International Journal of Pervasive Computing and Com...

A user-aware approach for describing and publishing context aware composite Web service

Sihem Cherif Raoudha Ben Ben Djemaa Ikram Amous

Received 25 January 2016
Revised 25 January 2016
Accepted 1 February 2016

A user-aware approach for describing and publishing context aware composite Web service

Sihem Cherif, Raoudha Ben Ben Djemaa and Ikram Amous
MIRACL, ISIMS, Sfax, Tunisia

Abstract

Purpose – The purpose of this paper is to describe the composite service and the context properties related to the users in the business process execution language (BPEL) file.

Design/methodology/approach – The authors' approach allows expressing requirements by taking into account potential users' context in addition to the functional one.

Findings – In this paper, the authors introduce a new context-aware approach that provides a dynamic adaptation of service compositions.

Originality/value – This paper introduces a user-aware approach for describing and publishing context-aware composite Web service.

Keywords AWS-WSDL, CAC-WSR registry, Dynamic context, SABPEL, UDDI

Paper type Research paper

1. Introduction

A characterizing feature of service-oriented architectures (SOA)-based systems (Erl, 2009) is their high dynamicity in selecting the functions that satisfy user requirements. Service composition plays a fundamental role in these kinds of software systems. So far, many researchers have investigated different techniques to support the automatic generation of service compositions from a set of published services (domain), given a specific goal to reach (problem). SOA suffers from a number of limitations and weaknesses in the context of composition on demand; hence, Web services adaptation to context remains essential to better exploit services. Web services composition based on "context" can give a personalized behavior to the client by using information about the client. The information can be anything, such as a person's profile, which includes name, country, language and his current location, or it can also be the device on which he intends to invoke the Web services. For example, if the client wants to access a Web page on his mobile device, then the page needs to be modified or customized so that it can fit into the requirements of the mobile device, i.e. the memory to be used.

It can be customized so that the heavy graphics can be avoided, considering the low bandwidth of the device.

In addition, most of the existing Web services compositions are not originally developed to be adaptable. Essentially, tools and mechanisms are required to describe and publish adaptable atomic Web service and composite Web service. In fact, current Web service standards Web service description language (WSDL), universal

International Journal of Pervasive Computing and Communications
Vol. 12 No. 2, 2016
pp. 174-193

1742-7371
DOI 10.1108/IJPCC-01-2016-0011

description discovery and integration (UDDI) and business process execution language (BPEL) are limited to functional view and do not support context information.

A number of efforts have been made to deal with nonfunctional characteristics. Due to Adaptable Web Service Description Language (AWS-WSDL) (El Hog *et al.*, 2014), the authors are able to describe the functional and nonfunctional properties related to the users' profile in the same description file.

Even though previous solutions take into account nonfunctional properties of Web service, they are limited to the atomic Web service. Moreover, most of the researches focus only on the publishing of an adaptable atomic service description level; they have not dealt with the publishing issue of the adaptable service composition.

In the following, we propose a UDDI extension that supports context-aware service composition based on the AWS-WSDL (El Hog *et al.*, 2014) and self-adaptive business process execution language (SABPEL) (Cherif *et al.*, 2015a). Due to SABPEL, we describe the composite service and the context properties related to the users in the BPEL file. We propose a new registry called context-aware composition Web service registry (CAC-WSR) that publishes adaptable composition Web services and allows users to select composite services according to their profiles.

The remainder of this article is organized as follows. Section 2 describes the related works. Section 3 explains the adaptable service composition architecture. Section 4 describes the SABPEL. Section 4 explains CAC-WSR. Section 5 summarizes the publishing of adaptable atomic and composite Web service. Section 6 describes the CAC-WSR experimentation. Finally, Section 7 discusses and concludes the study.

2. Related works

In this section, we take a look at some research works interested in the possibilities of integrating the self-adaptive mechanisms in the Web service composition. We provide an overview of some of these works.

Portchelvi and Venkatesan (2015) proposed a goal-directed orchestration (GDO) approach which uses an orchestration engine to provide flexibility in responding to the changes in a dynamic services environment. In the GDO process, the user request for service is given to the engine and the composition request generator generates an abstract goal tree and concrete goal tree. The abstract goal tree is mapped to abstract task tree and the concrete goal tree is mapped to the concrete task tree, and thereby a composition plan is generated with abstract service descriptions. If services are not available to construct a composition plan, the goal cannot be achieved, leading to goal failure. This failure is reported back to the composition requestor, and an alternate sub-goal for the failed sub-goal is found and given to the composition plan generator. However, the authors need to formalize this goal-directed approach using formal methods.

Martin (2013) deals with the problem of the flexible composition by automated planning for which he proposes a model. The sequence and the choice operators are defined and used to characterize flexible plans. Two other operators are then derived from the sequence and the choice operators: the interleaving and the iteration operators. The author refers to this framework to define the flexibility produced by my planner, Lambda-GraphPlan (LGP), which is based on the planning graph. The originality of LGP is to produce iterations. Martin (2013) shows that LGP is very efficient on domains that allow the construction of iterative structures.

Vukovic (2007) proposed a framework for building context-aware applications on demand as dynamically composed sequences of calls to services. She presents the design and implementation of a system, which uses goal-oriented inferencing for assembling composite services, dynamically monitors their execution and adapts applications to deal with contextual changes. To handle composition failures, the author introduces GoalMorph, a system which transforms failed composition requests into alternative ones that can be solved.

Butt (2013) proposed a Trend-based Service Discovery Protocol (TRENDY), a new compact and adaptive registry-based service discovery potocol with context awareness for the Internet of Things, with more emphasis given to constrained networks. TRENDY's service selection mechanism collects and intelligently uses the context information to select appropriate services for user applications based on the available context information of users and services. In addition, TRENDY introduces an adaptive timer algorithm to minimize control overhead for status maintenance, which also reduces energy consumption. Its context-aware grouping technique divides the network at the application layer, by creating location-based groups. This grouping of nodes localizes the control overhead and provides the base for service composition, localized aggregation and processing of data.

