International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:9, No:7, 2015

Towards Natively Context-Aware Web Services

Hajer Taktak, Faouzi Moussa

***Abstract*—**With the ubiquitous computing's emergence and the evolution of enterprises' needs, one of the main challenges is to build context-aware applications based on Web services. These applications have become particularly relevant in the pervasive computing domain. In this paper, we introduce our approach that optimizes the use of Web services with context notions when dealing with contextual environments. We focus particularly on making Web services autonomous and natively context-aware. We implement and evaluate the proposed approach with a pedagogical example of a context-aware Web service treating temperature values.

***Keywords*—**Context-aware, CXF framework, ubiquitous computing, web service.

I. Introduction

RECENTLY, Web services and ubiquitous computing have started to be among the most important technologies. That is why the users found themselves faced with the necessity to build self-adaptive Web services. This strategy allows them to have more options as well as benefit not only of the advantages of Web services such as modularity and interoperability, but also to be aware of the user's context.

A Web service is an accessible application that other applications and humans can discover and invoke, and presents the following properties [1]: independent as much as possible from specific platforms and computing paradigms and primarily developed for inter-organizational situations.

Web services provide a standard means of interoperation between different software applications, running on a variety of platforms and/or frameworks.

W3C defines a Web service as a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically Web Service Description Language "WSDL"). Other systems interact with the Web service in a manner prescribed by its description using Simple Access Object Protocol "SOAP" messages, typically conveyed using Hypertext Transfer Protocol "HTTP" with an Extensible Markup Language "XML" serialization in conjunction with other Web-related standards.

Besides, Web services are defined as modular applications that perform specific tasks and follow a specific format. Their features are based on independent standards of a programming language or on a runtime platform. Web services have many standards as shown in Fig. 1:

Hajer Taktak is PhD student in Faculty of Sciences of Tunis, Tunisia (phone: 216-5578-7629, e-mail: taktakhajer@gmail.com).

Faouzi Moussa is Conference Master in Faculty of Sciences of Tunis, Tunisia (phone: 216-5590-9719, e-mail: faouzimoussa@gmail.com).

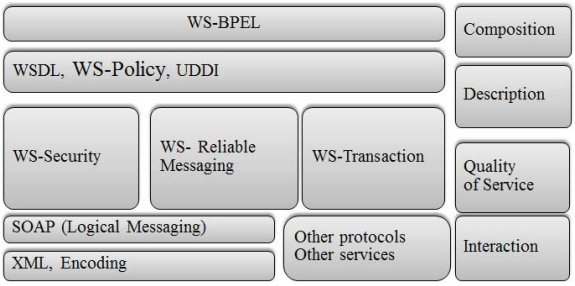

Fig. 1 Web services standards

We are going to give a brief description of the four main layers:

- The service discovery (UDDI) centralizes services into a common registry, and provides easy publish/find functionalities.
- The service description (WSDL) describes the public interface to a specific Web service.
- The XML messaging is responsible for encoding messages in a common XML format so that they can be understood at either ends.
- The transport layer transmits messages between applications [2].

Despite the widespread use of Web services, their definition still lacks the capacity to natively adapt their behavior to the context of use.

Context-aware Computing is a key aspect of the future computing environment. It aims to provide relevant services and information to the users, based on their situational conditions [3].

That's why building and deploying context-aware Web services allows systems-context interaction. It would be possible for example to consider the environments' aspects in which the Web services are to be executed [4].

It also raises a new set of research challenges.

These research works are dedicated to Web services and context-awareness. Nevertheless, these approaches are essentially focused on services composition, context manager and conceptual model that capture the domains' semantics.

In order to abstract the complexity of approaches cited below, we are going to add another layer responsible for the context-awareness so that Web services act dependently of the context of use. This will reduce the difficulty and the cost of building context-aware Web services.

In this paper, we first review literature on Web services involving context. Then, we introduce our approach for creating natively context-aware Web services. Finally, this approach will be illustrated by a pedagogical case study of a context-aware Web service treating temperature values.

