagement policies to back their acceptance or denial of participation in additional compositions.

Mediation policies deal with the semantic heterogeneity issues. Because Web services originate from different providers, engineering data exchange is crucial [48]. This engineering is of two types. The first type is about the input and output arguments of Web services that need to be mapped according to what we earlier referred to as global ontology. For instance, `cost` and `price` are semantically similar in an ontology for good description. The second type is about the values of the input and output arguments of Web services. These values need to be converted according to what we earlier referred to as local ontology. For instance, cost should be clearly stated in which currency it is like `the Euro` and how it needs to be converted into other currencies like `the UAE Dirhams`. Mediation policies are in charge of supporting the mappings and conversions between the component Web services, once these latter decide to participate in a composite Web service following the successful triggering of engagement policies.

Deployment policies aim at framing the interactions between component Web services and computing resources. We earlier argued that resources have limited computation capabilities, and scheduling execution requests of component Web services is deemed appropriate. Since resources are context-aware, they assess their commitments and use the deployment policies to back their decisions of accepting or denying a component Web service's execution request.

Reviewing policies happens when exceptions arise during the execution of these policies (Fig. 1). A Web service, which does not get executed because of lack of resources delays the execution of the rest of its composite Web service. A review of the deployment policy of this Web service is required so similar situations can be avoided in the future. Another example concerns Web services, which exchange non-meaningful information due to wrong selection of conversion functions. A review of the mediation policy of both Web services is required. Additional details on the whole exception handling approach is discussed in Section 6.

[2] Number is used for illustration purposes. Additional criteria could be execution load of Web services.

Table 2
Arguments of $\mathscr{W}$-context

Label: corresponds to the identifier of the Web service
Maximum number of participations: corresponds to the maximum number of compositions in which the Web service can participate at a time
Number of active participations: corresponds to the number of active compositions in which the Web service is now participating
Next possibility of participation: indicates when the Web service can participate in a new composition. This is subject to the successful termination of the current active participations
Resource & State per active participation: corresponds to the identifier of the selected resource and the state of the Web service in each active composition. State can be of types in-progress, suspended, aborted, or completed, and will be obtained out of the state argument of $\mathscr{R}$-context of this resource
Ontology of interpretation per active participation: refers to the ontology that the Web service binds to per active participation for interpreting the values of its input and output arguments when conversion functions are triggered
Previous Web services per active participation: indicates the Web services that were successfully completed before the Web service per active composition (null if there are no predecessors)
Current Web services per active participation: indicates the Web services that are concurrently being performed with the Web service per active composition (null if there is no concurrent processing)
Next Web services per active participation: indicates the Web services that will be executed after the Web service successfully completes its execution per active composition (null if there are no successors)
Regular actions: illustrates the actions that the Web service normally performs
Reasons of failure per active participation: informs about the reasons that are behind the failure of the execution of the Web service per active composition
Corrective actions per failure type and per active participation: illustrates the actions that the Web service has performed due to execution failure per active composition
Date: identifies the time of updating the arguments above

4. Structure of contexts

This section describes the internal structure of each type of context namely $\mathscr{W}$-context at the component level, $\mathscr{C}$-context at the composite level, $\mathscr{S}$-context at the semantic level, and $\mathscr{R}$-context at the resource level. By internal structure we mean the list of arguments of context. Currently, context oversees the four layers by returning various details on Web services like execution order, on resources like next period of unavailability, on ontologies like recently used-conversion functions, and on composite Web services like forthcoming executions of Web services. These details affect the process of loading and executing the policies.

$\mathscr{W}$-context encompasses the following arguments (Table 2): label, maximum number of participations, number of active participations, next possibility of participation, resource and state per active participation, local ontology per active participation, previous Web services per active participation, current Web services per active participation, next Web services per active participation, regular actions, reasons of failure per active participation, corrective actions per failure type and per active participation, and date.

$\mathscr{C}/\mathscr{S}/\mathscr{R}$-contexts consist of many arguments as described in Tables 3–5, respectively.

Table 3
Arguments of $\mathscr{C}$-context

Label: corresponds to the identifier of the composite Web service
Global Ontology: refers to the ontology that the composite Web service binds to. All the component Web service have to comply with this binding for their mapping functions of input and output arguments
Previous component Web services: indicates the component Web services of the composite Web service that have successfully been completed with regard to the current component Web services
Current component Web services: indicates the component Web services of the composite Web service that are currently under execution
Next component Web services: indicates the component Web services of the composite Web service that will be called for execution once the current component Web services are successfully completed
Beginning time: informs when the execution of the composite Web service has started
State/Component Web service: corresponds to the state of each component Web service of the composite Web service that is being executed (based on state argument of $\mathscr{W}$-context)
Date: identifies the time of updating the arguments above

Table 4
Arguments of $\mathscr{S}$-context

Label: corresponds to the identifier of the ontology
Current composite Web services: refers to the composite Web services that use the ontology. This is obtained out of the ontology argument of $\mathscr{C}$-context
Current component Web services: refers to the compositions that use the ontology. This is obtained out of the ontology argument of $\mathscr{W}$-context
Triggered mapping functions per current composite Web service: informs about the mapping functions that are used at the composition level
Triggered conversion functions per current component Web service: informs about the conversion functions that are used at the component level
Date: identifies the time of updating the arguments above

Table 5
Arguments of $\mathscr{R}$-context

Label: corresponds to the identifier of the resource
Maximum number of component Web services: corresponds to the maximum number of component Web services that can operate on the resource at a time
Number of active component Web services: corresponds to the number of active component Web services that are now operating on the resource
Next acceptance of component Web services: indicates when the resource can accept new component Web services for execution. This is subject to the successful termination of the current component Web services
Previous component Web services per active composition: indicates the component Web services that were successfully completed before the current Web services that are under execution per active composition
Current component Web services per active composition: indicates the component Web services that are now under execution per active composition
Consumption& State per component Web service and per active composition: corresponds to the resource consumption and state of the component Web service per active composition
Next component Web services per active composition: indicates the component Web services that will be executed after the current component Web services successfully complete their execution per active composition
Date: identifies the time of updating the arguments above

5. Web services and policies

Vidyasankar and Vossen presented a multi-level service composition model [51]. This model perceives service specification as a process that goes through several levels of abstraction: it starts from transactional concepts at the lowest level, and then gradually abstracts into higher levels that are close to service provider or even end-user. In the following we discuss the way the context-based multi-type policy approach we propose impacts the interactions between the components of Fig. 1.