Grati *et al.* (2012) proposed a quality of service (QoS) monitoring framework QoSMonitoring and detection of SLA violations (QMoDeSV) for composite Web services implemented using the BPEL process and deployed in the Cloud environment. The proposed framework is composed of three basic modules: to collect low- and high-level information, analyze the collected information and take corrective actions when service level agreement (SLA) violations are detected. This framework provides for a monitoring approach that modifies neither the server nor the client implementation. In addition, its monitoring approach is based on composition patterns to compute elementary QoS metrics for the composed Web service.

The Adaptation of Semantic Web Service Composition to Context (ASWSCC) method (Mcheick and Hannech, 2013) is a model which ensures, on the one hand, that this model allows management and takes into account the user context that makes the composition process adaptable to different instances of the use context, which may change during the same session. On the other hand, the purpose of matching Web services during the composition process by using domain ontology as the lexical database is to identify, classify and relate to semantic content and lexical language in different ways.

Furno and Zimeo (2014) presented a design approach based on a semantic model for context representation. It is an extension of the OWL-based Web service ontology (OWL-S) ontology aimed at enriching the expressiveness of each section of a typical OWL-S semantic service description, by means of context conditions and adaptation rules. The model proposed by the authors allows the use of context-awareness expressions in semantic service descriptions and their adoption during composition.

Hermosillo *et al.* (2010) described complex event processing for context-adaptive processes in pervasive and heterogeneous environments (CEVICHE), a framework that combines complex event processing (CEP) and aspect-oriented programming (AOP) to support dynamically adaptable business processes. The adaptation logic is defined as aspects (reconfiguration component), and adaptation situations are specified by CEP rules (monitoring component). However, decision-making is not specified as a component in this framework. It is integrated into the defined aspects.

As we have shown, work on adaptation at service composition modeling is prolific. Different scopes and mechanisms are defined for achieving such adaptations.

Cheng *et al.* (2015) presented an automatic Web service composition method that deals with both input/output compatibility and behavioral constraint compatibility of fuzzy semantic services. First, user input and output requirements are modeled as a set of facts and a goal statement in the Horn clauses, respectively. A service composition problem is transformed into a Horn clause logic reasoning problem. Next, a fuzzy predicate Petri net (FPPN) is applied to model the Horn clause set, and T-invariant technique is used to determine the existence of composite services fulfilling the user input/output requirements. Then, two algorithms are presented to obtain the composite service satisfying behavioral constraints and construct an FPPN model that shows the calling order of the selected services.

Madkour *et al.* (2013) proposed a three-phase adaptation approach: first, they select the suitable services to the current context and we recommend them to the adaptation process. Second, in the service adaptation phase, they perform adaptation by using fuzzy sets represented with linguistic variables and membership degrees to define the user's context and the rules for adopting the policies of implementing a service. Finally, they deal with the complex requirements of the user by the composition of the obtained adaptable atomics services.

Various techniques have been proposed in literature that integrates existing services based on several pieces of information. Although there is a rich landscape in adaptation-related researches, a complete and generic context-aware composition approach is still missing. Table I summarizes the comparison of context-aware service composition works. All of the identified features are not present in a single work, as they are focused on different research problems. Most surveyed architectures use a centralized composition model. They address dynamism in the composition, primarily from the perspective of unavailability of selected Web services, and deal with the issues of how to replace them with other equally capable Web services to perform the desired task.

Context-aware service composition frameworks should support the following features:

- *The automatic description composite service*: Static service composition involves precompilation of the composite service prior to a user's request. Description composite service is essential for exploiting the current state of available services and making adaptations based on runtime parameters, such as bandwidth and the cost of executing the various services.
- *User interaction*: Whilst service composition is an automated process, it is necessary to allow users to provide feedback when they so wish or moreover be integrated in the composition process. For example, aside from providing input parameters, users may need to guide the composition by selecting the services and re-defining the goals or guiding the failure recovery process.
- *Automatic composite service discovery*: Working with a limited domain of services or predefined service types limits the potential of service composition. Moreover, new services, possibly with new capabilities, may become available or existing ones may change their functionality. Having an automated means of service discovery is therefore an essential feature.

Table I. Main research challenges and features of automatic Web service composition

Feature	Research work									
	GoalMorph (Vukovic, 2007)	TRENDY (Butt, 2013)	QMoDeSV (Grati *et al.*, 2012)	ASWSCC method (Mcheick and Hannech, 2013)	Furno and Zimeo's (2014) approach	CEVICHE (Hermosillo *et al.*, 2010)	Three-phase adaptation approach (Madkour *et al.*, 2013)	Cheng *et al.*'s, (2015) method	GDO (Portchelvi and Venkatesan, 2015)	LGP (Martin, 2013)
Composition method	Context-aware goal model	6LoWPANs	BPEL extension	State planner	State planner	BPEL extension	Fuzzy sets	FPPN	Goal-oriented action planning	GraphPlan extension
Service markup	OWL-S	XML	BPEL4WS	BPEL4WS	OWL-S	BPEL	OWL	SAWSDL	ND	Graph
Composition model	Central	Distributed	Central	Central	Central	Central	Central	Central	Central	Central
Context monitoring	PDDL-based context goal	Directory agent (DA)	N	Pairs (attribute, value)	OWL-Ctx	Complex event processing	OWL	N	N	Y
Automatic description composite service	Y	Y	ND	N	Y	N	Y	N	Y	N
User interaction	Y	Y	ND	Y	Y	N	Y	N	PD	N
Automatic composite service discovery	Y	Y	N	ND	N	N	N	ND	N	N

Notes: Y = done; N = not done; P = partially done; PD = partially discussed; ND = not discussed

- *Context monitoring*: For the purpose of supporting dynamic adaptations, software should be aware of changes in its context. Context-aware systems are concerned with the acquisition of context, the abstraction and understanding of context and application behavior based on the recognized context.

3. Adaptable service composition architecture

To reach the full potential of Web services, they can be combined to achieve specific functionalities. If the implementation of Web service business logic involves the invocation of other Web services, it is called a composite service. The process of assembling a composite service is called service composition. Web services run in a context (e.g. their operating computing infrastructure). In an ideal scenario, Web service operations would do their job smoothly. However, several exceptional situations may arise in the complex, heterogeneous and changing contexts where they run.