II. Related Work

Research in the ubiquitous computing area has focused on pervasive computing and human-computer interaction. Web services technologies are designed for large-scale and loosely coupled environments. Such environments have to meet many requirements and must take context-awareness into consideration.

Reference [5] defines the context as any information describing a situation relative to persons, resources and services in service-oriented computing. It can include all information considered relevant in users' and services' interaction.

According to the observed works of [6]-[9], a context-aware system has many components that are separated and treated independently. These parts include context sensor, context storage, context consumer and context reasoner.

A. The Context Manager

The context manager is a component that manages context parameters of various entities. It also has the ability to detect every change occurring to the context of use.

The schema below shows a global view of the context manager's use:

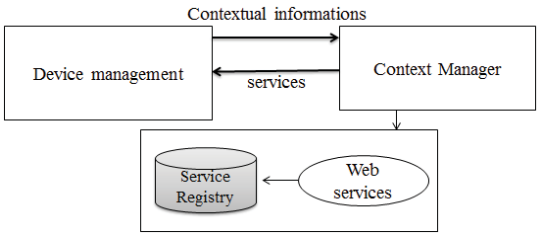

Fig. 2 Context-aware service architecture using context manager

Some approaches use the context manager to ensure the communication between services and the context provider.

In fact, to deal with context-aware Web services, [10] shows how to add to the architecture a context manager that manages the data's flow and transfers the main context's information to Web services. This technique uses essentially a context modeling system that helps in understanding which context is the most appropriate and what the information exchanged among Web services are.

Besides, [11] shows how to implement a context manager that communicates with the device and the service manager. This Framework maintains a knowledge base that stores and interprets the context's information in OWL format as well as its modeling, etc.

Reference [12] proposes a model (MUSIC) based on the context manager and other components to coordinate the process, support the execution, etc. Reference [13] proposes an approach gathering the relevant context information in the SOAP-Header. The context is extracted by this Framework and is transferred to the invoked Web service.

The Web service communicates with a (i) context plugin locally installed on the device and a (ii) context service defined using WSDL standards. This approach uses a context manager treating requests and responses.

B. The Services Composition

Reference [14] defines a context-aware pervasive Web service composition (CAPSC) that fits the evolution of the users' needs, the dynamic services' communication and the environment's changes.

The architecture is based essentially on two parts: contextualization and services' composition. In order to correctly implement these two parts, the authors used a Business Process Execution Language "BPEL" dealing with the initial service's composition with those containing context-awareness (Web services utilities). This approach is focused on an inference engine aware of the semantics and dependencies between Web services. It uses information from the BPEL application, decides which utility service to be included in the composition, then reports its findings to the BPEL process. Fig. 3 illustrates this work:

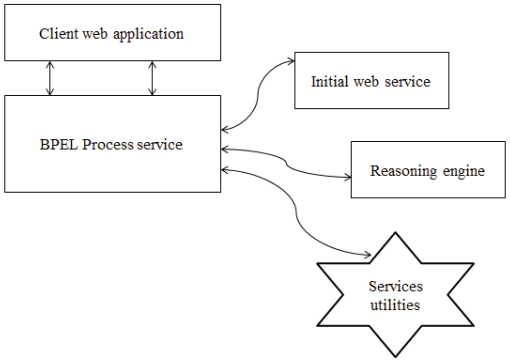

Fig. 3 Context-aware service architecture composition

Reference [15] defined a language (SVE-WSDL) improving Web service description in presence of semantic data. This architecture is based on a mediator allowing to deal with heterogeneous data and to avoid conflicts of interest in a services' composition. Thus, this work uses a semantic mediator to convert semantic values during exchanges between Web services.

Table I summarizes the techniques used by the approaches previously cited.

We compare these approaches by following several criteria: (i) how to register the context, (ii) how to detect the context and (iii) which tool is used to ensure the context-awareness.

We find out that the engine of the context management remains external and this can lead to orchestration, coordination, synchronization and execution issues.

