A Context-Aware Web Information System Based on Web Services

Bouchra Soukkarieh, Florence Sèdes

A Context-Aware Web Information System Based on Web Services

B. Soukkarieh, F. Sedes, *Member, IEEE*

***Abstract*— The increasing number of requirements for Web Information Systems and the increasing need for making their components interoperable ask for a new vision for the architecture of such Systems.**

We focus on two of these requirements: usability and interoperability. We present, in this paper, our contribution that aims at proposing a new architecture of Web Information Systems, supporting adaptation to the user's context and providing the user with a list of Web Services adapted to his context. This architecture is based on an extension of AHA! architecture through an adaptation layer containing various components dedicated to the context adaptation. So, our aim is to build an Adaptive Web Information System based on Web Services. Finally, we present an algorithm to build a user contextual profile "Profile-Context-User (UCP)" containing only the user's preferences that can be satisfied with the context.

***Index Terms*— Web Information System, Web Service, Adaptation, Context.**

I. Introduction

Modern Web Information Systems need to fulfil a large number of requirements. Among these requirements we mention the ability of a WIS to deliver to the user information that is not only adapted to his query, but also to his context (device, environment, etc.). Therefore, the context seems a promising idea in order to increase the usability of the WIS. In our work, we base on "Adaptive Hypermedia Architecture (AHA!)" that is regarded as a Web Information System (WIS) [14]. However, AHA! does not support the user's mobility; it only takes into account his characteristics (preferences, interests, etc.).

Today there is an increasing need for making WIS components interoperable. An isolated WIS is not able to provide all the information/computational power required by an organization. Web services (WS) appear to be the most popular mechanism to support application interoperability. Indeed, WIS is more and more based on the usage of Web Services (WS). But, the classical architecture of Web Services does not consider context adaptation.

In order to integrate the concept of adaptation in WIS and provide the user with a list of services adapted to his context, we propose a simple and general architecture extending the AHA!'s one. In this architecture, we integrate the classical Web Services architecture with the AHA! architecture adding to these architectures an adaptation layer containing various components dedicated to the context adaptation. So, our aim is to build an Adaptive Web Information System based on Web Services. The rest of this paper is as follows. In section 2, we present a state of the art about Adaptive Web Information System, taking the AHA! system as an example, and Web Services. In section 3, we detail our contribution which aims to propose a new architecture of Web Information System. This new architecture can, on one hand, support adaptation to the user's context and on the other hand, provide the user with Web Services adapted to his context. Then, we present an algorithm to build a user contextual profile "Profile-Context-User (UCP)" containing only the user's preferences that can be satisfied with the context. Finally, we illustrated our work by an example. We conclude, in section 4, by presenting our future works.

B. Soukkarieh, F. Sedes, IRIT, University Paul Sabatier, 118 Route de Narbonne, F-31062 TOULOUSE, (phone:33-0661152562; fax: 33-5 61 55 62 58; e-mail: {sokarieh, sedes}@irit.fr)

II. State of the art

A. Adaptive Web Information System (AWIS)

A Web Information System (WIS) is Information System first, and Web System second [3]. WIS uses the Web paradigm and technologies to retrieve information from sources connected to the Web, and present the information in a web to the user.

Indeed, the usability, the performance and the maintenance are the main measurements of the quality of WIS. Nowadays, users can reach these WIS through various devices (e.g., PDA, Smart Phone, PC, etc.). This heterogeneous, mobile and changing environment requires that the information and services, sent by the server, are adapted to precise conditions of their use. Therefore, the context seems a promising idea in order to increase the usability of the WIS.

In our research area "***Context-aware applications"***, the context is used to provide relevant information and services to the user [3]. The problem, in this case, is not only related to adapt content to the user computational capabilities. Other elements of the context, such as user's behavior or preferences, have to be considered in order to choose the most relevant information and service for the user.

The first work in our research area is interested only in the definition of the context, through a particularly limited vision. Schilit and Theimer [6], for example, consider only the localization of the user, the people who accompany him and

the objects which encounter him. Although the context is regarded as true centre of interest for research works of context-aware systems, *we note that the first work remains rather vague on the definition of context.*

Recent work in this area starts to define the context in a more general and clearer way. According to Lemlouma [17], for example, the context is considered as "the set of all information of the environment that can influence the adaptation process and the transmission of contents towards the end-user".