5.1. Impact of policies on Web services

In the approach for Web services composition of Fig. 1, policies have an impact first, on the behavior that Web services show towards composite Web services and second, on the behavior that composite Web services (through their component Web services) show towards computing resources. We identify this behavior with the following attributes: permission, restriction, and obligation.[3] In addition to the behavior dimension in the multi-type policy approach, an extra dimension is considered, which we refer to as reconciliation. This dimension is only dedicated to Web services. It should be noted that providers of Web services are responsible for specifying the multiple policies (Section 5.2).

[3] Marjanovic considers some of these attributes as part of the normative perspective that shows the parties being involved in Web services provisioning [36].

Engagement policies belong to the behavior dimension and apply to Web services. These policies associate three types of behavior with a Web service: permission, restriction, and obligation. Engagement policies satisfy the needs of Web services' providers who have interests and/or obligations in strengthening or restricting the participation of their Web services in some compositions. Additional details on each type of behavior are provided in Section 5.2.

- *Permission*: a Web service is authorized to accept the invitation of participation in a composite Web service once this Web service verifies its current state.
- *Restriction*: a Web service cannot get connected to other peers of a composite Web service because of non-compliance of some peers with this Web service's requirements.
- *Obligation*: it is a strong authorization for a Web service to get connected to other peers of a composite Web service despite negative permission of participation or existence of restriction from participation.

Mediation policies belong to the reconciliation dimension and apply to component Web services of a composite Web service. These policies deal with the semantic issues of the arguments of the component Web services. These arguments are subject to two types of operation: mapping and conversion.

- *Mapping*: is associated with the global ontology that relates the output arguments of a Web service to the input arguments of another Web service. For more details on mapping functions, readers are referred to [24].
- *Conversion*: is associated with the local ontology that relates the values assigned to the output arguments of a Web service to the values to be assigned to the input arguments of another Web service. For example, `5` and `50` are considered equivalent if they are interpreted as follows: `5(currency="euro") is semantically equivalent to 50(currency="yen")` when one euro worths ten yens. For more details on conversion functions, readers are referred to [39].

Deployment policies are part of the behavior dimension and apply to Web services of a composite Web service. These policies associate two types of behavior with a component Web service: permission and restriction. Deployment policies satisfy the needs of resources' owners who have interests and/or obligations in strengthening or restricting the execution of Web services on these resources.

- *Permission*: a Web service receives an authorization of execution from a resource once this resource checks its current state.
- *Restriction*: a Web service is not accepted for execution on a resource because of non-compliance of this Web service with the resource's requirements.

5.2. Specification of policies

In what follows, we specify the policies that define the behavior of Web services and thereby address the semantic heterogeneity between Web services. To this end, a policy definition language is required. The selection of this language is guided by some requirements that need to be satisfied as reported in [49]: *expressiveness* to support the wide range of policy requirements arising in the system being managed, *simplicity* to ease the policy definition tasks for people with various levels of expertise, *enforceability* to ensure a mapping of policy specification into concrete policies for various platforms, *scalability* to guarantee adequate performance, and *analyzability* to allow reasoning about and over policies. Additional requirements for policy definition languages are also given in [6].

In this paper, we use the Web Services Policy Language (WSPL) based on the arguments that Anderson puts forward [3]. She argues that a Web service has various aspects and features that can be controlled or described using policy rules [2]. Some of these aspects include authentication, quality-of-service, privacy, and reliable messaging. Anderson suggests WSPL to express policies and to achieve the interoperability of

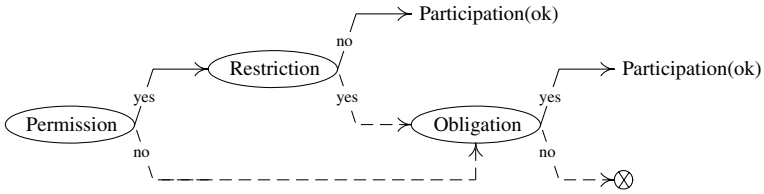


Fig. 4. Behaviors of a Web service in an engagement scenario.

Web services. The syntax of WSPL is strictly based on the OASIS eXtensible Access Control Markup Language (XACML) standard.[4]

Engagement policy

Fig. 4 illustrates the way the different behaviors of a Web service are connected together based on the execution outcome of the engagement policy per type of behavior. For instance, a positive permission (i.e., yes) is not confirmed until no restrictions are identified (i.e., no). More details about this figure are given in the description of each type of behavior. The dashed lines in Fig. 4 represent potential cases of exceptions to be discussed in Section 6.

1. An engagement policy for permission consists of authorizing a Web service to be part of a composite Web service. The authorization is based on the state of the Web service that is known using its respective $\mathscr{W}$-context (Table 2). The following illustrates an engagement policy for permission in WSPL. It states that a Web service participates in a composition subject to evaluating <Condition> to true. This latter refers to some arguments like the number of current active participations of the Web service in compositions and the next possibility of participation of the Web service in additional compositions. These arguments are known as vocabulary items in WSPL. In the policy, <TrueConclusion> shows the permission of participation, whereas <FalseConclusion> shows the contrary. In case of positive permission, yes-permission-participation procedure is executed, which results in updating the following arguments of $\mathscr{W}$-context of the Web service (Table 2): *local ontology*, *number of active participations*, *previous Web services*, *current Web services*, and *next Web services*.

```
Policy(Aspect="PermissionParticipation"){
<Rule xmlns="urn:oasis:names:tc:xacml:3.0:permission:policy:schema:wd:01"
xmlns:proc="permission-participation" RuleId="PermissionParticipationWS">
  <Condition>
   <Apply FunctionId="and">
   <Apply FunctionId="integer-less-than" DataType="boolean">
   <SubjectAttributeDesignator AttributeId="CurrentNumberOfParticipations"
   DataType="integer"/>
   <SubjectAttributeDesignator AttributeId="MaximumNumberOfParticipations"
   DataType="integer"/>
   </Apply>
    <SubjectAttributeDesignator AttributeId="Availability" DataType="bool-
  ean"/>
   </Apply>
  </Condition>
  <Conclusions>
   <TrueConclusion>
   <proc:do> yes-permission-participation </proc:do>
```

[4] http://www.oasis-open.org/committees/download. php/2406/oasis-xacml-1.0.pdf.

```
    </TrueConclusion>
    <FalseConclusion>
    <proc:do> no-permission-participation </proc:do>
    </FalseConclusion>
   </Conclusions>
</Rule>}
```

For the sake of simplicity, we do not list in the aforementioned policy the full URNs for the XACML DataTypes and FunctionIds, and use rather simple names for the vocabulary items (e.g., Current-NumberOfParticipations) and simple names for function identifiers (e.g., integer-less-than). The vocabulary items could also be elements in a schema instance that is passed to the policy evaluator, in which case "AttributeSelector" with an XPath expression would be used rather than <...AttributeDesignator>.