According to these works, most approaches in the contextual Web service area have focused their research on how to model, store and access context information, as well as how to deal with the context in a service composition.

In fact, the authors gather contextual information through the intermediary of a context manager and orchestration tools which need deeper knowledge of architectures to ensure context-awareness in Web services.

TABLE I
APPROACHES COMPARISON

	[15]	[14]	[13]	[11]
Context registration	XML tags (<message>) injected into the WSDL file to define the context and semantics (part) of its use	context base	A context-aware Framework	Knowledge base managing context information and saves the metadata.
Context detection	The trigger acts on the context between WS compounds to ensure the cascade outputs in input.	context sensing	Soap header (block containing the context to be treated)	Sensors that send the context and metadata's device to the context manager
Inference Engine	Semantic mediator + Trigger	Composition adaptor	Context Framework	Composition adaptor
QoS	No	No	No	Yes (execution time)
Composition of WS/ Elementary WS	Composition of WS	Composition of WS	Elementary WS	Elementary WS

The contextualization, according to [15], uses an architecture based on a semantic mediator to retrieve the context values and associate them with the Web service. This remains a valid solution in a simple composition of services.

These existing infrastructures supporting context-awareness suffer from openness and scalability problems [16].

Key features of our approach:

- Achieving a service contextualization without resorting to existing tools, thus avoiding their complexity and realizing simpler solutions. Services should become natively contextualized, thus no more need of a third component performing this work.
- When implementing a context-aware Web service, the developer implements its own execution context inside of it by specifying features such as user, platform, environment, etc. Thereby, the context value interception is done inside the Web service which becomes natively contextualized as a result.

Thus, by developing context-aware Web services, it's possible to take into consideration (i) environmental aspects where Web services should be executed, (ii) users who manipulate Web services and specify their preferences, (iii) and a platform changing its behavior depending on the context of use. This awareness is important because it improves the adaptation of services and applications to situational changes during their operations. Therefore, the main objective of this work is to design and deploy adaptable Web services able to change their behavior according to changes occurring to the context. Thus, the service consumer has to invoke the contextual Web service by specifying his context of use, such as geographical position, mobile device, etc. The service will automatically make the changes according to the contextual information.

III. THE PROPOSED APPROACH

A. Model Overview

In this section, we describe the proposed approach for a context-aware service execution.

The purpose is to integrate contextual information into the initial definition of a Web service to avoid the complexity of orchestration tools. We also don't deal with complicated architectures based on adding a context manager which captures the context and interprets it in an understandable language to be subsequently consumed by a Web service.

We propose a model of context-aware Web service that fits automatically the context of use. Our model is inspired by the SOA approach to enforce the contextualization in service oriented architecture. It relies on the low level definition of the Web service in order to be aware of the context. Services are then executed and context-awareness is enforced in returned results.

Our main idea is to define a natively and autonomous self-adaptive Web service without the intervention of external agents livered to treat the context.

We assume that, alongside the Web service standards such as WSDL UDDI etc., we add another one treating the context as shown in Fig. 4:

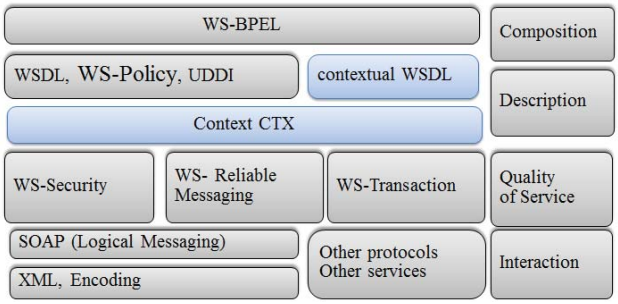

Fig. 4 Standards in a context-aware Web service

This model is inspired from the Web service's initial definition. The main goal is to add another layer to Web service standards to permanently have a context-sharing mechanism between Web services and clients.

According to our approach, the contextualization of a Web service doesn't introduce significant changes to the classic Web service architecture. In fact, the Web service itself contains implicitly the source code relative to context management.