A generalized notion of context has been proposed by Dey: "context is 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. Context is typically the location, identity and state of people, groups, computational and physical objects." [5].

The first WIS taking into account the context are applications that aim to deliver to the user (a tourist who visits a city, a museum, etc) contents adapted to his localization and to the tourist activity. For example, the GUIDES application proposed by Cheverest [11] is a system designed to provide visitors of a city, armed with mobile devices, tourist information sensitive to the context. The main goal, in these applications, is to deliver automatically information to the user. *The problems dealt by these applications are in particular discovery of the localization and the adaptation of the contents.*

Other research work, like that of Schilit and that of Lemlouma, rather concerns the adaptation of the presentation of these contents delivered to the user according to physical capacities of the user device. Schilit underlines the use of various techniques for the adaptation of the presentation of the Web pages.

In spite of the importance of concept of context, the most of context-awareness systems have a limited use of this concept. *According to [16] several works are limited to analyzing small quantities of contextual information and presents solutions to very specific needs. The first work, such as [7] considered only the user's localization and, still today, several systems are interested only in this aspect of the context [10] or in the aspects related to the use of the mobile devices [17].*

AHA! is a WIS, which supports user's mobility neither the user's localization nor the aspects related to the use of the mobile devices, but takes into account only user characteristics. The architecture of this system is detailed in the following paragraph.

Adaptive Hypermedia Architecture (AHA!)

AHA! is an Adaptive Hypermedia System based on the Web, it is inspired by the AHAM model. AHA! system is divided into components (Fig. 1) (for more details see [13]):

1) ***Java servlets*** are Java applications that enable enable dynamically generating the pages from the local file system or from external http servers.

2) The ***User Model (UM)*** contains the user's characteristics (identification, preferences, centre of interest, etc.).

3) The ***Domain Model (DM)*** helps to describe the hypermedia content and its organization.

4) The ***Adaptation Model (AM)*** describes how the adaptive hypermedia must make the personalization that represents the core of this system.

5) ***Authors*** typically create the domain/adaptation model through an authoring tool.

6) The ***Manager*** who configures AHA!, chooses the installation directory, path names, etc., and who creates accounts for authors.

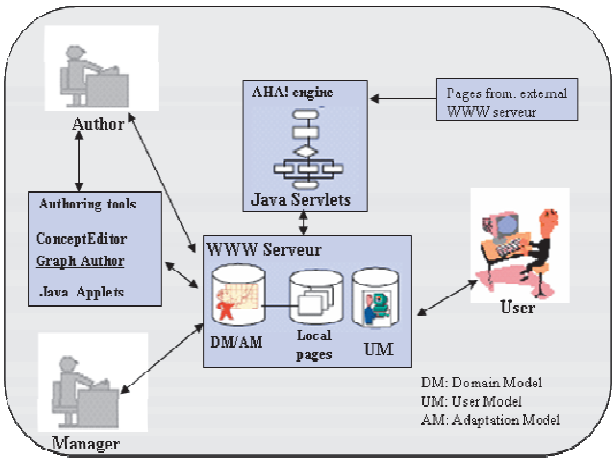

Fig. 1. Architecture of AHA!

This system is characterized by its simplicity. It carries out the adaptation of the presentation, the navigation and the content according to user's characteristics and preferences. But, since the user is not fixed in his office, it is obviously necessary to regard the user's context as a principal element for the adaptation. This context is presented by the user's dynamic and static characteristics and the characteristics of his environment (terminal, location, etc.). But, as we said, AHA! does not support the user's context. For that, *in our work, we will enable to this system to carry out the adaptation according to user's context while trying to use the context generally.*

B. Web services (WS)

Today there is an increasing need for making WIS component interoperable. So, WS appear to be the popular mechanism to support application interoperability. A WS is a software application which exists on the web accessible through networks. Traditionally, the architecture of WS (Fig. 2) is composed of three entities (provider, user, and registry) and is based on three standards (*WSDL* (Web Service Description Language) [20], *UDDI* (Universal Description, Discovery and integration) [19] and *SOAP* (Simple Object Access Protocol) [15]). The service provider builds the service and publishes its description in a registry. The user needs are translated into requests that are carried on by the WS registry. Once the service is found, the user will obtain direct interaction with the service [18].