2. An engagement policy for restriction consists of preventing a Web service from taking part in a composite Web service. A restriction does not implement a negative permission of participation, rather it follows a positive permission of participation (Fig. 4). Restrictions could be geared towards the quality issue (e.g., time of response, reputation, performance, throughput) of the component Web services with whom a Web service will interact. For example a Web service's owner has only interest in the Web services that have a "good" QoS record [37]. The following illustrates a restriction policy in WSPL. It states that a Web service can be restricted from participation subject to evaluating <Condition> to true. This latter checks that a positive permission of participation exists and the assessment level of the QoS of the Web services is low. These Web services are identified using *previous Web services per active participation* argument of $\mathscr{W}$-context of the Web service (Table 2). In this policy, QoSAssessment is an integer value that is the result of evaluating the QoS of a Web service, and QoSThreshold is the minimum QoS assessment value that is acceptable for composition.

```
Policy (Aspect="RestrictionPermission"){
<Rule xmlns="urn:oasis:names:tc:xacml:3.0:generalization:policy:schema:wd:01"
RuleId="RestrictionParticipationWS">
<Condition>
 <Apply FunctionId="and">
 <SubjectAttributeDesignator AttributeId="YesPermissionParticipation">
 DataType="boolean"/>
 <Apply FunctionId="integer-great-than-or-equal">
  <SubjectAttributeDesignator AttributeId="QoSAssessment" DataType="inte-
ger"/>
   <SubjectAttributeDesignator AttributeId="QoSThreshold" DataType="inte-
ger"/>
 </Apply>
 </Apply>
</Condition>
<Conclusions>
 <TrueConclusion RestrictionParticipation = "No"/>
 <FalseConclusion RestrictionParticipation = "Yes"/>
</Conclusions>
</Rule>}
```

3. Engagement policy for obligation guarantees that a Web service will definitely be part of a composite Web service, despite negative permission or existence of restrictions (Fig. 4). Obligations are aimed for the situations that call for an immediate handling of users' needs. In addition obligations result sometimes in dropping or suspending some of the ongoing compositions due to the maximum number of participations of a Web service in compositions at a time (Table 2). Because of this maximum number of participation, a

Web service might have to suspend or drop some of its ongoing participations due to the obligation of joining another composition. Compensation solutions are to be developed for the compositions that are affected by suspension or dropping decisions of participation. For example, a substitute Web service needs to be found if a Web service decides to cancel its participation in an ongoing participation. Furthermore, a composition needs to be resumed according to a certain time frame if a Web service temporarily suspends its participation.

The following illustrates an engagement policy for obligation in WSPL. It states that a Web service takes part in a composite Web service subject to evaluating <Condition> to true. This latter checks that either a no-permission of participation exists or a confirmation of restrictions exists. Once the participation is confirmed, obligation turns out to be like permission with regard to the actions to carry out.

```
Policy (Aspect="ObligationParticipation"){
<Rule xmlns="urn:oasis:names:tc:xacml:3.0:obligation:policy:schema:wd:01"
xmlns:proc="obligation-participation" RuleId="ObligationParticipationWS">
  <Condition>
   <Apply FunctionId="and">
   <Apply FunctionId="xor">
   <SubjectAttributeDesignator AttributeId="NoPermissionParticipation"
   DataType="boolean"/>
   <SubjectAttributeDesignator AttributeId="YesRestrictionParticipation"
   DataType="boolean"/>
   <Apply FunctionId="equal">
  <SubjectAttributeDesignator AttributeId="UserNeedLevel" DataType=
  "string"/>
   <AttributeValue DataType="integer"/> "high" </AttributeValue>
   </Apply>
   </Apply>
   </Apply>
  </Condition>
  <Conclusions>
   <TrueConclusion>
   <proc:do> yes-obligation-participation </proc:do>
   </TrueConclusion>
   <FalseConclusion>
   <proc:do> no-obligation-participation </proc:do>
   </FalseConclusion>
  </Conclusions>
</Rule>}
```

5.2.1. Mediation policy

Fig. 5 illustrates the way the semantic reconciliation between component Web services progresses. This progress is based on the execution outcome of the mediation policy per type of operation known as mapping and

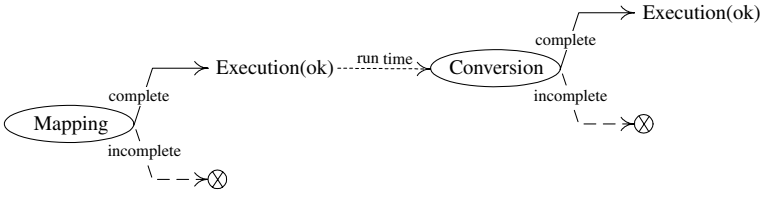


Fig. 5. Operations of a Web service in a mediation scenario.

conversion, respectively. For instance, mapping between component Web services' arguments needs to be successfully completed prior to converting values of component Web services' arguments. More details about this figure are given in the description of each type of operation. The dashed lines in Fig. 5 represent potential cases of exceptions and are discussed in Section 6.

1. A mediation policy for mapping is about the operations that permit establishing links between input and output arguments of the component Web services of a composite Web service. A link happens according to the global ontology that the composite Web service binds to. The global ontology is reported in $\mathscr{C}$-context of the composite Web service. The following illustrates a mediation policy for mapping in WSPL. It states that the mapping between arguments takes place subject to evaluating <Condition> to true. This latter checks the existence of previous Web services to the current Web service. These Web services are identified using *previous Web services per active participation* argument of $\mathscr{W}$-context of the Web service (Table 2). In case of positive mapping, `yes-mapping` procedure is executed, which results in mapping the output arguments of the previous Web services onto the input arguments of the current Web service.

```
Policy (Aspect="Mapping"){
<Rule xmlns="urn:oasis:names:tc:xacml:3.0:mapping:policy:schema:wd:01"
RuleId="MappingWS">
  <Condition>
   </Apply>
  <SubjectAttributeDesignator AttributeId="PreviousWebServicesPerActive-
  Participation"
   DataType="boolean"/>
   </Apply>
  </Condition>
  <Conclusions>
   <TrueConclusion>
   <proc:do> yes-mapping </proc:do>
   </TrueConclusion>
   <FalseConclusion>
   <proc:do> no-mapping </proc:do>
   </FalseConclusion>
  </Conclusions>
</Rule>}
```