For example, one of the partner services may go down or it may be updated to require new policies, etc. When such situations occur, the composite Web service will not operate appropriately and runtime faults will be thrown. Most available Web service orchestration engines do not provide automated support for detecting and reacting to such situations, and handling them can be done through manual human intervention, i.e. the fault must be detected, the Web service-based process that threw the fault needs to be undeployed, certain changes should be applied either to the process itself or to its deployment configuration (e.g. replacing the faulty partner service, changing the setting of the interactions to be secure, etc) and then restarting the composite service. Such an approach is inappropriate because the operation of the composite service is discontinued, because certain processes may be interrupted in the middle of a business transaction, and also because of the huge administrative overhead.

The need of managing adaptive services composition impels the incorporation of facilities to deal with users' context in all the services composition life cycle. We define the users' context as a set of context information composed of device characteristics, connection, location and a set of preferences in terms of content and presentation.

To achieve this objective, we propose in this paper a framework that supports the development, the description and the publication of adaptive service composition (ASC). Our goal is to offer relevant and suitable information depending on a user's particular profile. Therefore, developers have to integrate software facilities dealing with the context characteristics. Then, context annotations must be used to extend BPEL descriptions. Moreover, UDDI must be extended to include contextual information of the service composition in addition to the atomic and composite Web service information currently stored in UDDI registries. Our framework is organized in a layered architecture, and it consists mainly of four layers providing specification, description and publication solutions.

Figure 1 illustrates an overview of our framework's main layers. Each one describes our proposal to a given stage of the Web service life cycle.

- *Description layer*: It concentrates on the annotation of the BPEL description document, with contextual information managed by the composite service. The

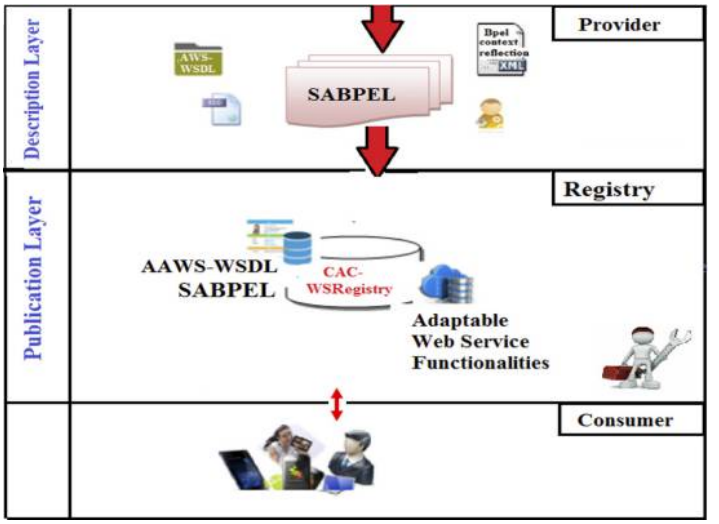

Figure 1.
Adaptable Web service composition main layers

adaptable description named SABPEL (Cherif *et al.*, 2015a) is automatically generated from the service implementation using our SABPEL generator.

- *Publication layer*: It gathers atomic Web services features, composite Web service and context supported on a UDDI extended registry named CAC-WSR (Cherif *et al.*, 2015b).

The following sections will concentrate on a detailed description of each layer.

4. Description layer

To allow the provider to publish adaptation criteria on the one hand and the consumer to choose from a list of published Web services compositions the most appropriate description according to his needs and profile on the other hand, we suggest additional information in the description file. We therefore propose an extension of the two standard WSDL and BPEL having the ability to describe composite and atomic service profile information. Our extension, named AAWS-WSDL and SABPEL, is detailed in the study of Cherif *et al.* (2015a). The SABPEL document is the output of our description layer.

The SABPEL is obtained through a model transformation from the adaptive design described in the previous section (Figure 2).

Figure 3 illustrates the transformation process in the description layer.

4.1 Context-aware service composition: intersection of atomic Web service context

For our study, we present the composite context information concept as shown in Figure 4. First, we have to define the composite context information's concept for our research.

We define the composite context information, i.e. "Enhanced high level context information by integration context information related with atomic service for decide or

supply the context aware service composition of application". To provide composite contextual service, a context ontology is constructed (Figure 5).

Each elementary Web service has an atomic context ontology which is stored in an OWL file. This atomic context ontology is automatically generated from the AAWS-WSDL description file. Figure 6 shows an example of composite context intersection using three atomic contexts:

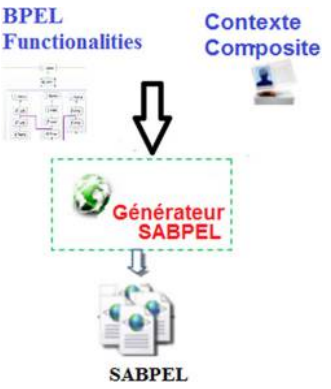

Figure 2.
Description layer of the adaptive service composition framework (ASCF)