Every Web service becomes able to change its initial behavior depending on the Web service consumer demand. Thus, the Web service becomes not only a platform-independent modular application, but also a computer program containing information relative to the context of use.

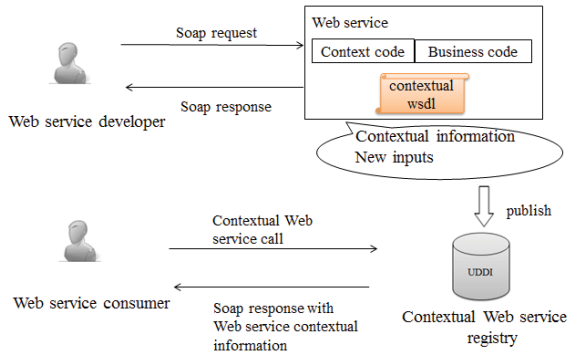

Fig. 5 Contextual Web service discovery and invoking

Fig. 5 shows a scenario invoking a contextual Web service.

The semantic data have to be defined by the developer to allow the service consumer to correctly interpret the service. The service developer must specify pertinently the context description parameters. During this work, we define the Web service's contextual data as a business code given by the developer in terms of his needs and preferences. This code will integrate the low level definition of the Web service to create a contextual one. Actually, the initial service Si has an input Is and an output Os. To have a context-aware result, we add another input for the context I c [1..n] leading to an output depending on the context Os,c (assign in Fig. 6). Note that the Web service can be executed with or without specifying the context parameters.

$$S_i <Is, O_s> \rightarrow S_i< (I_s, I_{c\,[1..n]}), O_{s,c} >$$

Fig. 6 The contextual Web service parameters

For example, during his reservation, a user in France may choose to have the price of a hotel in the USA in euro when the hotel booking web service computes the price in dollars.

Therefore, to achieve this goal, the user should enter the date of his reservation as an initial input of the service and a contextual input (geographical location) Ic[1..n] as a complementary data to satisfy his needs. Thus, the developer defines the Web service and its context so that we provide a personalized service according to the user's geographical position. Let's note that the 'context' and 'initial Web service' parts are developed separately to be able to change any of them without impacting the initial service execution. Besides, Web services should maintain their characteristics so that a Web service natively context-aware remains flexible, reusable and made with a delicate granularity.

When deploying the Web service, the system should be able to apply the treatment relative to the context's information in order to have a context-aware result and an output given according to the context data.

Fig. 7 presents the general service oriented architecture 'SOA' solution:

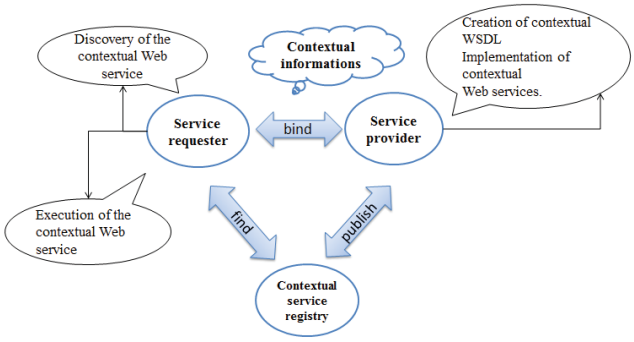

Fig. 7 SOA architecture including contextual information

With such an architecture including contextual information (assign in Fig. 7), a service provider can describe a context-aware Web service in the contextual WSDL language to specify how to invoke it during situational changes.

The contextual service registry creates an interoperable platform enabling a service requester to easily locate Web services over the Internet.

Our contributions are summarized as follows:

- We propose a context-aware Web service containing natively contextual information. We don't use composition of services anymore. The context modeling tool (ontologies) and service composition are not required. All requirements are summarized on (i) an initial Web service and (ii) contextual business code defined by the developer.
- The context-aware Web service will be autonomous.
- This work was integrated in the apache CXF architecture [17] and carried out in a thorough experimental evaluation.

