An Ontology-Based Context Model to Manage Users Preferences And Conflicts

Salima Bourougaa Tria
Tebessa university, Tebessa, Algéria
E-mail: bourougaasalima@gmail.com

Hassina Seridi-Bouchelaghem
Annaba University, Algéria
E-mail: h_seridi@yahoo.com

Farid Mokhati
Oum-el-bouaghi university
E-mail: mokhati@yahoo.fr

Keywords: context of use, user profile, web-based information system, nomadic environment, ontology, user preferences, conflicts, ubiquitous computing, web service

Received: April 22, 2015

In the last decade, ubiquitous computing (UC) has become an aspiration of the computing community. Nowadays, it is so profound that it is increasingly indistinguishable from the overall agenda of computing research. In UC, the main objective is to provide users the ability to access services and resources anytime, anywhere, in particular using Mobile Devices (MD). Applications in this domain are sensitive to the context. They have to be able to perceive this context and to adapt their behaviours by considering data that deals with the context of use and user preferences. Actually, ensuring access by nomadic users to information Systems through various devices and the adaptation of responses to nomadic users profile and context of use are two bound problems. In this paper, we attempt to answer to these problems and we propose a novel approach allowing essentially: (1) representing the context and preferences of nomadic users through ontology, to support context representation and reasoning (2) resolving conflicts that may arise between user preferences and, (3) adapting such applications to the context of use and user's profile. The approach is supported by a visual tool we developed. A case study is presented to give more illustration.

Povzetek: Opisana je uporaba konteksta s pomočjo ontologije za preference in profile mobilnih uporabnikov.

1 Introduction

Currently, Web users access to a large mass of various data situations through distinct devices, to have answers to their requests that are usually very numerous, from multiple sources of information (heterogeneous and remote). Such answers are not all equally interesting and relevant, and they do not answer all the user's wishes, which may decrease the user satisfaction. This complexity is increased if the user is nomadic (user who frequently changes localization) and appealed the SIW (System Information on the Web), anywhere and anytime via mobile devices (PDAs, phones, laptops) because the change of localization, for example, causes a change in working conditions and consequently a change in the general context of use. Consequently, developers are incited to integrate these mobiles devices into their applications, giving rise to new information systems called pervasive or ubiquitous [1]. In this case, these applications must considering the user's situation called contextual situation. This latter includes the context of use as well as information on its profile. Adapting all application's behaviors, in order to return to users relevant responses (i.e. while considering content and time), is the subjacent idea of ubiquitous computing, where applications are sensitive to the context (context-aware applications) [2].

Actually, ensuring access by nomadic users to information Systems through various devices and the adaptation of responses to nomadic users profile and context of use are two bound problems. Dealing with these problems requires answers to the following questions:

- How to perceive the user's context?
- How to model the context of use and the nomadic user profile?
- How to resolve conflicts that may arise between user's preferences?
- How to adapt the context-aware application behavior to satisfy the needs of these mobile users?

In order to answer these questions, we propose, in this paper, a novel approach which essentially allows: (1) to model the context of use and the user's preferences using a developed ontology "Contology", basing on a new definition of the context which separates application data from contextual data. The ontology is useful to support context representation and reasoning, and the Dynamic requirements can be defined as context constraints and need to be supported by context reasoning features of the ontology, and they are most expressive and most promising for context description in an environment sensitive to the context. (2) To resolve conflicts that may occur when managing user's preferences, we propose to model conflicts and their solution in the ontology as rules by using the semantic web rule language (SWRL). Finally, to ensure the dynamic functional adaptation of context-aware applications, Web Service based architecture is proposed to show the effectiveness of our proposal in the context model.

The remainder of this paper is organized as follows: In Section 2, we give a brief overview of major related works. Section 3 outlines the motivation for using ontology, while section 4 presents the context model and the conflict management. We explain the ontology process building in Section 5. Section 6 details the context rules description and the ontology implementation is given in section 7. Section 8 details the adaptation process (ontology exploitation). We present a case study in section 9. Finally, we discuss our actual research, draw some conclusions and give some future work directions.

2 Related works

We distinguish four categories of context modeling approaches. The first category consists in storing the context by using key-value pairs (attribute, value) or by using a set of triplets. Famous examples of this category are: Context Toolkit of [3] and approaches used by [4] .The second category of the model-oriented approaches includes: (1) Markup Scheme Models: represent the context by using RDF. For example: CC/PP [5], [6] and ConteXtML [7], (2) Graphical Models: use UML (Unified Modeling Language) to model the context. For example: ContextUML [8] and CML [9], (3) Object Oriented Models use principal advantages of the modeling object. For example: Active Object Model [10] and the TEA project CUES [11]. The third category represents the context by a logic-based model. The context is defined like facts, expressions and rules. An early representative of this approach type is: ' Extended Situation Theory' [12], [13] and [14]. (4) The last category models the context by using ontologies. The most referred modeling are: CoOL [15] , SOUPA [16] , [17] a formal context model based on ontology using OWL to address issues including semantic context representation, context reasoning and knowledge sharing, context classification, context dependency and quality of context , [18] and [19] COBRA-ONT an ontology to support pervasive context-aware systems. COBRA-ONT, expressed in the Web Ontology Language OWL, is a collection of ontologies to describe places, agents and events and their associated properties in an intelligent meeting-room domain. [20] an intelligent web portal to serve as a service provider in the airlines travelling tasks , [21] a metadata model encoding semantic tourism destination information in an RDF-based P2P network architecture. The model combines ontological structures with information for tourism destinations and peers, [22] an approach based on ontologies provide the elements and guidelines to define and create a user profile in any multimedia domain. In order to describe the multimedia context and ontologies of PUMAS a framework based on the agents [23], [14] and [1].