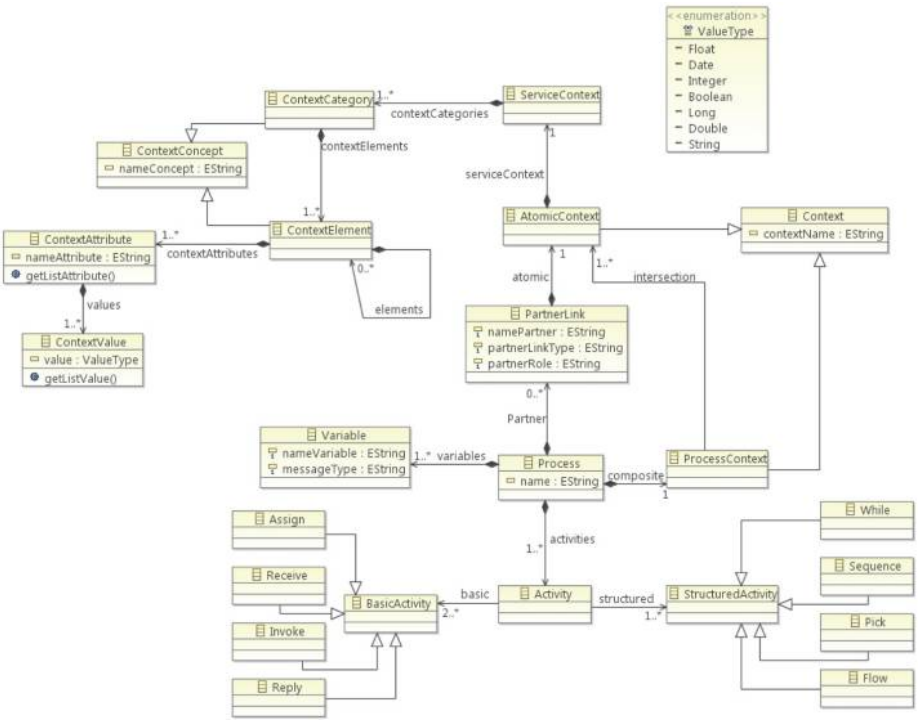

Figure 3.
Metamodel of the SABPEL

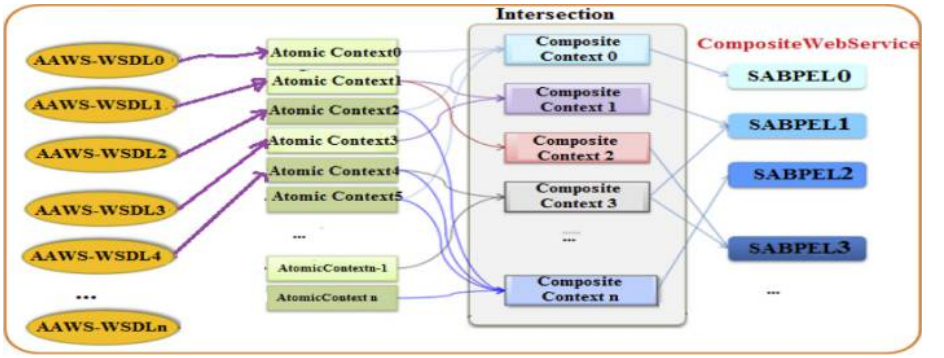

Figure 4.
Composite context information concept

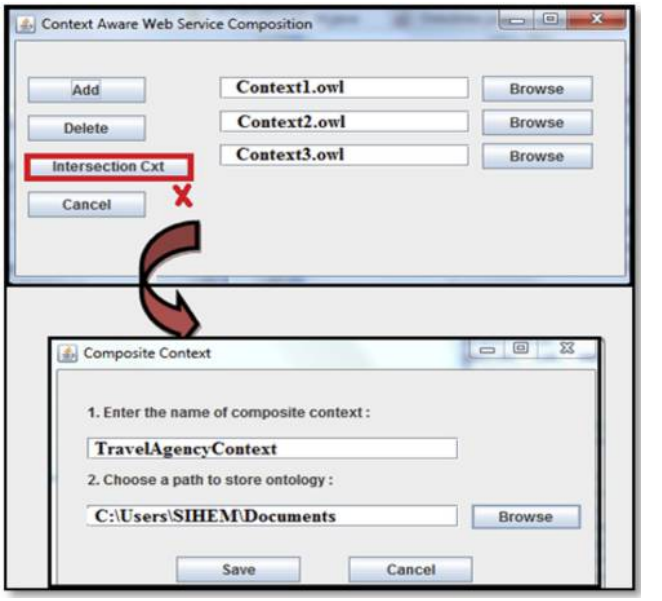

Figure 5.
Intersection of atomic Web service context

(1) *Context 1*: OWL describes the context of a hotel service (as shown in Figure 6).

(2) *Context 1*: OWL describes the context of an airline service (as shown in Figure 6).

(3) *Context 1*: OWL describes the context of a bank service (as shown in Figure 6).

4.2 Context-aware semantic service design

In fact, Figure 7 shows a composite context of travel agency service, which later uses three atomic contexts (i.e hotel, airline and bank contexts). The hotel service uses three languages