2. A mediation policy for conversion is about the operations that manage values of input and output arguments of the component Web services of a composite Web service. This happens according to the local ontology that the component services bind to. The local ontology is reported in $\mathscr{W}$-context of a component Web service. The following illustrates a mediation policy for conversion in WSPL. The policy states that the conversion between arguments takes place subject to evaluating <Condition> to true. This latter checks the successful mapping between all the Web services' arguments. In case of positive conversion, `yes-conversion` procedure is executed, which results in converting the values of the output arguments of the previous Web services into values required by the input arguments of the current Web service.

```
Policy (Aspect="Conversion"){
<Rule xmlns="urn:oasis:names:tc:xacml:3.0:permission:policy:schema:wd:01"
xmlns:proc="conversion" RuleId="ConversionWS">
 <Condition>
  </Apply>
 <SubjectAttributeDesignator AttributeId="YesMapping" DataType="boolean"/>
  </Apply>
 </Condition>
```

```
 <Conclusions>
  <TrueConclusion>
  <proc:do> yes-conversion </proc:do>
  </TrueConclusion>
  <FalseConclusion>
  <proc:do> no-conversion </proc:do>
  </FalseConclusion>
 </Conclusions>
</Rule>}
```

5.2.2. Deployment policy

Fig. 6 illustrates the way the different behaviors of a Web service get connected together based on the execution outcome of the deployment policy per type of behavior. For instance a positive permission for execution (i.e., yes) is confirmed if there are no-restrictions (i.e., no) that could deny this permission at run time. More details about this figure are given in the description of each type of behavior. The dashed lines in Fig. 6 represent potential cases of exceptions and are discussed in Section 6.

1. A deployment policy for permission is about a Web service that receives the necessary authorizations from a resource to be executed over it. The authorizations are based on the state of the resource, which is reflected using its respective $\mathscr{R}$-context (Table 5). The following illustrates a deployment policy for permission in WSPL. It states that a resource accepts the execution request of a Web service subject to evaluating `<Condition>` to true. This latter refers to some arguments like the number of active component Web services that the resource supports their execution and the next acceptance of the resource for additional component Web services. In the policy, `<TrueConclusion>` shows the permission of execution, whereas `<FalseConclusion>` shows the contrary. In case of positive permission of execution, `yes-permission-deployment` procedure is executed, which results in updating the following arguments: *resource* & *state per active participation* of $\mathscr{W}$-context of the Web service (Table 2) and *number of active component Web services* of $\mathscr{R}$-context of the resource (Table 5).

```
Policy (Aspect="PermissionDeployment"){
<Rule xmlns="urn:oasis:names:tc:xacml:3.0:permission:policy:schema:wd:01"
xmlns:proc="permission-deployment" RuleId="PermissionDeploymentWS">
  <Condition>
   <Apply FunctionId="and">
   <Apply FunctionId="integer-less-than" DataType="boolean">
  <SubjectAttributeDesignator AttributeId="NumberofActiveComponent-
  WebServices"
   DataType="integer"/>
  <SubjectAttributeDesignator AttributeId="MaximumNumberofComponent-
  WebServices"
   DataType="integer"/>
   </Apply>
```

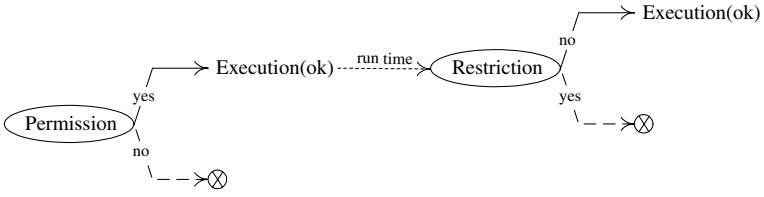


Fig. 6. Behaviors of a Web service in a deployment scenario.

```
<SubjectAttributeDesignator AttributeId="NextAcceptanceofComponent-
WebServices"
 DataType="boolean"/>
 </Apply>
</Condition>
<Conclusions>
 <TrueConclusion>
 <proc:do> yes-permission-deployment </proc:do>
 </TrueConclusion>
 <FalseConclusion>
 <proc:do> no-permission-deployment </proc:do>
 </FalseConclusion>
</Conclusions>
</Rule>}
```

2. A deployment policy for restriction consists of preventing a Web service form being engaged in an execution over a resource. A restriction does not implement a negative permission of deployment, rather it follows a positive permission of deployment (Fig. 4). Besides the example of resource failure, restrictions could be geared towards the reinforcement of the execution features that were agreed on between a Web service and a resource. For example a Web service binds to a resource for execution prior to the scheduled time. The following illustrates a deployment policy for restriction in WSPL. It states that a Web service can be restricted from execution subject to evaluating <Condition> to true. This latter checks that a positive permission of execution has been issued and the agreed execution time is valid. The execution time of a Web service is identified using *next component Web services per active participation* argument of $\mathscr{R}$-context of the resource (Table 5).

```
Policy (Aspect="RestrictionDeployment"){
<Rule xmlns="urn:oasis:names:tc:xacml:3.0:generalization:policy:schema:wd:01"
RuleId="RestrictionDeploymentWS">
  <Condition>
   <Apply FunctionId="and">
   <SubjectAttributeDesignator AttributeId="YesPermissionDeployment"
   DataType="boolean"/>
   <Apply FunctionId="equal" DataType="boolean">
  <SubjectAttributeDesignator AttributeId="ExecutionTime" DataType=
  "String"/>
   </Apply>
   </Apply>
  </Condition>
  <Conclusions>
   <TrueConclusion RestrictionDeployment = "No"/>
   <FalseConclusion RestrictionDeployment = "Yes"/>
  </Conclusions>
</Rule>}
```

6. Exception handling

6.1. Rationale and motives

Exception handling needs to be a normal practice in any system development. Yu et al. adopt the example of a programmer who writes BPEL business processes [52]. The complexity of some domains results sometimes in processes with errors. When a process is put into operation, exception handling has to occur otherwise, angry

customers and loss of revenues will arise. Another motive to investigate exception handling in the policy approach of Fig. 1 is the risk of policy alteration. We see policies as resources that can be subject to attacks like any other resource exposed to external parties [35]. For example, a Web service gets engaged in a composition despite existence of restrictions. The relevant policy did not get triggered due to unpermitted changes in its content.

In our policy approach, exception handling means reviewing policies' outcomes and taking appropriate actions in a top-down way of consulting the four levels shown in Fig. 1.

An exception could take place at one of the following levels:

- *Resource level*: there is no guarantee that a particular resource is up at time of a Web service execution. A resource could be down due to power failure.
- *Semantic level*: there is no guarantee that appropriate mapping or conversion functions are identified during semantic mediation. The complexity of some domains may hinder the availability and even preclude the existence of such functions.
- *Composite level*: there is no guarantee that the specification of a composite Web service is free-of-errors (unless the specification is of type formal). Conflicting actions like concurrent accept and reject and deadlocks may occur during this specification execution.
- *Component level*: there is no guarantee that a particular Web service is still available at time of request. A provider could withdraw its Web service from a composition without prior notice.