In [24] and [6], we find a synthesis on the characteristics of the context modeling approaches and this let us deduce that in spite of the principal disadvantage of the ontology approaches which is the ontology's complexity execution and the reasoning weight on their facts and their entities. They are most expressive and most promising for context description in an environment sensitive to the context. This is our motivation to choose ontology in context modeling in this work. Those works have considerably forwarded the domain by proposing novel strategies to context modeling. However, they omit some important aspects which can be summarized as follows: firstly, none of existing ontologies of context modeling separate between the context data and the applications data. According to [25] and [6], this separation is very necessary to a reliable modeling of context. Also, the user's preferences management was only considered by PUMAS [23], [14] and [1]. Although, it represents a very important point to satisfy the user and to return him answer adapted to its context. Finally, the conflict's resolution is considered only by PUMAS [23]. It defined some conflicts and presents their corresponding solutions. But this approach does not solve this problem, because it has not considered all conflicts which can arise during the user's preferences checking.

3 Motivations for using ontology

The main goal of the proposed approach is to model the context of the user by use of a semantic representation and resolve conflicts that may arise during these preferences verification. This proposed context modeling objective is to adapt the initial request of use to this context, to have a contextual query, used to give to user a response adapted to his context. We opted, in the context of this work, for the use of ontologies for the advantages they procure. They provide the means to describe semantically information, share described data, easily to be used by other applications and to extend the initial description when new needs arise. Ontology languages can create expressive, scalable, reusable, sharable models, and on which we can reason using inference engine. OWL [26] for example, is a W3C recommended language to describe ontologies. It provides a simple and effective means based on an XML description model to

share described data, reasoning about these data and adding axioms to describe specific relationships between information. Finally, ontologies are most expressive and most promising to context description in an environment sensitive to the context [24], [1].

In existing context-aware systems, notations like XML, XMbased CC/PP [27], UML [28], Topic Maps [29] and OWL [30], [31] are used in context modeling. We use the OWL to formalize context relationships based on the underlying DL representation. The choice of OWL is motivated by its reasoning support. It provides a logical language support to reasoning (OWL-DL) and supports Semantic Web Rule Language (SWRL) to enable rule-based reasoning [1]. The logical language (DL) supports context composition and context constraints enhancements. OWL facilitates the sharing of conceptualizations. The core elements of the DL used as an underlying abstract language shall be introduced. The Attributive Language with Complements (ALC) is the basis of many DL languages. The OWL-DL, the DL variant of OWL corresponds to SHOIN(D) [32], a DL language based on ALC with transitive roles, role hierarchies, nominals (enumerated classes of object value restrictions), inverse properties, cardinality restrictions and concrete data types[1]. In order to encode context aspects in SHOIN(D), and eventually in OWL-DL, an introduction of the constructors for SHOIN(D) is necessary. Their semantics is based on the usual interpretations of first-order logic. C denotes concepts, and R denotes property relationships. A DL specification can be constructed as a set of axioms. The basic constructors of SHOIN(D) can be used with either the subsumption or equivalence ≡ symbols to create DL statements. Axioms can be terminological axioms (TBox) or assertional axioms (ABox). Terminological axioms (statements about entities such as concepts and roles, but not individuals) can be subsumption or equivalence axioms. Assertional axioms (pertain only to individuals) can be concept assertions or role assertions axioms. A Subsumption axiom gives necessary conditions for some a concept tobe included (Subclassed) in another, e.g. A $\sqsubseteq$B where A, B are concepts. An equality axiom has the form A≡ B. A concept assertion is of the form C(i) where C is a concept from a TBox and i is an individual. A role assertion is of the form R(a, b), where R is some role from a TBox and a and b are individuals.

4 The Context model representation

We will describe how we can define the context concepts. For the development of our Context Ontology "Contology", we used "METHONTOLOGY" [33]. According to [33], it is important to bear in mind that knowledge acquisition is an independent activity in the ontology development process. However, it is coincident with other activities. Most of the acquisition is done simultaneously with the requirements specification phase, and decreases as the ontology development process moves forward. Experts, books, handbooks, figures, tables and even other ontologies are sources of context from which the context can be elucidated using in conjunction techniques such as: brainstorming, interviews, formal and informal analysis of texts, and knowledge acquisition tools. In our approach the knowledge is the context of the user. The used techniques in the Context acquisition are: (1) Non-structured interviews with experts, to build a preliminary draft of the requirements specification document. (2) Informal text analysis, to study the main concepts given in books and handbooks. This study enables to fill in the set of intermediate representations of the conceptualization. (3) Formal text analysis. The first thing to do is to identify the structures to be detected (definition, affirmation, etc.) and the kind of knowledge contributed by each one (concepts, attributes, values, and relationships). (4) Structured interviews with experts to get specific and detailed knowledge about concepts, their properties and their relationships, to evaluate the conceptual model once the conceptualization activity has been finished, and to evaluate implementation. (5) All given definitions of context given by researchers and experts of context-awareness domain.