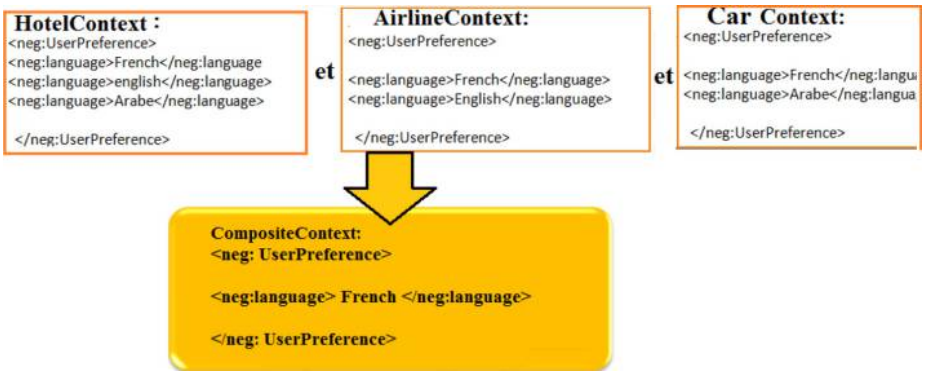

Figure 6.
Example of composite context

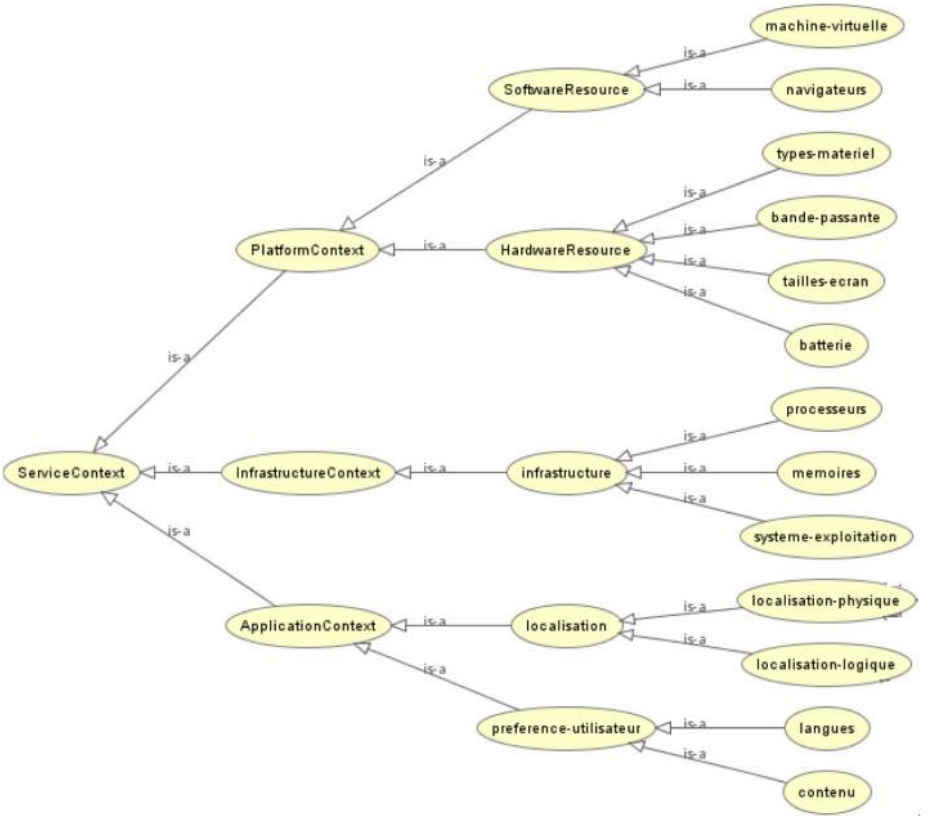

Figure 7.
OWL-SABPEL: ontology language

as users's preference (French, English and Arabic); the airline service uses two languages as users' preference (French and English); and the bank service uses two languages as users' preference (French and Arabic). Therefore, the composite context of travel agency uses one language (French) that is an intersection of the three atomic contexts.

To support design and composition of context-aware services, we propose an OWL ontology (OWL-SABPEL), supporting the description of sets of contexts in specific domains and an extension of the OWL-S ontology for services.

Context-aware semantic descriptions can be exploited during the service composition process to automatically generate compositions better-tuned to the requestor's behaviors and preferences and to the particular situation of the environment.

Figure 7 shows OWL-SABPEL, an extensible OWL ontology, for describing contexts according to AAWS-WSDL, described in Section 3. Profile, context and preferences are the main concepts in this ontology.

We can distinguish two types of contexts: composite and atomic. Composite context can be further intersected in one or more atomic contexts that can be helpful for a better characterization of the associated context aspect.

5. Publication layer

A service provider publishes its service at a service registry using a publishing interface. The public interfaces and binding information of the registered services are clearly defined in the WSDL standard language. A registry organizes the published services and provides a query interface that enables a service consumer to search for a needed service, and obtain its provider's location information (Figure 8).

A service consumer then interacts with a service provider through the simple object access protocol (SOAP) protocol. The most known registry is UDDI which is an XML-based standard that was proposed for allowing providers to publish their Web services, so that they can be located afterwards by consumers. A UDDI registry is structured into three components:

(1) BusinessEntity or white pages;

(2) BusinessService or yellow pages; and

(3) BindingTemplate or green pages.

These components allow the search for a suitable Web service in the UDDI1 registry according to the three data types. Unfortunately, Web service discovery through

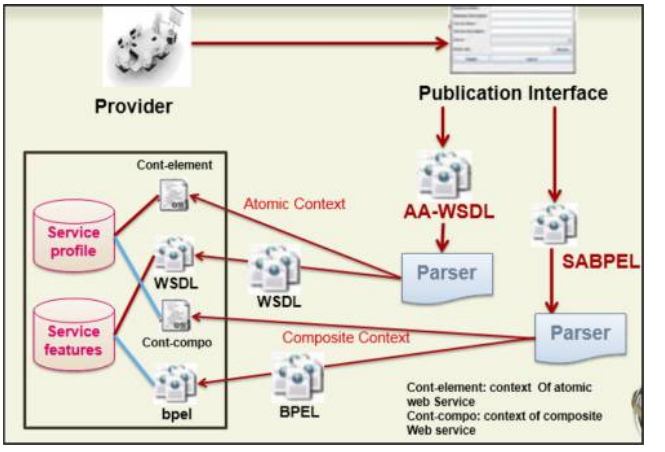

Figure 8.
Publication process in the CAC-WSR

UDDI-based registries is insufficient to carry out consumer adaptation requirements. This necessity is increasing every day with the emergence of portable devices and various users' preferences. Hence, the provider and the consumer want to have a registry providing a fairly precise publishing and search capabilities to render the use of user-aware Web services more efficiently. To overcome these needs, we propose a Web service registry offering the ability to save adaptation criteria of atomic and composite Web services. The registry is called CAC-WSR, and it supports the publication of the SABPEL description presented in Section 4.

We extend the UDDI data structure by an atomic adaptation criteria element which makes reference to the item UserProfile in the description file AAWS-WSDL. In addition, we extend the UDDI data structure by a composite adaptation criteria element which makes reference to the item UserProfile in the description file SABPEL. When a supplier requests the publication of his service at the registry, an analyzer module retrieves the AAWS-WSDL and SABPEL description files. Then, the description of the features and access points will be transferred to the services features database, and those corresponding to the adaptation criteria will be transmitted to the service profile database.

To implement our registry, we propose an extension of the jUDDI v3.2 API.

Therefore, we use the Apache Tomcat as a Web server and the MySQL to implement the UDDI database. We offer a user interface allowing the provider to specify atomic and composite service information and to load the description file of its service. Then, a parser analyzes two files (the AAWS-WSDL and SABPEL). It allows the extraction of the functional description carried out by the elements (types, portype, message, binding and service) and stores it in the database ServiceProfile.

Then, it extracts the profile description contained in the element UserProfile and saves it in the database ServiceFeature. The db-atomic-context table stores the profile of the atomic Web service. This table refers to the binding-template table that stores atomic Web service instances.

The db-bpel-context table stores the profile of the composite Web service. This table refers to the binding-template table that stores composite Web service instances.

Our extended registry is compatible with the basic UDDI and both of them could coexist in the same environment. The publication process is shown in Figure 9. The definition document which is associated with the supported profile is chosen by the provider via the publishing interface, as illustrated in Figure 10. The provider has to specify the enterprise information, the service information, the Web server and the description file.