6.2. Exceptions per policy type

Deployment policy. Permission and restriction behaviors are related to a Web service following the triggering of a deployment policy. Potential origins of an exception are as follows. First, an execution permission for a Web service was not issued but there was a tentative of execution on a resource. Second, an execution restriction for a Web service was detected but this was not properly reported.

Exception in case of permission could happen because of the inaccurate assessment of some arguments in a resource's $\mathscr{R}$-context:

- Number of active component Web services argument: the resource did not exactly assess the number of Web services that are currently running on top of it. This number should not exceed the maximum number of authorized Web services for execution.
- Next acceptance of component Web services argument: the execution of a Web service got delayed, which has negatively affected the execution scheduling of the forthcoming Web services on the resource.

Exception in case of restriction could happen because of the inaccurate assessment of some arguments in a resource's $\mathscr{R}$-context like next component Web services per active participation argument. The execution of a Web service should not occur prior to the time that was agreed upon between the resource and the Web service.

Mediation policy. Mapping and conversion operations involve a Web service during the triggering of a mediation policy. Potential origins of an exception are as follows. First, a relevant mapping function to match inputs and outputs of component Web services was not found, but there was a tentative of performing the matching. Second, the execution of a conversion function of inputs' and outputs' values did not succeed but this was not properly reported.

Exception in case of mapping could happen because of the inaccurate assessment of various arguments related to the global ontology and composite Web service:

- Current composite Web services argument of an ontology's $\mathscr{S}$-context: it may happen that a composite Web service was not reported at the level of this argument. Thus the composite Web service is not authorized to use the mapping functions of the global ontology.
- Global ontology argument of a composite Web service's $\mathscr{C}$-context: it may happen that some component Web services of the composite Web service did not bind to the global ontology due to unpredicted failure.

Exception in case of conversion could happen because of the inaccurate assessment of various arguments related to the local ontology and component Web services:

- Current component Web services argument of an ontology's $\mathscr{S}$-context: it may happen that a component Web service was not reported at the level of this argument. Thus, the component Web service is not authorized to use the conversion functions of the local ontology.
- Local ontology per active composition argument of a component Web service's $\mathscr{W}$-context: it may happen that a Web service did not bind to the local ontology due to unpredicted failure.

Engagement policy. Permission, restriction, and obligation behaviors are related to a Web service following the triggering of an engagement policy. The dashed lines in Fig. 4 represent the potential origins of an exception. First, a participation permission for a Web service was not issued, but there was a tentative of participation. Second, a participation restriction for a Web service was detected, but it has not been taken effectively. Finally, a participation obligation for a Web service was issued, but this participation did not happen as expected.

Exception in case of permission could happen because of the inaccurate assessment of some arguments in a Web service's $\mathscr{W}$-context:

- Number of active participations argument: the Web service did not exactly assess its current participations in compositions. This number should not exceed the maximum number of authorized compositions. In the engagement policy for permission given in Section 5.2, the condition of participation is shown with the following statements.

```
<Apply FunctionId="integer-less-than"...>
<SubjectAttributeDesignatorAttributeId="CurrentNumberOfParticipations"
.../>
<SubjectAttributeDesignator   AttributeId="MaximumNumberOfParticipations"
.../>
</Apply>
```

  Since there is no permission of participation the invocation of the Web service fails due to unavailability. A policy needs to be triggered so actions are taken such as reviewing the current number of active participations of the Web service and notifying the relevant composite Web service about this Web service disengagement.
- Next possibility of participation argument: the execution of a Web service got delayed, which has negatively affected the participation scheduling of the Web service in forthcoming compositions.

Exception in case of restriction could happen because of the inaccurate assessment of some arguments of a Web service's $\mathscr{W}$-context like next Web services per active participation argument. A Web service cannot connect other peers if they are not listed in this argument.

Exception in case of obligation could happen because of the inaccurate assessment of some arguments of a Web service's $\mathscr{W}$-context like maximum number of participations. A Web service cannot exceed its maximum number of participation when connecting other peers. In case this number is reached, a Web service has to suspend or drop some of its ongoing participations because of the obligation of joining another composition. Compensation solutions are to be developed for the compositions that got affected by suspension or dropping decisions of participation. For example, a substitute Web service needs to be found if a Web service decides to cancel its participation in an ongoing participation. Furthermore, a composition needs to be resumed according to a certain time frame if a Web service temporarily suspends its participation.

7. Implementation

Fig. 7 illustrates the general architecture of the prototype for implementing the proposed context-based multi-type policy approach for Web services composition. The following summarizes the list of programming

tools and languages: Eclipse to build the prototype, XACML to specify policies, XML to specify contexts of Web services, Jena to access and manipulate context information, w3c.dom and javax.xml parsers to access and manipulate XML-based policy information, Java SWT (Standard Widget Toolkit) to create GUIs, finally, core java to integrate all the programming APIs together.

The prototype architecture consists of five repositories and three managers. The five repositories are dedicated to Web services, composite Web services, ontologies, resources, and finally policies. The three managers are labelled as context, composition, and policy. The composition manager interacts with users and let them select the specifications of composite Web services according to their needs. The context manager oversees the content of $\mathscr{W}/\mathscr{C}/\mathscr{S}/\mathscr{R}$-contexts. The policy manager is in charge of triggering the various policies in terms of engagement, mediation, and deployment.

During composite Web service execution, the three managers engage in interactions. First, the composition manager requires contextual information from the context manager for instance about the state of the component Web services or state of resources. Second, the composition manager completes its specification prior to execution based on the feedback it receives from the policy manager. This one triggers policies using the various contextual details it, also, receives from the context manager.

Fig. 8 shows various snapshots of the prototype. It shows the initial values of contexts and policies for `Weather WS` when the prototype is launched. For illustration purposes we assumed that `Weather WS` provides at most two Web service instances at any point of time. Fig. 8(a) and (b) shows $\mathscr{W}$-context's and $\mathscr{R}$-context's arguments of `Weather WS` and associated resource. In the same figure, (c) , (d), and (e) show different policies associated with `Weather WS`. Fig. 9 also illustrates the engagement policy for restriction of `Weather`

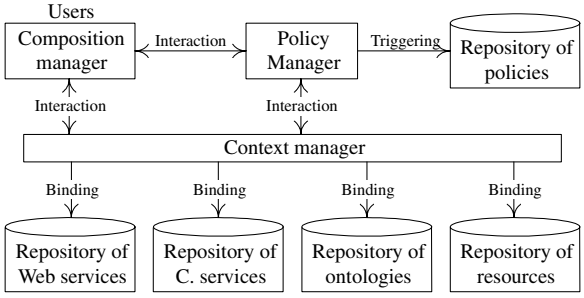


Fig. 7. Architecture of the prototype.