Based on a scenario, we identify the following challenges addressed in this paper. The first one is how to guarantee autonomous context information treatment. Even if service consumption is not contextualized, the Web service must maintain execution independently of context (backward compatibility). The second one is how to preserve the Web service criteria such as simplicity, flexibility and low coupling and how to deal with heterogeneous data.

B. Rewriting the Web Service's Source Code Using CXF Framework to Take into Account the Context of Use

At this stage, the model presented extends the CXF Framework's architecture to which we add methods to recover a part of the context information entered by the developer.

This extensibility comes with the necessity to add context-awareness to the Web service under CXF Framework. As shown in Fig. 8, we extended the CXF architecture with a context module using both its input and output interceptors implemented as:

- Input interceptor: handles the input SOAP (Simple Object Access Protocol) messages and uses the methods to extract the content's request with the contextual data. The Web service is invoked via the contextual CXF Framework.

- Output interceptor: intercepts SOAP responses before their transmission to the service engine to make sure that Web services are built with context inclusion.

The output SOAP is built and the contextual results are sent to the service consumer.

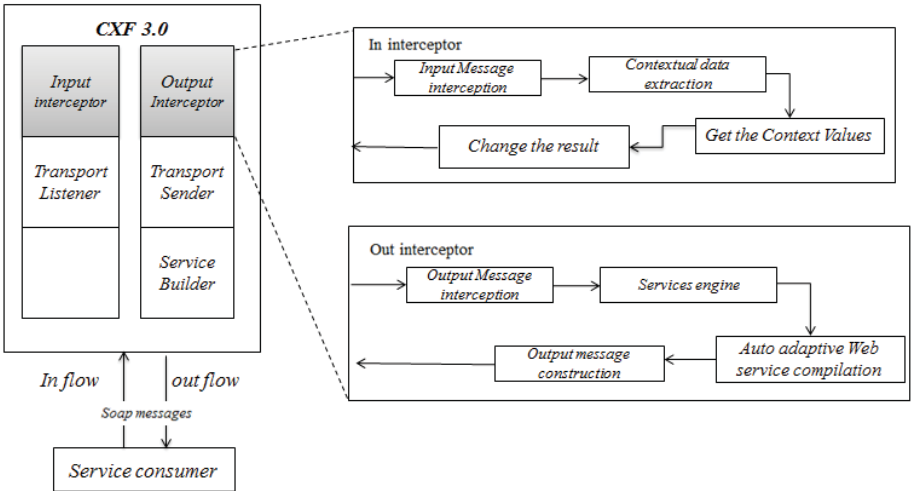

Fig. 8 Extensions made to the CXF Framework

Fig. 9 shows how CXF Framework behaves facing occurring changes during a contextual Web service's invocation.

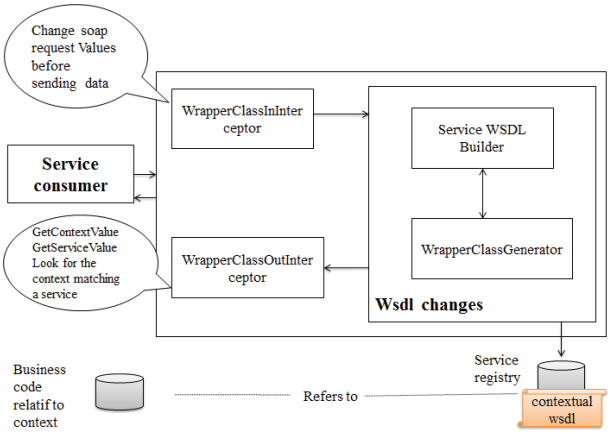

Fig. 9 CXF Framework behavior

Every SOAP request passes through the class "WrapperClassInInterceptor" that handles request data and identifies contextual information.

We extend this class with the ability to read and extract data specifying the context of use. Indeed, this class allows processing protocols and changing request values.