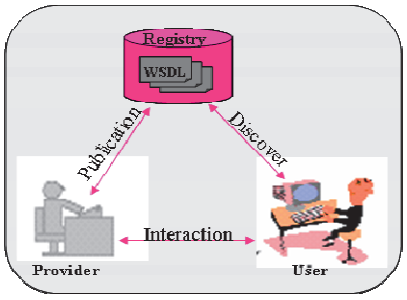

Fig. 2. Classical architecture of WS

Although the context becomes the ear and the eye of WIS and the use of WS within WIS is increasingly frequent, the classical architecture of WS does not consider context adaptation.

In order to integrate the adaptation concept in the WIS and provide the user with a list of services adapted to his context, we will integrate the classical architecture of WS with the architecture of AHA!, adding to these architectures an adaptation layer containing various components dedicated to the context adaptation.

III. New Vision of Web Information System

We note that the majority of proposed works in the area of adaptive WIS aim to build a WIS that is able to provide the user with relevant information to his context. However, these works are limited in analyzing small quantities of contextual information and present solutions at very specific needs. In addition, none of these works may return to the user the most WS adapted to his context.

The objective of our work is, therefore, to build an Adaptive Web Information System based on Web Services. To achieve this objective, we propose to extend the AHA! architecture in order to realize a general architecture of Adaptive Web Information Systems, where this system is based on the Web Service. This system allows, on one hand, to integrate a generic context model in AHA!, and on the other hand, to provide the user with a list of the most WS adapted to his context.

In this part, we illustrate our contribution, which aims to support the context adaptation and WS in WIS. We look, first, our architecture. Then, we show how the context is defined and represented in our architecture. Next, we detail the adaptation process. Finally, we will show our work through an example.

A. *Our architecture*

The next figure illustrates our proposal (for more details see [8]). In this architecture, the role of some components of AHA! does not change. For each component of the architecture AHA! we define a new component that supports its role in our architecture.

The ***user*** does not change. But in our architecture, the user can launch his query using not only a PC but can use another device types such as a mobile device or PDA.

The User Model (UM) is replaced, in our architecture, by the ***User Context***. This context includes information from the user's environment, which may influence the adaptation process.

The Domain Model (DM) is replaced, in our architecture, by the ***Web Services context***. This context includes information on the environment service, for example, with what kind of terminal we can display services.

The author plays the role of ***service provider*** that builds and publishes his description in the Registry and which will be responsible for providing the context of each WS and stores its in DM.

The role of (AM) is played by the ***Answer Generator*** which manages the adaptation process. This process aims to provide the user with a list of the most WS adapted to his context.

The local and external pages are replaced by the **WS** provided by several WIS.

So our architecture is composed of three layers (Fig. 3):

1) ***The Registry layer*** that corresponds to a registry of service description offering facilities that publish services for providers and that searches services for users. The provider was presented by the WIS where each WIS has a WS. The user will launch his query to the Registry; a list of services adapted to his query will be forwarded to the second layer.

2) **The Answer Generator layer** is responsible for managing the adaptation process and generates the final answer (the list of WS adapted to the user's context) to the user.

3) ***The Context layer*** is responsible for capturing and managing of user's and services context.