4.1 The context definition

Researches in the context-awareness domain have not yet led to a generic and pragmatic definition of context. Several definitions for the context were advanced [34], [35], [36], [25], [6]and [1].The definitions issued so far are very abstract or very specific to a particular domain, making the formalization of the context very difficult. The [3] definition is widely accepted as a "good" definition. According to [25], this definition does not help in separating the contextual data from the application data, and the core of the application should be designed in a context in dependent way. This separation separating the contextual data from the application data, and the core of the application should be designed in a context in dependent way. This separation according to [25] is very important, before beginning the design of an application sensitive to the context. A data defined as contextual in a field can be a data application in another field. For example, GPS localization is part of application data in a traffic regulation system, but is part of context data in a telemedicine application. Separation between the contextual data and the application data is also important in modeling context. [25] define 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". Consequently, in the determination of the most descriptive concepts of information which constitutes the context, we chose the separation of the contextual data of the application data according to the definition of [25] of the context, because it seems to us relevant and generic. According to this definition, we can divide the concepts of context into two parts: the concepts which represent the context of use of a user and the concepts which represent the user profile. The context of use in our approach presents the set of

data which allows indicating the situation of the user when it connects to the ubiquitous application. For example, it is represented by the following concepts: The user; the session; the used mobile device (MD) and location of the user. The user profile is presented by a set of preferences of user. We detailed these concepts in the following sections.

4.2 The Context representation: preferences, conflicts

Among the concepts of the user's context, we find the preferences. In this part we will define the concept of preference of the user and we detail a classification of different types of preferences. We will detail the concept of conflict and we will present its causes and solutions.

4.2.1 Preferences

By the concept of user preference, we refer to a set of descriptions covering what the user likes to receive as services, also the display of results choice. We define two types of preferences: Requested Service Preferences and Display Preferences and five conflicts.

a. Requested Service Preferences

Describe how the user chooses its services in the system. We define this type of preferences as follows: During his first contact with our system, the user can define the contents of each of his preferred services. The user can define from the beginning when he asks the service "S" what implies automatically the contents: C1, C2 ....Etc.

Service (S) →contents (C1, C2,……. etc.)

As example to illustrate our proposition let us consider a user in travel who wants to have the list of restaurants in his entourage. He prefers that this list is displayed as a map. His user profile can, for example, specify that when it executes the service "consultation list of restaurants. » this user is only interested in restaurants offering dishes which respect his diet, because he has health problems. Thus, the preference says that user wants to execute the service "S" = "consultation list of restaurants" whose content is C = "restaurants that offer adequate food.", and preferably in the form of display image. Therefore, the preference of requested service is represented as follows:

Requested_Service_Preferences(S, {content},
{ associated_ Requested_Services**}).**

S: is the service which the user wishes to carry out in the system. ***{Content}:*** is a list of the contents defined by the user from his first contact with our system. ***{Associated_Requested_Service}*** is a list of the associated services which the user wants to execute if he asks the service S. As example, we consider that a user wishes to execute service "S" which consists of one or several *contents* and possess one or several *associated_ RequestedServices*. Every time a teacher consults "the list of the planned meetings ", he wishes to know the meetings of the current week. Also, he executes associated_RequestedServices "possibility meeting", to see the possibilities of fixing a meeting between teachers by specifying the day, the hour and the list of the concerned teachers, and the associated_ Requesed_Services "the other possible dates " to know all the possible dates of meeting of one or several teachers (days and hours free).We can represent the data: Requested_Service_Preferences as follows:

S1 = Possibility meeting (list of teaching concerned, day, hour). S2 = the other possible dates (free day, free hours, list teachers). C1= meetings of the current week. Then, the Requested_Service_Preferences is presented as follows:

Requested_Service_ Preference (S: "the list of the planned meetings ", {S1, S2}, {C1})

In the following, we present the display preferences.

b. Display Preferences

Display Preferences describe how the user wants the information to be displayed on his MD (for example, the user only wants information in text format). At every service is associated a Display preference. It is represented as follows:

Display _Preference (format, characteristics)

Format which can take the value: "video", "text", "image", "sound". Each format is based on a set of characteristics. Following sections, detail the conflict in our approach, present their causes and details there solutions.

4.2.2 Conflict

By conflict we refer to problems which can arise during the verification of user preferences. For example, "Contradiction between the display preferences and the characteristics of used MD", this conflict can arise when user requests a display which is not supported by his used MD. For these problems (conflicts) that we will define later, we offer some solutions to solve them. At every type of conflict is associated a solution. It is represented as follows:

Conflict (Type, Solution, Suggestion)

Type: represent the conflict which can arise. ***Solution:*** allows defining how to take action to resolve the conflict that occurred. ***Suggestion:*** represents the proposal of the user in cases where the system cannot find a solution to the conflict that occurred.