CAC-WSR also contains a query analyzer module used in the selection and discovery phase. This module communicates with the database Web service functionalities and ServiceProfile to select the suitable list of Web services meeting the user's query in terms of functionality and adaptation. The query analyzer allows the customer to:

- extract the needs in terms of functionality through keywords; and
- extract the needs in terms of adaptation through a comparison between the user's profile and the service profile stored in the CAC-WSR. The user's profile is automatically detected by a detection module named WildCat Extension Detected module which is an extension of the WildCat API and it is stored in an XML file.

Composite service finding process over CAC-WSR is depicted in Figure 11.

Figure 9.
Publish AAWS-WSDL description of the atomic Web service

Figure 10.
Publish SABPEL description of the composite Web service

6. Illustration of use

In this section, we illustrate the feasibility of our proposed framework through the example of a travel agency composite Web service. The service client wants to book a room in a hotel based on his profile characteristics. We provide corresponding solution to each layer.

6.1 Description layer

The travel agency Web service modeled above using our tool service component-architecture (SCA) diagram is then implemented in Java. The next step is to generate the SABPEL description file gathering all needed information about the available functionalities, access links and supported profiles. Figure 12 illustrates the SABPEL generator interfaces.

As input, we provide the BPEL and the WSDL of the composition of the travel agency adaptable service. Then, the provider could select, through the plug-in interface, the composite context of travel agency (Figure 12). This later is an intersection of three atomic contexts (i.e. hotel, bank and airline). This composite context is included in the BPEL description file.

An extract of the generated description file (SABPEL) is illustrated in Figure 13.

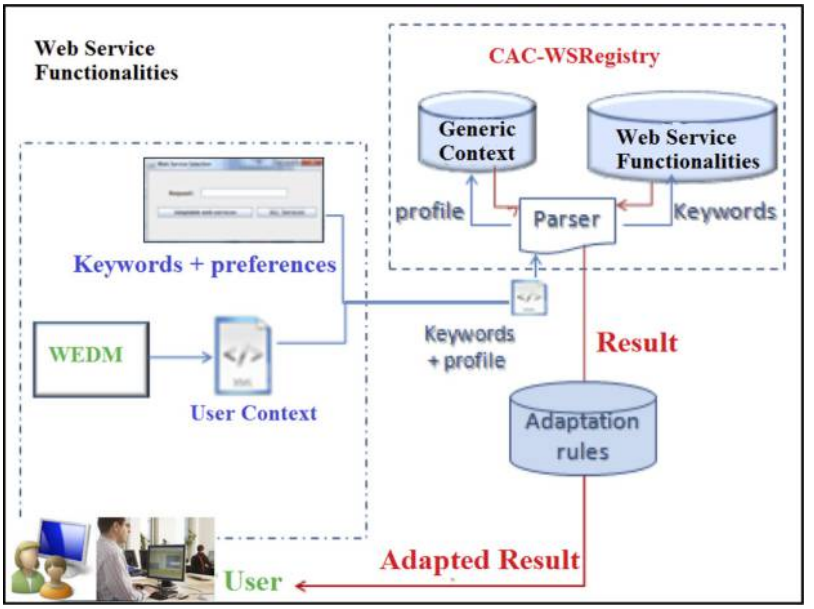

Figure 11.
Finding composite service process in the CAC-WSR

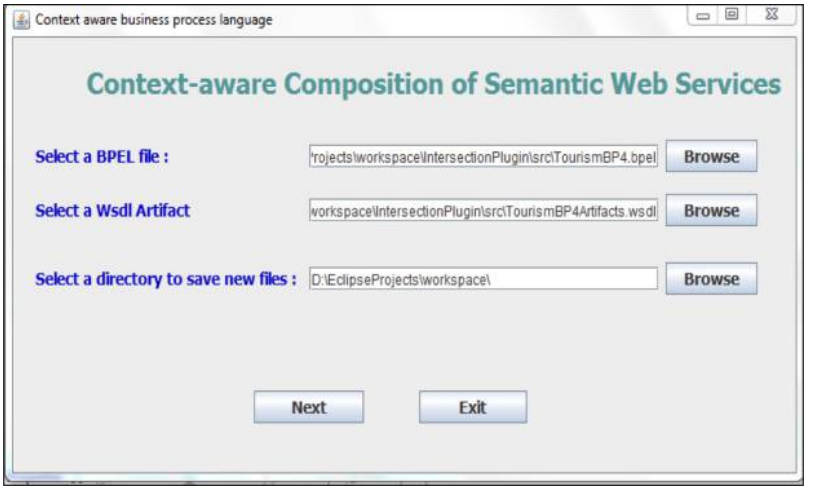

Figure 12.
The SABPEL generator graphical user interface (GUI) interfaces

6.2 Publication layer

Next step is to publish the SABPEL description of the composite service in our CAC-WSR. The provider has to specify the enterprise information, the service information, the Web server and the description file to load. The BPEL document enriched with the supported profile is saved in the db-bpel-context table (Figure 14).

Below, we present experiment results of the publication process in our CAC-WSR:

```
        </operation>
    </binding>

<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     SERVICE DEFINITION - A service groups a set of ports into
     a service unit.
     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
    <service name="TourismBP4Service">
        <port binding="tns:TourismBP4Binding" name="TourismBP4Port">
            <soap:address location="http://localhost:8283/ode/processes/TourismBP4"/>
        </port>
    </service>
  <!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     BPEL CONTEXT
     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
    <sabpel:context>
      <sabpel:location> TourismContext.owl   </sabpel:location>
      </sabpel:context>

</definitions>
```

Figure 13.
Extract from the travel agency description file

Filter: Edit: Export: Autosize:

entityKey	name	path
uddi:juddi.apache.org:a3718c48-6b90-49bf-b175-494d9c5a1dd8	carYoussefContext.owl	D:/EclipseProjects/workspace/AgenceDeVoyage(J
uddi:juddi.apache.org:cc92dfe0-984a-4cd5-a3c8-996e0925f68e	AirLineBookYoussefContext.owl	D:/EclipseProjects/workspace/AgenceDeVoyage(J
uddi:juddi.apache.org:f604747c-97e3-4741-975d-e2059f769eee	hotelContext23688.owl	D:/EclipseProjects/workspace/AgenceDeVoyage(J

Figure 14.
jUDDI publication of the airline and hotel Web service in the CAC-WSR

- The first experiment determined the performance of the standard UDDI by measuring the speed of the publishing function dealing with the BPEL document.
- The second experiment determined the performance of the CAC-WSR by measuring the speed of the publishing function dealing with the SABPEL document.