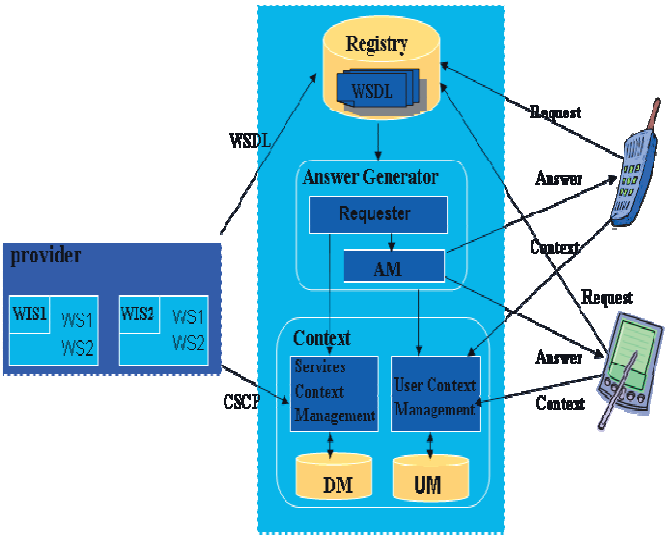

Fig. 3. Architecture of WIS with context and WS

The *Fundamental layer* in our architecture is the Answer Generator which is responsible for managing the adaptation process. This layer is composed of two components (Fig. 4).

1) The first "***Requester***" asks the Registry the list of services adapted to the user's query. Then he contacted the Services Context Management for request services contexts found in the list returned to the Registry.

2) The second "***Adaptation Model***" (AM) is responsible for the adaptation process. AM performs a process of "matching" between the user's context and the context of each service by comparing the attributes of their metadata. The list of WS returned to the user will be ordered according to the degree of similarity calculated in the matching.

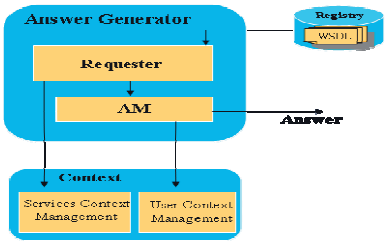

Fig. 4. Answer Generator

To restore the list of the most services adapted to context, we need to define the necessary means to capture and store the user's context and the services context. The Context layer is responsible for managing this process. It is composed of module of the User Context Management and the Services Context Management (for more de details see [8]).

1) The first one called "***User Context Management***" is responsible for capturing the user's context and storing it in a database (UM).

2) The second called "***Services Context Management***" is responsible for the extraction of contexts of Web Services, the storage of these contexts in a database (Domain Model (DM)), and restitution to The Answer Generator contexts of services that meet the needs of the user. In this component, the service context is expressed directly by the service provider.

B. Context modeling

Definition and representation the contextual user profile

In our domain, the definition of Dey is the most accepted by the majority of researchers. But Chaari [16] considers that this definition may be a source of conflict because it does not help in separating the contextual data from the application data. To provide more precision in Dey's definition, Chaari defines the context as: "the set of the external parameters that can influence the behavior of the application by defining new views on its data and its available services. These parameters may be dynamic and may change during the execution".

Based in this definition, we will define four principal elements of the context: the ***user characteristics*** which are composed of *static characteristics* (langue, first name, etc) and of *evolutive characteristics* defined by his *environment* (his localization, time, etc) and his *preferences*, the ***device characteristics*** (hardware, software), the ***network characteristics*** (type, bandwidth) and the ***session characteristics*** (date, hour, etc).

Fig. 5 presents the general structure of our context model. For each user session we associate a profile "Context-Profile" which describes the four elements of the context "Context-Element". Each "Context-Element" may contain a "Context-Sub-Element". Each "Context-Sub-Element" may contain another "Context-Sub-Element" or a "Context-Attribute". The contextual situation is defined by the values associated with Context-Sub-Element of the context. The modification of one of these values corresponds to a transition to another contextual situation. Each Sub-Element of the context can be static if it does not change in a user session or dynamic if its value can vary in the same user session. That's why we associate for each Sub-element, an attribute "Type" to differentiate dynamic elements from static ones. This differentiation is very important for the adaptation process.

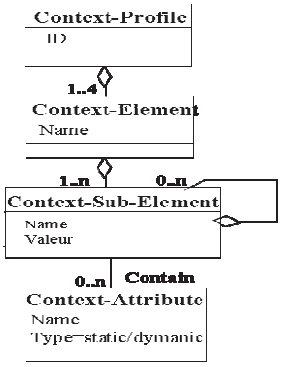

Fig.5. General structure of our context model

In order to return to the user the list of the most services relevant to his context, we must take into account not only the user's context but also the services context. In our architecture, we define the service context by the four elements identified before to the user.

To represent the contextual information, the majority of researchers used the format CC/PP [9]. CC/PP is a decomposable, uniform and extensible standard but we did not use it to represent our context because of its lack of structured functionalities where its two leveled strict hierarchy is not appropriate to capture structures of complex profiles [12].