Our approach manages five conflicts which can be arising between the user preferences during the check of these last ones. The following two tables present our proposal to conflicts resolution. Table1 presents the conflicts and their causes, while Table2 presents the conflicts and their solutions in our proposal.

N° Conflict	**Conflict**	**Cause**
1	a. Contradiction between TheRequested_Service_Preferences and access rights of the user	• The user requests a service which does not suit with these access rights.
2	b. Contradiction between the display preferences and the characteristics of used MD	• The user requests a display which is not supported by his used MD.
3	c. Various wishes of Display for the same service.	• This conflict can arise in two cases: **a.** The user did not specify display preferences. **b.** Display preferences are not suitable to the characteristics of MD. In these two cases the system will returns to the Context Ontology "Contology" for resolve it.
4	d. Absence of display preferences after checking the historic of the user.	• The user cannot specify display preferences, in this case the system will return to the historic of the user, and it cannot find display preference for favorite service.
5	e. Contradiction between the Display preferences requested and display capabilities expressed	• The user can request the service in a format not offered by the system. For example, if the user wants a list of restaurants in card format, while the system has this information in text format only.

Table 1: Conflicts and Causes.

Conflict	Solution
1	• The system returns to the user to inform him that he has not the right to access these services and asks consequently, suggestions for this problem. If the user does not give suggestions, the system stops.
2	• Our approach execute one of the following cases: **a.** Uses the ontology "Contology" for searching and reasoning about a solution for the conflict, using the information of the precedents sessions, to extract the display preferences that agrees with the characteristics of the used MD. **b**. if no, Returns to the user and demands suggestions.**c**. if no in the 2 previous alternative, he takes a default display preference which suits with the characteristics of the used MD.
3	• We propose using an arithmetic operation that gives us the number of specification of every encountered preference. The system will perform a comparison and it will retain the preference which has the maximum number of specification by the user. In the case of equality between preferences, we propose to use a default preference which suits with the characteristics of MD used.
4	• The system executes one of the following cases: **a**. It returns to the user and asks for these suggestions, **b**. It uses a default preference.
5	• In this case the system executes one of the following cases: **a**. Uses the ontology "Contology" for searching and reasoning about a solution for the conflict, using the information of the precedents sessions, to extract the display preferences that agrees with the characteristics of the used MD.**b**. it returns to the user and asks these suggestions, **c**. if no in the 2 previous alternative, he takes a default display preference which suits with the characteristics of the used MD.

Table 2: Conflicts and Solutions.

After detailing the context acquisition, defining what means context in our work, and presetting the context representation. In the following section we will present the ontology process building based on the method "METHONDOLOGY".

5 Ontology process building

This section presents the steps followed to build the ontology of context "ContoLogy", for this, we use a construction process in the development of the ontology

starting from raw knowledge and arriving at an operational ontology represented by OWL. The main steps of this process are based on the methodology of ontology construction "METHONTOLOGY" [33] which is the basic support for the conceptualization of the ontology to create, through a series of semi-formal intermediate representations. The logic descriptions, is the used formalism to express the semi-formal ontology. OWL language for defining ontologies is chosen to codify the ontology using the Protégé OWL ontology editor. Finally, the inference RACER (Renamed Abox and Concept Expression Reasoner) system is used to test the consistency of the ontology throughout the development process. This process consists of five steps: (1) Specification of Requirements, (2) Conceptualization, (3) Formalization., (4) Ontology implementation, (5) Test & evolution of ontology. We start this part by the motivation of the build method choice. Then, we detail the steps process.

5.1 Ontology method build choice

Born of the needs of knowledge representation, ontologies are currently at the center of the research in knowledge engineering. The construction of ontology requires both a study of human knowledge and the definition of representation languages and the realization of systems to handle them. The knowledge engineering has given birth to the ontological engineering, where the ontology is the key item that needs to be addressed. Several studies propose methods of constructing ontologies. In this case, we have study some methods for creating ontologies such as: ENTERPRISE [37], TOVE [38] and METHONTOLOGY [33] and we present a comparative study in order to choose a method.Table3 summarizes the comparable study on the various methodologies and methods. Each cell in of the table may be filled with three types of values. Value "++" means that the method or methodology describes how to execute each task in the proposed activity (specification, conceptualization….)? When to do? Who should do it? ... Etc. The value "+" means that the just methodology identifies the process. The value "-" means that public documentation does not mention the activity.

"METHONTOLOGY" is the approach that provides the most precise descriptions of each activity. Most approaches are carried on of the activities of development, particularly on the implementation of the ontology, and they not interested in furthering other important aspects related to the management, development and evaluation of ontologies. This is because the field of conception of ontology is a relatively new field. However, low conformity with the formally established criteria does not mean poor quality methodology or method. The most approaches have drawbacks. According to table2, we choose "METHONTOLOGY" for the construction of our Context Ontology.