The load test involves exposing an application to real-use conditions before being put into production to predict the behavior of the system and diagnose problems of the application or infrastructure. In our framework, we need to perform scalability tests for our Web application.

These loads tests are justified by the large number of simultaneous users that can access CAC-WSR at a given time to achieve publication. To achieve these tests, we use the Apache JMeter tool.

First, we prepare and configure the test scenario of the composite service publication in the CAC-WSR (Figure 15).

6.3 Context-aware composition Web service registry evaluation

In this section, we provide experiment results publishing Web services in the CAC-WSR. We use the jUDDI v3.2 API which provides a Java-based open-source implementation of

the UDDI register. To save Web services description, the jUDDI could communicate with any relational database. In our work, we have chosen the MySQL database. We have extended the jUDDI API to be able to extract and save profile characteristics from the SABPEL document. We also extend the jUDDI database with a table named db-profile to save the service profile's information. Our experiments are organized into two parts.

The load test involves exposing an application to real-use conditions before being put into the production to predict the behavior of the system and diagnose problems of the application or infrastructure.

In our framework, we need to perform scalability tests for our Web application. These loads tests are justified by the large number of simultaneous users that can access CAC-WSR.

These loads tests are justified by the large number of simultaneous users that can access CAC-WSR at a given time to achieve publication. To achieve these tests, we use the Apache JMeter[1] tool. Apache JMeter may be used to test the performance both on static and dynamic resources (Web services [SOAP/REST], Web dynamic languages, etc.). It can be used to simulate a heavy load on a server, group of servers, network or object to test its strength or to analyze the overall performance under different load types. You can use it to make a graphical analysis of performance or to test your server/script/object behavior under heavy concurrent load.

We used Apache JMeter for testing the performance of the Apache Tomcat when publishing the context-aware composition.

First, we prepare and configure the test plan of the composite service publication in the CAC-WSR.

Figure 16 illustrates the simulation of Apache Tomcat server performance when running the load test.

The red curve shows the occupation rate of the server which is at around 50 when all threads are executing. The blue curve shows the Apache Tomcat server support. The green curve shows the health of the server. When the server is loaded by running tests, its health is 70 and this value returns to 100 after stopping the tests. In conclusion, our server can manage this load test and answer 1,000 requests simultaneously without problems. Figure 17 illustrates the simulation of CPU, memory and I/O disk performance. The red curve corresponds to the use of the memory. This curve is almost linear. The pink curve has a low disk usage during the first 15 s. The collected values are between 0 and 10. The blue curve shows the CPU utilization. The curve is sinusoidal, showing a significant variation in the CPU utilization.

We conclude that all the resources of our system (CPU, memory and disk) can excellently manage these load tests and recover stability at the end of these tests.

7. Conclusion

The services composition can meet the increasingly complex needs of a user, by combining Web services within a single business process. However, despite this

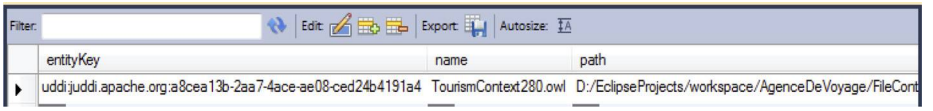

Figure 15.
Publication of the travel agency Web service in the CAC-WSR

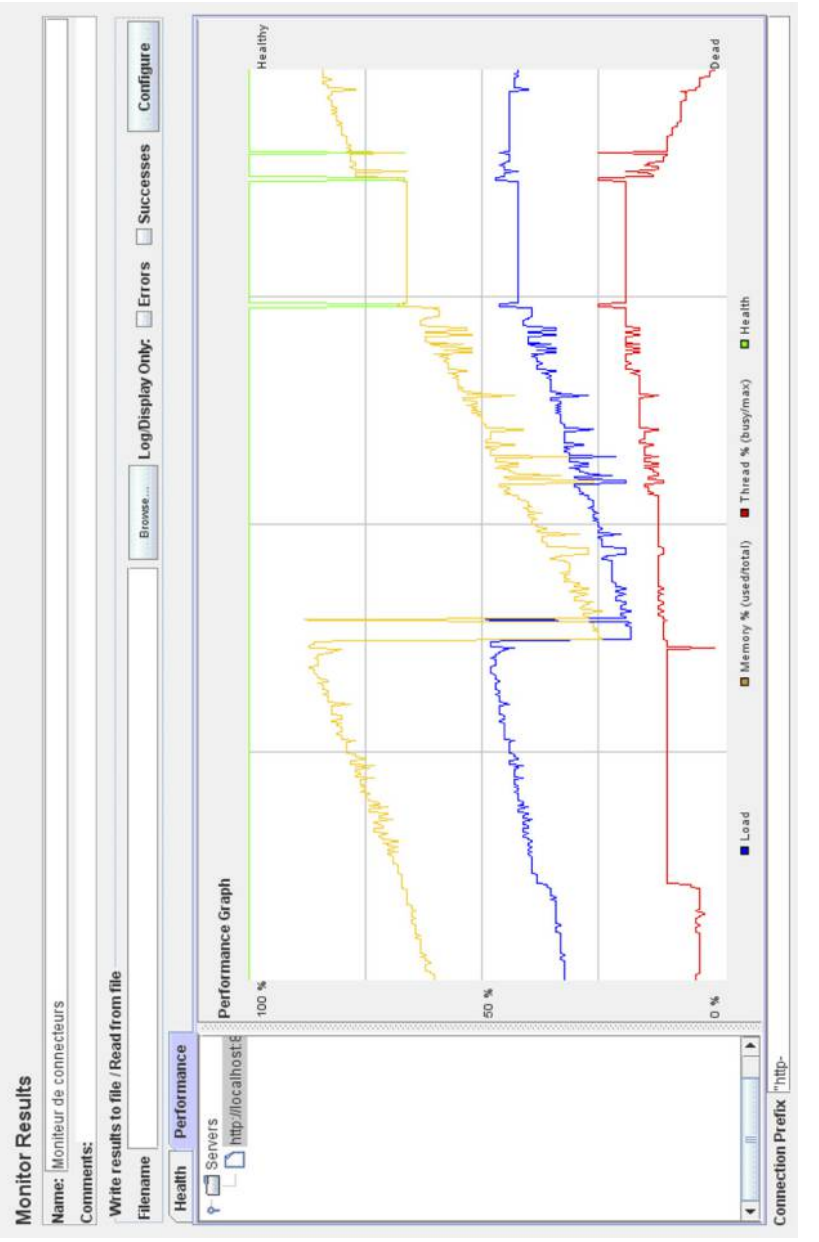

Figure 16.
Apache Tomcat server performance when running the load test

Figure 17.
Simulation performance (CPU, memory and disk I/O)