We use the CSCP format to overcome the structure deficits of CC/PP. CSCP provide a multileveled structure which allows the representation of all the types of contextual information [2]. So, we stock all the contextual information in an XML document based on this CSCP model (Fig. 8). Each element of this document presents a context element.

An algorithm for constructing User-Context-Profile (UCP)

To solve the conflict problem between the user's preferences and his context, we use an algorithm that realizes a filtering process in order to build a general profile of the user's context "User-Context-Profile". This algorithm analyzes each user preference to assess whether this preference can be satisfied with the context or not. This context allows the system to select only the preferences that are compatible with it. For example, among the preferences that concern the display type of query result; we will only retain the preferences which can correspond with access device characteristics.

An algorithm constructing User -Context-Profile (UCP)

(1) **User-Context-Profile (UCP)**
(2) begin
(3) user identification
(4) Save (A)
(5) if the first connexion then

```
(6)        Save (B)
(7)     Save (C)
(8)          enter User Preferences (UP)
(9)          Function-Preferences (First)
(10) else
(11)        Save (C)
(12)        enter User Preferences (UP)
(13)        Function-Preferences (Second)
(14) end if
(15) end
```

```
(1) Save (A)
(2) capture the session characteristics (SC) and the environment characteristics (EC)
(3) save SC and EC in the document (UPC)
(1) Save (B)
(2) enter static characteristics (STC) // name, surname, , etc.
(3) save STC in the document XML (UPC
(1) Save (C)
(2) enter the device characteristics (DC) and the network characteristics (NC)
(3) save DC and NC in the document (UPC)
```

(A) In this algorithm, if the user connects to the system for the first time. He must enter SC, DC, NC and their preferences. Then, for each one of these preferences Pi, we apply the function Function-Preferences (First) to determine if this preference can be added to the UCP (cf. lines 4, 5) au to LPNS (cf. line 7).

Algorithm that concerns the Function-Preferences (First)

```
(1) Function-Preferences (First)
(2) i=0
(3) while (i<=n)   //   0 <= i <= n  the number of user preferences
(4) if (Pi can be satisfied) then
(5)     add Pi in  UCP
(6) else
(7)  add in LPNS // LPNS content the list of preferences which can't be satisfied
(8) end if
(9) i=i+1
(10) end while
```

(B) If this is not the first connexion of the user. He must only enter DC, NC and their preferences. Then, for each one of these preferences, we apply the function Function-Preferences (Second). This function realizes the following steps:

1) If this preference can be satisfied with the context, and if this preference is found in LPNS having the same value, we must remove it from LPNS and add it in the UCP. Otherwise, if the preference can be satisfied with the context, and if this preference is found in LPNS with other values, we must know whether this preference with the different value is satisfied with the context or not. If yes, we add this value in the UCP (cf. lines 5, 6, 7, 8, 9, 10, 11, 12, 13, 14).

2) If this preference is not satisfied with the context, and if this preference is found in LPNS with other values, we must know whether this preference with the different value is satisfied with the context or not. If yes, we add this value in the UCP. Otherwise, we must delete it (cf. lines 16, 17, 18, 19, 20, 21, 22, 23).

Algorithm that concerns the Function-Preferences (Second)

```
(1) Function-Preferences (Second)
(2) i=0
(3) while (i<=n)  //  i= 0, …n
(4) case
(5)  if (Pi can be satisfied) then
(6)     if   (Pi is in LPNS having the same value) then
(7)              delete Pi from LPNS
(8)           else
(9)                 if (Pi can be satisfied) then
(10)                 delete Pi form LPNS
(11)              end if
(12)    end if
(13)             add Pi in UCP
(14)  end if
(15)   otherwise
(16)         if  (Pi is in LPNS having other value) then
(17)                  if  (Pi can be satisfied) then
(18)               delete Pi from LPNS
(19)                    end if
(20)                 add Pi in UCP
(21)             else
(22)               add Pi in LPNS
(23)    end if
(24)  end case
(25) i = i+1
(26) end while
```

If a preference in our UCP has two or more value, we give priority to the value entered by the user during the current session. If the user in the current session has not defined preferences, the system considers the history of the user, i.e. previous sessions.