Criterias of comparison	**TOVE**	**ENTER-EPRISE**	**METHO-NTOLOGY**	**OTK**
Specification	++	+	++	++
Acquisition of knowledge	+	+	++	++
Conceptualisati-on	++	-	++	+
Formalisation	++	-	++	++
Evaluation	+	+	++	+
supports tools	specifi c tools	specific tools	ODE, WebODE,Pro tégé-2000	OntoE-dit

Table 3: Comparison of methods for developing ontologies [39] [40].

5.2 Specification and Requirements.

The goal of the specification phase is to produce either an informal, semi-formal or formal ontology specification document written in natural language, using a set of intermediate representations or using competency questions, respectively. See figure1

ONTOLOGY REQUIREMENT SPECIFICATION DOCUMENT
Domain : context-aware application (Ubiquitous applications) **Date** : January, 15th 2014 **Conceptualized-by** : authors **Implemented-by**: authors **Purpose**: Context modeling ontology in context-aware applications to be used by our architecture of adaptation based on Web service. **Level of Formality**: Semi- formal. **Scope**: List of 33elements of substances: **List of concepts** : ContextModel, ApplicationContext, ServicesApplication, ConflictContext……..etc **At least information about the following properties:**IsConceredBy, HasSugg, AttachedTo, CausedBy, OccuredIn, **Sourcesofknowledge**: Definitions of the context in the domain of context-aware applications.

Figure 1: Ontology Requirements Specification.

5.3 Conceptualization

In this step, we will structure the domain knowledge in a conceptual model that describes the problem and its solution in terms of the domain vocabulary identified in the ontology specification activity [Fernandez, 1997]. This phase comprises several stages which are: the Construction of: (1)Terms glossary,(2)Concepts classification diagram, (3)Binary relations diagram,(4)Dictionary concepts, (5)Tables of binary relations, (6) Attributes table, (7)Logical axioms table, (8) Instances Table.

a) Construction of Terms Glossary:

This glossary contains the definition of all the terms relating to the field (concepts, instances, attributes,

relations) which will be represented in final ontology, we have **128** terms, for example: UserContext and ContextModel are concepts, PreferredBy and CoveredByrepresent relations,…etc.The table4 provides an example of some used terms in the ontology:

Name of the term	**Synonyms**	**Description**
ContextModel	The model of context	• Model all the concepts of the context related to the ubiquitous environment.
ApplicationContext	• -	• Represent the ubiquitous application
ServicesApplication	• -	• Represent the services offered by the application in question.
……..	• ……	• …….

Table 4: Glossary of Terms.

b) Concepts Diagram

In this step, we build the diagram classification of concepts. The classification hierarchy of concepts demonstrates the organization of ontology concepts in a hierarchy that expresses the relationships in the sub-class (see figure2). A universal concept "Thing" that generalizes all the roots concepts of the different concept hierarchies is used to form one global hierarchy. To build the taxonomy of concepts, METHONTOLOGY proposes to use the four relationship,s: *Subclass-Of, Disjoint-Decomposition, Exhaustive-Decomposition*, and*Partition*. A concept C1 is a subclass of concept C2 if and only if every instance of C1 is an instance C2. for example, *CauseConflict* is a subclass of *ConflictContext*. A *Disjoint-Decomposition of a* C is a set of subclasses of C which not cover C and do not have common instances.

For example, the concepts: DevicesPreferences and NetworkPreferences constitute a *Disjoint-Decomposition* of the concept PreferencesContext. *Exhaustive-Decomposition* of a concept C is a set of subclasses of C which cover C and may have common instances.A *Partition* of a concept « C » is a set of subclasses of C which cover C and may have common instances have no common instance. For example, the concept CauseConflict and SolutionConflict constitute a *Partition* of the concept ConflictContext. Figure.2 presents the concepts classification diagram.

c) Binary Relations Diagram:

A binary relation is used to connect two concepts together (a source concept and a target concept). This activity consists in building a binary relationship diagram (see figure3) which allows representing graphically the various relations existing between the various concepts of the same or different hierarchy.

d) Concepts Dictionary:

The concept dictionary contains the domain concepts. For each concept we define its known Concept name, Instances, Attributes instance, Relationships (see table5 for some concepts).

Concept name	**Instances**	**Attributes Instance**	**Relationships**
ContextModel	-	IDContMod Description	-
ConflictContext	Conflict1, conflct2 Conflict3, conflict4 Conflict5	IDConf DescripConfl	HasSugg AttachedTo CausedBy OccuredIn
CauseConflict	C1,C2,C3,C4,C5	IDCause DescripCause	HasSolution
SolutionConflict	S1,S2,S3,S4,S5	IDSolution DescripSolution	ConcernCause
…… ……	…… ……	…… ……	…… …

Table 5: Concepts Dictionary.

e) Table of Binary Relations:

This table defines for each relation used in the diagram of binary relations: Name relationship, Source concept, source cardinality (max), Target concept and inverse relationship (see table 6 for some relations).

Name relationship	**Source concept**	**Source cardina-lity, (max)**	**Target concept**	**inverse relation-ship**
IsConceredBy	ServicesApplication	• N	Requested Service Preferences	Concern
HasSugg	ConflictContext	• N	Conflict Suggestion	Concern-Conf
AttachedTo	ConflictContext	• N	Display Preferences	Occur
…… …… …	…… …….	… …	…… ……	… ……..