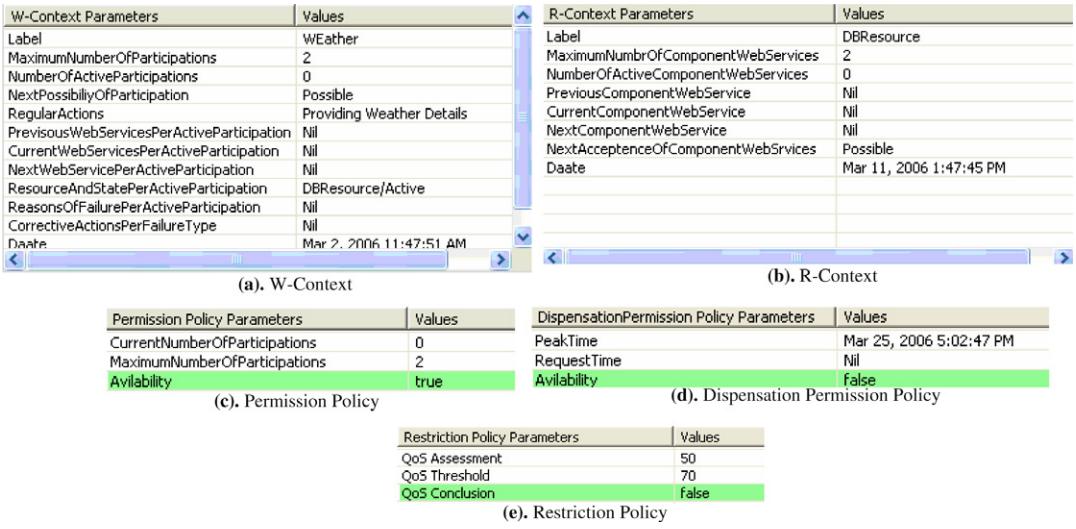


Fig. 8. Snapshots from the prototype: Initial values of contexts and policies.

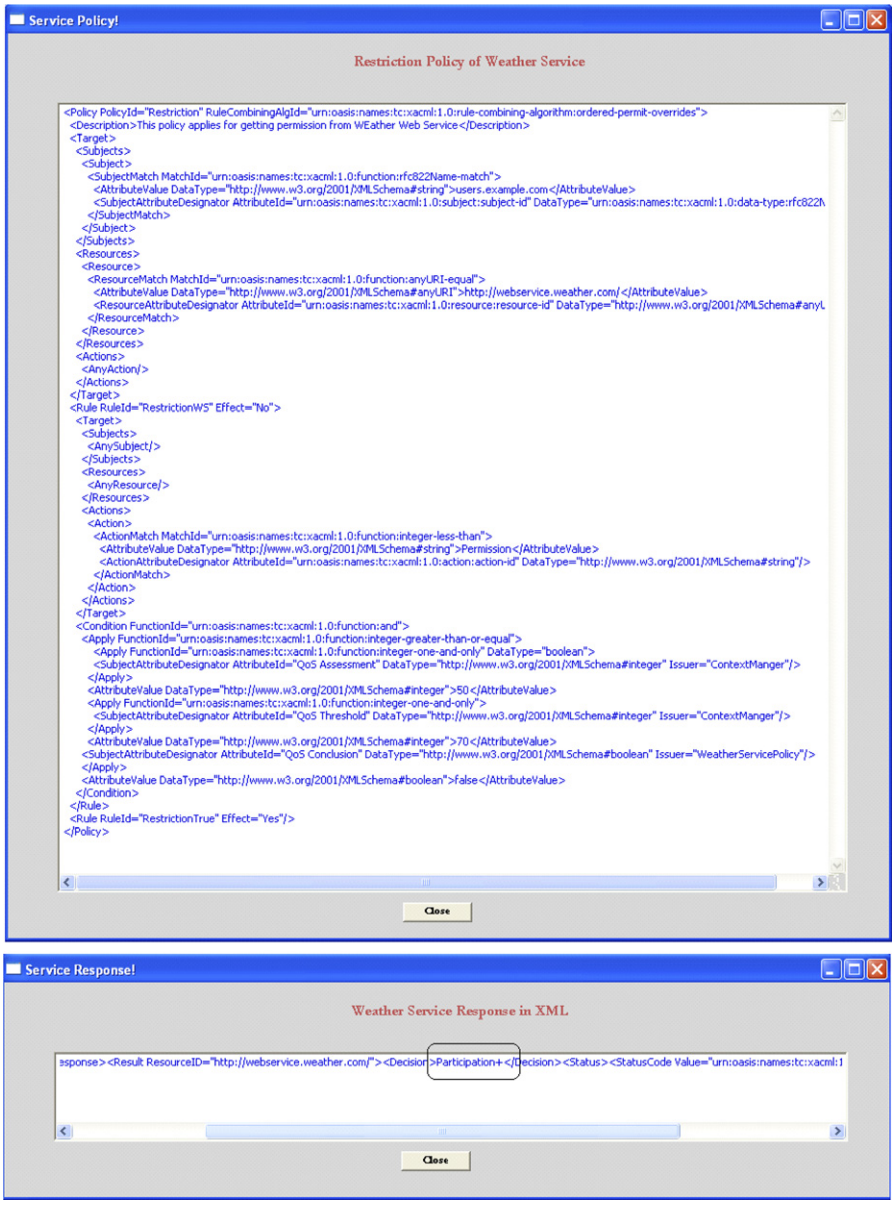


Fig. 9. Snapshots from the prototype: engagement policy for restriction for Weather WS and its response.

WS and second, the response of this Web service when it gets requested for participation (the response is based on the content of contexts and active policies).

8. Conclusion

In this paper we proposed a context-based multi-type policy approach for Web services composition. The role of policies and context in this composition has been depicted as follows.

- Policies manage the transitions between the component, composite, semantic, and resource levels. Transiting from one level to another requires activating policies. Three types of policies were developed: engagement policies assess a Web service participation in a given composition, mediation policies deal with the

semantic heterogeneity by tacking into account the mappings and conversions between a set of component Web services, and deployment policies manage the interactions between component Web services and computing resources.

- Context provides useful information concerning the environment wherein the composition of Web services occurs. Context caters the necessary information that enables tracking the whole composition process by enabling for instance to trigger the appropriate policies and to regulate the interactions between Web services according to the current state of the environment. Four types of context were defined, namely $\mathscr{W}/\mathscr{C}/\mathscr{S}/\mathscr{R}$-context standing for context of Web service, context of composite Web service, context of ontology, and context of resource.

Various reasons motivated our adoption of policies for Web services. For instance, we considered policies as a means for separating the guidelines that manage Web services composition from the guidelines that manage their semantic mediation and deployment. Indeed policies can be used in various scenarios ranging from announcement of Web services to compatibility between Web services. We also argued that Web services need to be aware of the environment in which they operate. As a result, they can consider their internal execution state, can question if they would be rewarded for their participation in composition, can manage their performance in case it gets disrupted, can secure their interactions with peers, and can adapt their behavior in case of failure.