C. Adaptation process

In order to realize the adaptation process, we present the two profiles in the form of trees. The tree that represents the user's profile must be weighed, i.e. weights are associated with nodes of the tree. The weight represents the degree of importance of the information associated with nodes. Its value is given directly by the user when he launches his query. The user assigns each element a real number between 0 and 1: 1 being the criteria for adaptation priority and 0 as the default, designating a criterion of adaptation without special hierarchy. As an example, suppose a user wants to travel from Toulouse to Paris. He wants to know price of the train ticket going from the nearest station to his location with importance 0.6. He hoped that the service responds in English with importance 0.4. While the answer in English is less important than the location. Then, we perform a matching process between the two contexts (user, services) by comparing the content of contexts elements and by calculating a degree of similarity between the two. In order to calculate the degree of similarity between the profiles, we use measures of similarity proposed by Alilaouar [1] on our team and presented by the following equation:

$$w(T) = \left(\sum_{i=1}^{n} \frac{niv(u_i) + w(u_i)}{niv(u_i) + 1} \, w(T[u_i]) \right) \Big/ n$$

T: context tree, n: the number of nodes in the tree, ui: child's of the root, niv (ui)=1,...M : the number of the level where we find le node ui, W(ui) : the weight of the node ui.

With this equation, the weight of a tree is calculated progressively starting with the weight of the leaf nodes (tree to a single node) and going up to the root of the tree.

The list of Web Services returned to the user by the Answer

Generator will be ordered according to the degree of similarity which was calculated in the matching. So the most appropriate Web Service will be found at the top of the list.

D. An example

A user wants to travel from Paris to Bordeaux. He wants to know price of the train ticket going from the nearest station to his location. The user also prefers that the service would answer in English and be able to adapt to his device. The user will launch his query to the Registry asking a service for reserving the cheapest train ticket from Paris to Bordeaux. At that time, User Context Manager captures explicitly his preferences via an interface that allows him to express his preferences and the priorities associated with them. The service will answer the user query in English and would provide him with the nearest station of his departure. The user precise the weight (0.4) for the first attribute "Language" and the weight (0.6) for the second attribute “place”. The User Context Manager captures also the information concerning the device type and the network implicitly. Then, it stores all this information in an XML document in the database UM (Fig.6).

```
< ?xml version= "1.0" ?>
<rdf :RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:cc=p="http://content-www.org/CSCP/CSCPProfileSyntax#"
   ...>
<ContextProfile rdf::ID=Context>
<User>
  <use :UserProfile>
    <use :CharStatic> . . </use :CharStatic>
      <use :CharDynamic>
        <use :Environment>
           <use :Location weight ="0.6"> Near the care station </use :Location>
           <use :Time>18 h</use :Time> ..
        </use :Environment>
        <use :Preference>
          <use:Language>
            <ref:Bag> <use:li weight="0.4"> English</use:li> </ref:Bag>
          </use:Language>
                        ...
          </use :Preference>
      </use :CharDynamic>
  </use :UseProfile>
</User> ...
</ContextProfile >
</rdf:RDF>
```

Figi.6. An example of CSCP profile

In order to make the filtering process the Answer Generator calculates a score of correspondence between the descriptions of user’s context and services context by taking into account the priorities expressed by the user. In this example, the Generator gives priority to the attribute place because it has a higher weight. The list of WS returned to the user by the Answer Generator will be ordered according to the degree of similarity calculated in the matching. Where the most WS adapted will be placed at the top of the list of services.

IV. Conclusion

In this article, we propose a simple and general architecture that enables, on one hand, to support the adaptation process according to the user’s context and on the other hand, to provide the user with a list of Web Services relevant to his context. So, our aim is to build an Adaptive Web Information System based on Web Services where the user can reach the list of the most relevant services to his query and his context. Then, we presented an algorithm that aims to build a user contextual profile “Profile-Context-User (UCP)” containing only the user’s preferences that can be satisfied with the context. Next, we mentioned the adaptation process that performs a matching process between the two contexts (user, services). Finally, we illustrated our work by an example.