Table 6: Table of Binary Relations.

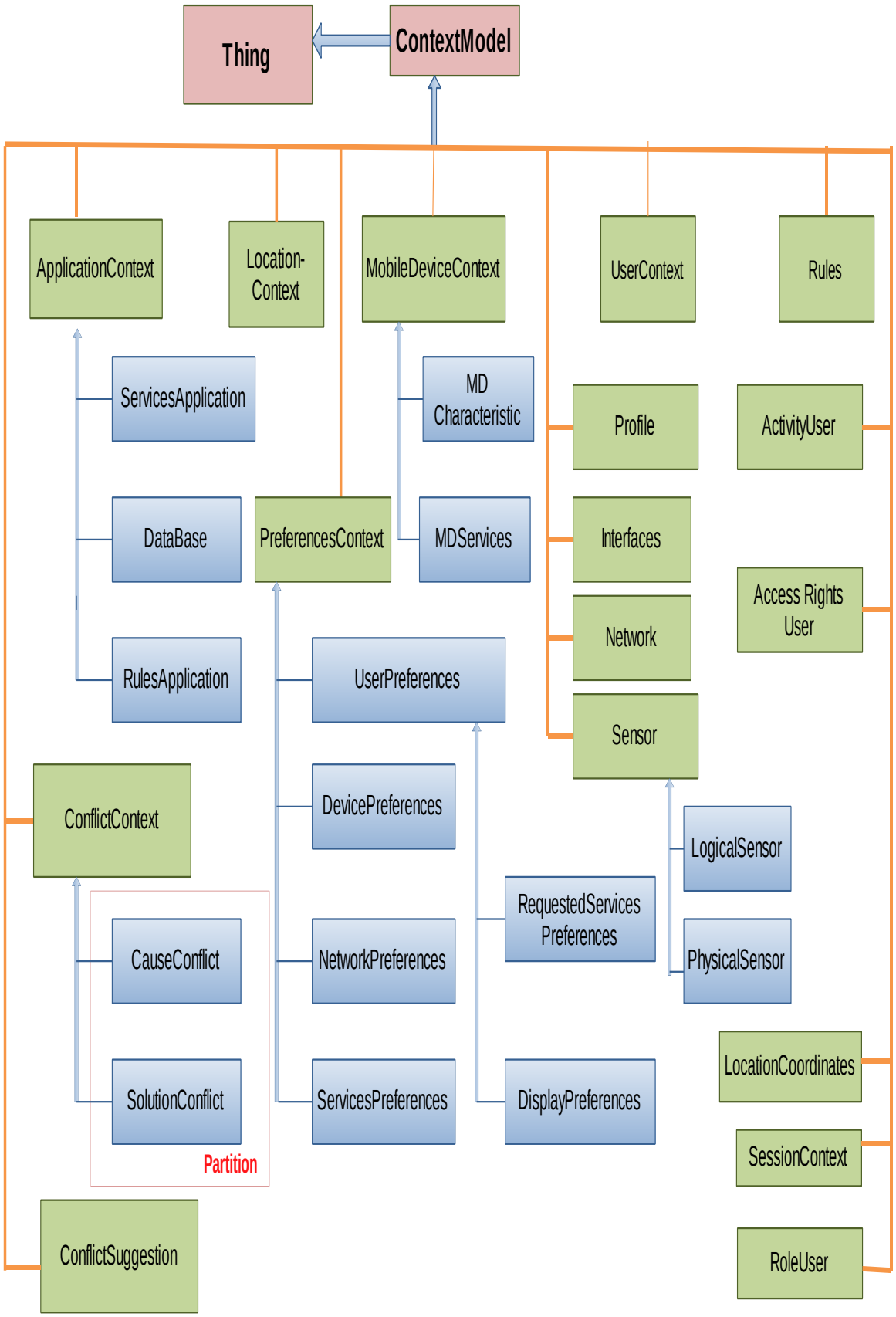

Figure 2: Concepts Classification Diagram.