Although the integration of context into a Web services composition approach has various benefits [32], there are situations that call for more information besides context. These situations identify the *regulations* that businesses have the right to impose on their assets like Web services and resources. Our future work consists of looking into the ways of dealing with regulations in Web services composition. Our initial analysis of regulations is motivated by the work of Giblin et al. [17].

References

[1] S. Agarwal, B. Sprick, Specification of access control and certification policies for semantic Web services, in: Proceedings of The 6th International Conference on Electronic Commerce and Web Technologies (EC-Web'2005), Copenhagen, Denmark, 2005.
[2] A.H. Anderson, An introduction to the Web services policy language (SWPL), in: Proceedings of The 5th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'2004), New-York, USA, 2004.
[3] A.H. Anderson, Predicates for boolean Web service policy language (SWPL), in: Proceedings of the International Workshop on Policy Management for The Web (PM4W'2005) held in conjunction with The Fourthen International World Wide Web Conference (WWW'2005), Chiba, Japan, 2005.
[4] M. Baldauf, S. Dustdar, F. Rosenberg, A survey on context aware systems, International Journal of Ad Hoc and Ubiquitous Computing, forthcoming.
[5] L. Baresi, S. Guinea, P. Plebani, WS-policy for service monitoring, in: Proceedings of the 6th VLDB Workshop on Technologies for E-Services (TES'2005) held in conjunction with the 31st International Conference on Very Large Data Bases (VLDB'2005), Trondheim, Norway, 2005.
[6] T. Breaux, J. Niehaus, Requirements for a policy-enforceable agent architecture, Technical Report, TR-2005-17, North Carolina State University, Department of Computer Science, Raleigh, North Carolina, USA, 2005.
[7] I. Budak Arpinar, B. Aleman-Meza, R. Zhang, A. Maduko, Ontology-driven Web services composition platform, in: Proceedings of The IEEE International Conference on E-Commerce Technology (CEC'2004), San-Diego, USA, 2004.
[8] C. Bussler, D. Fensel, A. Maedche, A conceptual architecture for semantic Web enabled Web services, SIGMOD Record 31 (4) (2002).
[9] J. Coutaz, J.L. Crowley, S. Dobson, D. Garlan, Context is key, Communications of the ACM 48 (3) (2005).
[10] F. Curbera, R. Khalaf, N. Mukhi, S. Tai, S. Weerawarana, The next step in Web services, Communications of the ACM 46 (10) (2003).
[11] N. Damianou, N. Dulay, E. Lupu, M. Sloman, The Ponder policy specification language, in: Proceedings of the 2nd IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'2001), Bristol, UK, 2001.
[12] N. Desai, A.U. Mallya, A.K. Chopra, M.P. Singh, Processes = Protocols + Policies: a methodology for business process development, in: Proceedings of the 14th International World Wide Web Conference (WWW'2005), Chiba, Japan, 2005.
[13] A.K. Dey, G.D. Abowd, Towards a better understanding of context and context-awareness, in: Proceedings of the Workshop on the What, Who, Where, When, and How of Context-Awareness held in conjunction with CHI'2000, The Hague, The Netherlands, 2000.
[14] A.K. Dey, G.D. Abowd, D. Salber, A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications, Human–Computer Interaction Journal, Special Issue on Context-Aware Computing 16 (1) (2001).
[15] Anind K. Dey, Context-aware computing: the CyberDesk Project, in: Proceedings of the AAAI'98 Spring Symposium on Intelligent Environments (AAAI'1998), 1998.