In the near future, we envisage to implement our proposal and to specify the service context more precisely and to use ontology for Web Services search as WSDL-S since the UDDI registry supports only keyword matching.

References

[1] A. Alilaouar, “Contribution à l’interrogation flexible de données semi-structurées,”. PHD Thesis, Paul Sabatier University of Toulouse, IRIT, 2007.

[2] A. Held, S. Bouchholz, and A. Shill, “Modeling of Context for Pervasive Computing Application,” In Proceedings of the 16th world Multiconference on Systemics, Cybernetics and Informatics (SCI). Orlando, FL, USA, 2002.

[3] A. Isakowitz, M. Bieber and F. Vitali, “Web information systems,” Communications of the ACM vol.41, pp.78-80, July 1998.

[4] A. Jameson, “Modeling both the context and the user,” Personal and Ubiquitous Computing Journal, 2001, pp. 29–33.

[5] A.K. Dey, “Providing Architectural Support for Building Context-Aware Applications,” PhD Thesis, Georgia Institute of Technology, 2000.

[6] B.N. Schilit, and M.M. Theimer, “Disseminating active map information to mobile hosts,” in IEEE Network, vol. 8, n°5, 1994, pp. 22-32.

[7] B.N. Schilit, N. Adams, and R. Want, “Context-Aware Computing Applications,” Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, USA, December 1994, pp. 85-90.

[8] B. Soukkarieh, and F. Sedes, “Towards an Adaptive Web Information System Based on Web Services,” The Fourth International Conference on Autonomic and Autonomous Systems ICAS 2008, Gosier , Guadeloupe, March 16-21, 2008.

[9] CC/PP, W3C, Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0. W3C Recommendation 15 January 2004. The last version: http://www.w3.org/TR/CCPP-struct-vocab/ (Janvier 2007).

[10] H.K. Rubinsztejn, M.Endler, V. Sacramento, K. Gonçalvez, and F. Nascimento, “Support for context-aware collaboration,” in First International Workshop on Mobility Aware Technologies and Applications - MATA 2004, Florianopolis, Brazil, Springer-Verlag, 2004, pp.37-47.

[11] K. Cheverest, K. Mitchell, and N. Davies, “The role of adaptive hypermedia in a context-aware tourist guide,” in Communication of ACM, vol. 45, n° 5, Mars 2002, ACM Press, pp. 47-51.

[12] K. Mostéfaoui, J. Pasquier-Rocha, and P. Brézillon, “Context-aware computing: a guide for the pervasive computing community,” Proceedings of the IEEE/ACS International Conference on Pervasive Services (IPCS’04), IEEE Computer Society, 2004, pp. 39-48.

[13] P. De Bra, G. J. Houben, and H. Wu, “Aham : A dexter-based reference model for adaptive hypermedia,” in tenth ACM conference on hypertext and hypermedia, 1999, pp. 147-156.

[14] P. De Bra, A. Aerts, B. Berden, B. De Lange, B. Rousseau, T. Santic, D. Smits, and N. Stash, “AHA ! The Adaptive Hypermedia Architecture,” HT’03, Nottingham, United Kingdom, August 26–30, 2003, pp 81-84.

[15] SOAP, W3C, Recommandation W3C 24 Juin 2003. The last version: http://www.w3.org/TR/soap12-part0/ (December 2006).

[16] T. Chaari, F. Laforest, and A. Flory, “Adaptation des applications au contexte en utilisant les services Web, ” Second Francophone workshop: Mobility and Ubiquity, Grenoble, France, 31 May- 3 June 2005.

[17] T. Lemlouma, “Architecture de négociation et d'adaptation de Services Multimédia dans des Environnements Hétérogènes,” PHD, Institut National Polytechnique de Grenoble, Grenoble, France, April 2004.

[18] T. Melliti, “Interopérabilité des Services Web complexes. Application aux systems multi-agents,” PHD, Paris IX Dauphine University, 2004.

[19] UDDI, W3Schools, http://www.w3schools.com/wsdl/wsdl_uddi.asp (December 2006).

[20] WSDL, W3C, Web Services Description Language (WSDL), the last version: http://www.w3.org/TR/wsdl. (December 2006).