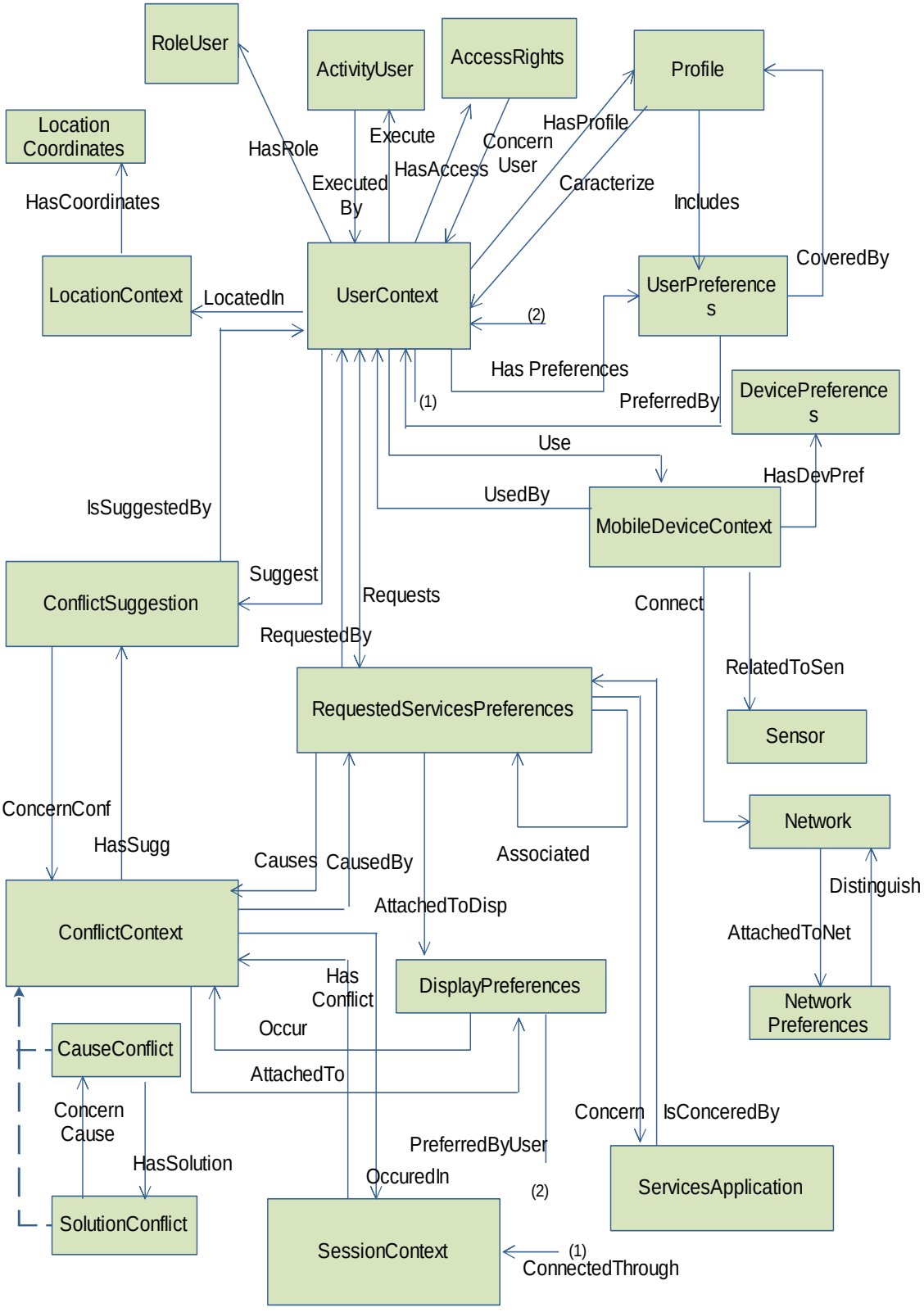

Figure 3: Binary Relations Diagram.

f) **Attributes Table**:

The attribute table (see table 7 for some attributes) specifies for each attribute included in the dictionary of concepts, the set of constraints and restrictions on these values.

Attribute name	Concept name	Value Type	Value range	Cardinality
IDContMod	Context Model	String	• -	• (1,1)
Description	Context Model	String	• -	• (1,1)
IDApp	ApplicationContext	String	•	• (1,1)
DescriptApp	ApplicationContext	String	• -	• (1,1)
…… ……	…… ……	……	……	……..

Table 7: Table of Attributes.

g) Instances Table.

This table describes the known instances that are already identified in the dictionary of concepts. For each instance, specify the instance name, the concept where she belongs, these attributes and values that are associated with it. Table 8 illustrates some instances created.

Instance Name	Concept name	Attributes	Values
ContextConflict_1	Conflict Context	IDConf DescripConfl	Contradiction between the Requested_Service_Preferences and access rights of the user
CauseConflict_1	CauseConflict	IDCause DescripCause	The user requests a Service which does not suit with these access rights.
SolutionConflict_1	Solution Conflict	IDSolution DescripSolution	Suggestion
SolutionConflict_2	Solution Conflict	IDSolution DescripSolution	Stop
SolutionConflict_3	Solution Conflict	IDSolution DescripSolution	ContoLogy
…… ……	…… …… ….	…… …….	…… ……

Table 8: Instances Table.

h) Logical Axioms Table

The table of axioms defines the concepts using logical expressions. Each axiom includes the name of the concept on which gate the axiom, a natural language definition and logical expression (see table9, for some logical axioms).

Concept	Description	Expression logique
UserContext	A user has rights access, execute activities, request services, has a role, hasa profile, exist in a location, prefer display preferences	∀(X), UserContext(X) ⇒ ∃(Y), AccessRights (Y) ∧HasAccess (X, Y) ∧ ∃(Z), ActivityUser (Z) ∧ Execute (X, Z) ∧ ∃(W), RequestedServicePreferences (W) ∧ Requests (X, W) ∧ ∃(R),RoleUser (R) ∧ HasRole(X,R)∧ ∃(P),Profile (P) ∧ HasProfile(X,P) ∧ ∃(L),LocationContext (L) ∧ LocatedIn(X, L)∧ ∃ (D), DisplayPreferences (D) ∧ PreferredByUser (X, D)…….
…… ……..	…… …….	……

Table 9: Logical axioms.