Regarding the class « WrapperClassInInterceptor »,

« ServiceWSDLBuilder » helps to interpret the context and to intercept operation values. Thanks to this class, we may produce a contextual WSDL model. Thus, « ServiceWSDLBuilder » generates the service description containing contextual information. From the operation's input, this class identifies context values and creates variables to be added as service values finally by « WrapperClassGenerator ».

Here we present a development plan of a service provider:

- Develop core functionality of a Web service and add the classes responsible for contextual information thanks to the annotations.
- Provide a service descriptor (WSDL) associated to the context-aware Web service; it must contain all information relevant to the context of use.
- Deploy and publish the self-adaptive Web service's specifications.

We note that contextual Web services can be used as simple Web services if users do not need context-awareness in their execution. Indeed, the service contextualization doesn't affect the current Web services' performance.

IV. Motivating Scenario

We are going to consider a scenario in which the user needs to know temperature values at a given date. The scenario is illustrated in Fig. 10:

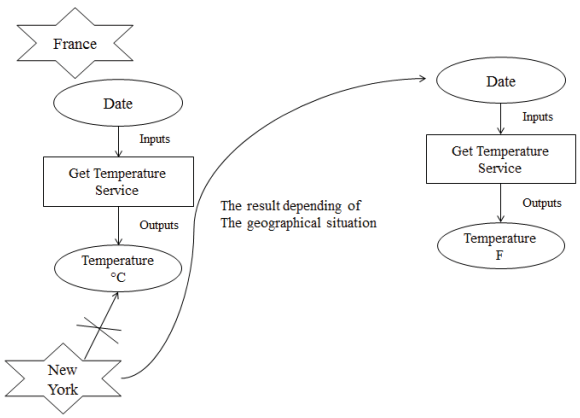

Fig. 10 The motivating scenario

The service is composed by an operation named « getTemperature ». This method allows us to have the temperature's degree at a given date. The initial output value is not context-aware and is always in Celsius degree. It doesn't match the unit of measure (°C, F) associated to the appropriate geographic position.

Based on our scenario, we assume that the first user is in New York and wants the result converted to Fahrenheit, unlike the second user who is in France and wants a result with Celsius degree. Obviously, the initial service can't do this work by its own as its results are independent of the users' positions and preferences.

This query can be solved by introducing the Web service to context notions for it to be natively context-aware and able to automatically change its behavior.

When the two service consumers call the same initial service, they have to see two different results depending on their positions as shown in Fig. 10. To do so, we contextualize the initial service by adding a "Geographical Position" variable retrieving the user's geographic location. The Framework intercepts the "Geographical Position" value then, according to it, converts the operation's return value to the desired unit of measure.

The Web service description language is modified according to the context (the added data). This change occurs when deploying the Web service. Thus, making it depends not only on the temperature but also on the context of use (geographical position).

The framework, called Context-aware Web Service Framework "CAWSF", presents a priori many contributions and shows how performant the context-aware Web service is.

Coming works will be more focused on characteristics managed by the quality of service (QoS). We present some of them in Table II:

The main aspect that differentiates our contribution from existing works is the transparent integration of context without modifying the initial structure of either the Web service, or the protocols, or the query languages giving access to services. The work is also done without adding external layers to treat the problematic.

TABLE II
VERIFIED CHARACTERISTICS OF QOS

	CAWSF
Performance	Treatment of context within the Web service and elimination of external components has improved the service's performance
Reuse and possibility of extension	The independence of context from the service offers the ability to add or modify context without having to change the service
Ease of implementation	The Web service contains the context management.
Interoperability	Using interceptor offers the possibility to interact with other systems
Flexibility	Adding context through an annotation retains the Web service's simplicity and warrants its flexibility by dynamically interacting with its environment

The work is entirely done inside the Web service. Developers or service consumers must enter their preferences to get a Web service depending on their context of use.

V. CONCLUSION

In this paper, we proposed an approach integrating context-awareness into Web service's structure without adding new resources. Our model exploits contextualization in the Web service and allows for contextual Web services. We implemented our model into the architecture CXF and evaluated its efficiency in a context-aware Web service scenario treating temperature values. As a future work, we intend to work on a comparative study with a more complete example to show all the advantages of our approach. Then we are going to focus on the quality of service (QoS) to verify the most relevant criteria important for the approach's reliability.