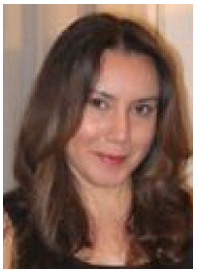
[16] S. Dustdar, W. A Schreiner, Survey on Web services composition, International Journal of Web and Grid Services 1 (1) (2005).
[17] C. Giblin, A.Y. Liu, S. Mueller, B. Pfitzmann, X. Zhou, Regulations expressed as logical models (REALM). Technical Report, RZ 3616, IBM Research Division, Zurich, 2005.
[18] R.M. Gustavsen, Condor – an application framework for mobility-based context-aware applications, in: Proceedings of the Workshop on Concepts and Models for Ubiquitous Computing held in conjunction with the 4th International Conference on Ubiquitous Computing (UBICOMP'2002), Göeteborg, Sweden, 2002.
[19] T. Hofer, W. Schwinger, M. Pichler, G. Leonhartsberger, J. Altmann, Context-awareness on mobile devices – the hydrogen approach, in: Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'2002), Big Island, Hawaii, USA, 2002.
[20] I. Horrocks, P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean, SWRL A Semantic Web Rule Language Combining OWL and RuleML. W3C Submission, 21 May 2004, http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/.
[21] R. Hull, P. Neaves, J. Bedford-Roberts, Towards situated computing, in: Proceedings of the First International Symposium on Wearable Computers, Cambridge, Massachusetts USA, 1997.
[22] L. Kagal, T. Finin, A. Joshi, A policy language for a pervasive computing environment, in: Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'2003) in conjunction with the 8th ACM Symposium on Access Control Models and Technologies (SACMAT'2003), Lake Como, Italy, 2003.
[23] L. Kagal, M. Paolucci, N. Srinivasan, G. Denker, K. Finin, T. Sycara, Authorization and privacy for semantic Web services, IEEE Intelligent Systems 19 (4) (2004).
[24] Y. Kalfoglou, M. Schorlemmer, Ontology mapping: the state of the art, The Knowledge Engineering Review 18 (1) (2003).
[25] M. Keidl, A. Kemper, A framework for context-aware adaptable Web services, in: Proceedings of the 9th International Conference on Extending Database Technology (EDBT'2004), Heraklion, Crete, 2004.
[26] E. Leclercq, D. Benslimane, K. Yetongnon, ISIS: a semantic mediation model and an agent based architecture for GIS interoperability, in: Proceedings of the International Database Engineering and Applications Symposium (IDEAS'1999), Montreal, Canada, 1999.
[27] F. Leymann, Web Services Flow Language (WSFL 1.0), Technical report, IBM Corporation, 2001.
[28] E. Lupu, M. Sloman, Conflicts in policy-based distributed systems management, IEEE Transactions on Software Engineering 25 (6) (1999).
[29] K.J. Ma, Web services: what's real and what's not, IEEE IT Professional 7 (2) (2005).
[30] Z. Maamar, B. Benatallah, W. Mansoor, Service chart diagrams – description & application, in: Proceedings of the Alternate Tracks of The 12th International World Wide Web Conference (WWW'2003), Budapest, Hungary, 2003.
[31] Z. Maamar, D. Benslimane, A. Anderson, Using policies to manage composite Web services, IEEE IT Professional, 2006 (forthcoming).
[32] Z. Maamar, D. Benslimane, N.C. Narendra, What can context do for Web services? Communications of the ACM, forthcoming.
[33] Z. Maamar, S. Kouadri Mostéfaoui, Q.H. Mahmoud, On personalizing Web services using context, International Journal of E-Business Research, Special Issue on E-Services, The Idea Group Inc. 1 (3) (2005).
[34] Z. Maamar, S. Kouadri Mostéfaoui, H. Yahyaoui, Towards an agent-based and context-oriented approach for Web services composition, IEEE Transactions on Knowledge and Data Engineering 17 (5) (2005).
[35] Z. Maamar, G. Kouadri MostTfaoui, D. Benslimane, Policy-based approach to secure context in a Web services environment, in: Proceedings of the 8th International Conference on Enterprise Information Systems (ICEIS'2006), Paphos, Cyprus, 2006.
[36] O. Marjanovic, Managing the normative context of composite E-services, in: Proceedings of the International Conference on Web Services (ICWS-Europe'2003), Erfurt, Germany, 2003.
[37] D.A. Menascé, QoS issues in Web services, IEEE Internet Computing 6 (6) (2002).
[38] T. Moses, XACML profile for Web services. Working Draft 04, 29 September 2003, http://www.oasis-open.org/committees/download.php/3661/draft-xacml-wspl- 04.p.
[39] M. Mrissa, D. Benslimane, Z. Maamar, C. Ghedira, Towards a semantic- and context-based approach for composing Web services, International Journal of Web and Grid Services 1 (3/4) (2005).
[40] Mukhi N, P. Plebani, Supporting policy-driven behaviors in Web services: experiences and issues, in: Proceedings of the Second International Conference on Service-Oriented Computing (ICSOC'2004), New York City, NY, USA, 2004.
[41] M. Paolucci, T. Kawamura, T.R. Payne, K. Sycara, Importing the Semantic Web in UDDI, in: Proceedings of the Workshop on Web Services, E-business, and Semantic Web (WES'2002) held in conjunction with he 14th International Conference on Advanced Information Systems (CAiSE'2002), Toronto, Canada, 2002.
[42] P. Prekop, M. Burnett, Activities, context, and ubiquitous computing, Computer Communications. Special Issue on Ubiquitous Compuitng 26 (11) (2003).
[43] Nick Ryan, Jason Pascoe, David Morse, Enhanced reality fieldwork: the context-aware archaeological assistant, Computer Applications in Archaeology (1997).
[44] M. Satyanarayanan, Pervasive computing: vision and challenges, IEEE Personal Communications 8 (4) (2001).
[45] B. Schilit, M. Theimer, Disseminating active map information to mobile hosts, IEEE Network 8 (5) (1994).
[46] J. Schlimmer, Web Services Policy Framework (WS-Policy), 2004. http://www-128.ibm.com/developerworks/webservices/library/specification/ws-p.
[47] E. Sciore, M. Siegel, A. Rosenthal, Using semantic values to facilitate interoperability among heterogeneous information systems, ACM Transactions of Database Systems 19 (2) (1994).

[48] A. Tolk, S.Y. Diallo, Model-based data engineering for Web services, IEEE Internet Computing 9 (4) (2005).
[49] G. Tonti, J. Bradshaw, R. Jeffers, R. Montanari, N. Suri, A. Uszok, Semantic Web languages for policy representation and reasoning: a comparison of KAoS, Rei, and Ponder, in: Proceedings of the 2nd International Semantic Web Conference (ISWC'2003), Sanibel Island, Florida, USA, 2003.
[50] A. Uszok, J. Bradshaw, R. Jeffers, N. Suri, P. Hayes, M. Breedy, L. Bunch, M. Johnson, S. Kulkarni, J. Lott, KAoS policy and domain services: toward a description-logic approach to policy representation, deconfliction, and enforcement, in: Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'2003) in conjunction with the 8th ACM Symposium on Access Control Models and Technologies (SACMAT 2003), Lake Como, Italy, 2003.
[51] K. Vidyasankar, G. Vossen, A multi-level model for Web service composition, in: Proceedings of the IEEE International Conference on Web Services (ICWS'2004), San-Diego, California, USA, 2004.
[52] X. Yu, L. Zhang, Y. Li, Y. Chen, WSCE: a flexible Web service composition environment, in: Proceedings of the IEEE International Conference on Web Services (ICWS'2004), San-Diego, California, USA, 2004.

Zakaria Maamar is an associate professor in computer sciences at Zayed University, Dubai, United Arab Emirates. His research interests include Web services, software agents, and context-aware computing. Maamar has a Ph.D. in computer sciences from Laval University. Contact him at zakaria.maamar@zu.ac.ae.

Djamal Benslimane is a full professor in computer sciences at Claude Bernard Lyon 1 University and a member of the Laboratoire d'InfoRmatique en Images et Systèmes d'information – Centre National De la Recherche Scientifique (LIRIS-CNRS), both in Lyon, France. His research interests include interoperability, Web services, and ontologies. Benslimane has a Ph.D. in computer sciences from Blaise Pascal University. Contact him at djamal.benslimane@liris.cnrs.fr.

Philippe Thiran is an associate professor in management information system at the Department of Business Administration of the University of Namur, Belgium. He is also a member of the Information Management Research Unit (IMRU). His research interests include Web services, databases and distributed information systems. Thiran has a Ph.D. in computer sciences from the University of Namur. Contact him at philippe.thiran@fundp.ac.be.



Chirine Ghedira is an associate professor in computer sciences at Claude Bernard Lyon 1 University and a member of the Laboratoire d'InfoRmatique en Images et Systèmes d'information – Centre National De la Recherche Scientifique (LIRIS-CNRS), both in Lyon, France. Her research interests include Web services and context-aware computing. Ghedira has a Ph.D. from in computer sciences from the National Institute for Applied Sciences (INSA). Contact her at chirine.ghedira@liris.cnrs.fr.

Schahram Dustdar is a full professor at Vienna University of Technology. He is also an honorary professor at the University of Groningen, The Netherlands. He co-authored more than 120 publications in journals, conferences and book chapters. More information can be found at: www.infosys.tuwien.ac.at/Staff/sd.

Subramanian Sattanathan is a Software Engineer at IBM India Software Labs, Bangalore, India. His research interests include Web services, distributed computing, and security. Sattanathan has a Ph.D. in computer sciences from the National Institute of Technology Karnataka, India. Contact him at sattanathan@in.ibm.com, sattanathan@gmail.com.