widespread adoption of Web services, many obstacles prevent their reconciliation in the composition, or may occur within a BPEL process at runtime the context for example. In this paper, we proposed an adaptable Web service registry called CAC-WSR. Our solution was based on SABPEL and AWSDL to publish a context-aware composition. It supports saving and searching atomic and composite Web services according to adaptation criteria together with functional needs. We detailed our registry data structure and implementation. Our extension shows the importance of context awareness in Web service composition.

For future work, we will extend our approach to be used in the open world, in which the services composition should react to continuous and unanticipated changes in complex and uncertain contexts.

Note

1. http://jmeter.apache.org

References

Butt, A.T. (2013), "Provision of adaptive and context-aware service discovery for the internet of things", Doctoral thesis, Loughborough University, Loughborough.

Cheng, J., Liu, C., Zhou, M.C., Zeng, Q.T. and Yla-Jaaski, A. (2015), "Automatic composition of semantic web services based on fuzzy predicate Petri nets", *IEEE Transactions on Automation Science and Engineering*, Vol. 12 No. 2.

Cherif, S.R., Djemaa, R.B. and Amous, I. (2015a), *SABPEL: Creating Self-Adaptive Business Processes*, ICIS 2015, Las-Vigas, NV, pp. 619-626.

Cherif, S.R., Djemaa, R.B. and Amous, I. (2015b), "Adaptable web service registry for publishing context aware service composition", *The 17th International Conference on Information Integration and Web-based Applications & Services (iiWAS2015)*, Brussels.

El Hog, C., Djemaa, R.B. and Amous, I. (2014), "A user-aware approach to provide adaptive web services", *The Journal of Universal Computer Science*, Vol. 20 No. 7, pp. 944-963.

Erl, T. (2009), "SOA design patterns", The Prentice Hall Service-Oriented Computing Series.

Furno, A. and Zimeo, E. (2014), "Context-aware composition of semantic web services", *Journal: Mobile Networks and Applications*, Vol. 19 No. 2, pp. 235-248.

Grati, R., Boukadi, K. and Ben-Abdallah, H. (2012), "A QoS monitoring framework for composite web services in the cloud", *ADVCOMP 2012: The Sixth International Conference on Advanced Engineering Computing and Applications in Sciences*, Barcelona, 23-28 September.

Hermosillo, G., Seinturier, L. and Duchien, L. (2010), "Using complex event processing for dynamic business process adaptation", *Proceedings of the 7th IEEE 2010 International Conference on Services Computing (SCC 2010)*, *Miami, FL*.

Mcheick, H. and Hannech, A. (2013), "Semantic web services adaptation and composition method", *ICIW 2013: The Eighth International Conference on Internet and Web Applications and Services*, Rome, 23-28 June.

Madkour, M., El Ghanami, I. and Maach, A. (2013), "Context-aware service adaptation: an approach based on fuzzy sets and service composition", *Journal of Information Science and Engineering*, Vol. 29, pp. 1-16.

Martin, C. (2013), "Composition flexible par planification automatique", Doctoral thesis, University Joseph Fourier, Laboratory of Informatics of Grenoble, Saint-Martin-d'Hères.

Portchelvi, V. and Venkatesan, P. (2015), "A goal-directed orchestration approach for agile service composition", *International Journal on Information Technology and Computer Science*, Vol. 3 No. 1, pp. 60-67.

Vukovic, M. (2007), "Context aware service composition", Technical Report number 700.UCAM-CL-TR-700-ISSN-1476-2986.

Further reading

Ardagna, D., Baresi, L., Comai, S., Comuzzi, M. and Pernici, B. (2011), "A service-based framework for flexible business processes", *IEEE Software*, Vol. 28 No. 2, pp. 61-67.

Azmeh, Z. (2011), "A web service selection framework for an assisted SOA", Doctoral thesis, LIRMM – University of Montpellier 2, Montpellier.

Cherif, S.R., Djemaa, R.B. and Amous, I. (2013), "ReMoSSA: reference model for specification of self-adaptive service-oriented-architecture", ADBIS'2013, Genoa, pp. 121-128.

Hafiddi, H., Baidouri, H., Nassar, M. and Kriouile, A. (2012), Context- Awareness for Service Oriented Systems, CoRR, abs/, 1211.3229.

Koning, M., Sun, C., Sinnema, M. and Avgeriou, P. (2009), "Vxbpel: supporting variability for web services in BPEL", *Information and Software Technology*, Vol. 51 No. 2, pp. 258-269.

Maamar, Z., Wives, L., Badr, Y., Elnaffar, S., Boukadi, K. and Faci, N. (2011), "LinkedWS: a novel Web services discovery model based on the Metaphor of social networks", *Simulation Modelling Practice and Theory*, Vol. 19, pp. 121-132.

Tigli, J.Y., Lavirotte, S., Rey, G., Hourdin, V. and Riveill, M. (2009), "Lightweight service oriented architecture for pervasive computing", *IJCSI International Journal of Computer Science Issues*, Vol. 4 No. 1, pp. 1694-1714.

UDDI Spec Technical Committee Draft, available at: http://uddi.org/pubs/uddiv3.html . (accessed 2007).

Yu. J., Han, J., Quan, Z.S. and Steven O.G. (2012), "PerCAS: an approach to enabling dynamic and personalized adaptation for context-aware services", *Proceedings of the 10th International Conference on Service-Oriented Computing-(ICSOC-2012)*, *Springer-Verlag, Berlin, Heidelberg*, pp. 173-190.

Corresponding author

Sihem Cherif can be contacted at: cherifsihem16@gmail.com