REFERENCES

[1] B. Benatallah, Q. Z. Sheng, and M. Dumas, "The Self-Serv Environment for Web Services Composition" In IEEE Internet Computing, 7(1), January/February 2003.
[2] E. Cerami, "Web Services Essentials" Distributed Applications with XML-RPC, SOAP, UDDI & WSDL Publisher: O'Reilly First Edition February 2002, pp.11-15.
[3] Chen, H., Finin, T., and Joshi, A. "Semantic Web in the Context Broker Architecture" In Proceedings of the Second IEEE international Conference on Pervasive Computing and Communications (Percom'04) (March 14 - 17, 2004).
[4] Z. Maamar, D. Benslimane and N. C. Narendra "What can context do for web services" ACM New York, NY, USA Volume 49 Issue 12, December 2006 pp. 98-103
[5] Schilit, B.N., Adams, N.I., Want, R. "Context-Aware Computing Applications." In: Procs of the IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'94), Dec. 1994, Santa Cruz, California, USA. pp.85-90.
[6] Chaari, T., Laforest, F. and Celantano, A. "Design of context-aware applications based on Web services", Technical report LIRIS UMR 5205 CNRS/INSA de Lyon/University´ Claude Bernard, Lyon 2004.
[7] M. Keidl and A. Kemper. "A framework for context-aware adaptable Web services." In Proceedings of The 9th International Conference on Extending Database Technology (EDBT'2004), Heraklion, Greece, 2004
[8] Mikalsen, M., Floch, J., Paspallis, N., Papadopoulos, G.A. and Ruiz, P.A. (2006), "Putting context in context: the role and design of context management in a mobility and adaptation enabling middleware", 7th International Conference on Mobile Data Management (MDM 2006), Nara, IEEE Computer Society, Nara.
[9] Henricksen, K., Indulska, J., McFadden, T. and Balasubramaniam, S. (2005), "Middleware for distributed context-aware systems", OTM Confederated International Conferences, Springer-Verlag, Heidelberg, pp. 846-63.
[10] Hong-Linh Truong and Schahram Dustdar SURVEY PAPER"A survey on context-aware Web service systems". This paper is partially funded by the EU FP6 WORKPAD, EU FP6 inContext and EU FP7 COIN projects.
[11] Sridevi S., Sayantani Bhattacharya, Pitchiah R. "Context-aware Framework." MobiQuitous conference 2010: pp. 358-363.
[12] Romain Rouvoy, Paolo Barone, Yun Ding, Frank Eliassen, Svein Hallsteinsen, Jorge Lorenzo, Alessandro Mamelli, and Ulrich Scholz:"MUSIC: Middleware Support for Self-Adaptation in Ubiquitous and Service-Oriented Environments." Springer-Verlag Berlin Heidelberg 2009
[13] Markus Keidl, Alfons Kemper: "Towards context-aware adaptable Web services". WWW (Alternate Track Papers & Posters) 2004: pp 55-65.
[14] Jiehan Zhou, Ekaterina Gilman, JuhaPalola, JukkaRiekki, Mika Ylianttila, Jun-Zhao Sun: "Context-aware pervasive service composition and its implementation." Personal and Ubiquitous Computing 15(3), 2011 pp 291-303.

[15] Michael Mrissa, Chirine Ghedira, Djamal Ben slimane, Zakaria Maamar: “Context and Semantic Composition of Web Services”. DEXA 2006: pp 266-275
[16] Aisha M. Salama Elsafty, Sherif G. Aly, Ahmed Sameh “The Context Oriented Architecture: Integrating Context into Semantic Web Services,” IEEE conference on semantic media adaptation and personalization, Athens 2006, pp 74-79.
[17] Naveen Balani and Rajeev Hathi “Apache CXF Web Service Development, Develop and deploy SOAP and RESTful Web Services” ISBN 9781847195401, pp 336