5.4 Formalization

In this step, we use the formalism of description logic to formalize the conceptual model that we obtained in the previous stage of conceptualization. We Define the ContextModel as follows**:** ***ContextModel =(T ,A) with*** T= (Tbox) et A=(Abox)

a) The TBox Construction:

We build the TBox concepts by defining concepts, roles and using constructors provided by description logics. For example, the definition « a 'ActivityUser' must be at least performed by a 'user' , can be written in description logic : ***ActivityUser≡ ∃ExecutedBy***

In addition, we build the TBox by specifying subsumption relations between the various concepts / roles; for example, specify that the class 'User Context ' is subsumed by the class' ContextModel we written: ***UserContext⊑ContextModel***

The definition of some concepts is illustrated in the table10 below.

Concept	Definition	Subsumption relations
ContextModel	≡ (UserContext ⊔MobileDeviseContext ⊔ LocationContext ⊔ ApplicationContext ⊔ ConflictContext ⊔ ConflictSuggestion ⊔ PreferencesContext ⊔ Profile ⊔ Intrefaces ⊔ Network ⊔ Sensor ⊔ Rules ⊔ ActivityUser ⊔ AcessRights ⊔	ContextModel ⊑ ⊤

	Location Coordinates ⊔ RoleUser)	
ConflictContext	≡ (CauseConflict ⊔ SolutionConflict) ⊓ ∃HasSugg.ConflictSuggestion ⊓ ∃CausedBy.RequestedServicesPreferences ⊓ ∃ AttachedTo.DisplayPreferences ⊓ ∃OccoredIn.SesionContext	ConflictContext ⊑ ContextModel
… …… …… …..	…… ……	…… ……

Table 10: Definition of TBox.

b) The ABox Construction:

We describe the facts by using the assertional language, as follows: (1) **A(C)**: To specify that A is an instance of class C, for example: CauseConflict(CauseConflict_1). (2) **R (A1, A2)**: To specify that the two individuals A1 and A2 are connected by the relation R. For example: HasSolution (ConflictContext_1, Solutionconflict_1). In both Tables: Table11 and Table12, we define some assertions:

Concept	Definition
Conflict Context	• ConflictContext(ConflictContext_1),ConflictContext(ConflictContext_2) • ……
Cause Conflict	• CauseConflict(CauseConflict_1) • ……
Solution Conflict	• SolutionConflict(SolutionConflict_1) ……

Table 11: Concepts Assertional Part.

Relation	Definition
HasSolution	HasSolution(ConflictContext_1, SolutionConflict_1)
…… ……..	……

Table 12: Relations Assertional Part.

6 The context rules description: conflicts manage rules

By using the ontology "ContoLogy", we can derive a new context. The context derived is an implicit context derived from explicit context. In our context ontology, derived based on rules in the form **antecedent** → **consequent**. Antecedent and consequent are composed of one or more concepts of context and the description of roles. Derived context can affect other contextual aspects. For example: ConflictContext is a context derived from MobileDeviceContext, UserContext and UserPreferences. In our work, we planned to resolve all conflicts which can arise when checking the user's preference. In the precedent section, we have defined five conflicts that may arise during the verification of user preferences. To manage these conflicts, we used the Semantic Web Rule Language (SWRL), we have defined five SWRL to derive conflicts, five SWRL to resolve these conflicts and we have created these rules under ProtégeOwl [41].

6.1 SWRL to Derive Conflicts

We define five rules to derive the five conflicts (see table N°9)

- ✓ ***Rule1: derive the Conflict1: "*** Contradiction between The Requested_Service_Preferences and access rights of the user " : *UserContext(?x) ∧ RequestedServicesPreferences(?A) ∧ AcessRightsUser(?AR) ∧ differentFrom(?A, ?AR) ∧ ConflictContext(?c) → Causes(?A, ?c)*
- ✓ ***Rule2: derive the Conflict2: "***Contradiction between the display preferences and the characteristics of used MD ": *UserContext(?x) ∧ DisplayPreferences(?d) ∧ MobileDeviceContext(?dm) ∧ differentFrom(?dm, ?d) ∧ ConflictContext(?c) → Occur(?d, ?c)*
- ✓ ***Rule3: derive the Conflict3: "***Various wishes of Display for the same service": *UserContext(?x) ∧ RequestedServicesPreferences(?A) ∧ MobileDeviceContext(?dm) ∧ differentFrom(?dm, ?d) ∧ sqwrl:isEmpty(?d) ∧ ConflictContext(?c) → Causes(?A, ?c)*
- ✓ ***Rule4: derive the Conflict4: "***Absence of display preferences after checking the historic of the user": *UserContext(?x) ∧ RequestedServicesPreferences(?A) ∧ sqwrl:isEmpty(?d)∧ Notprefered(?d, ?x) ∧ ConflictContext(?c) → Causes(?A, ?c)*
- ✓ ***Rule5: derive the Conflict5: "*** Contradiction between the Display preferences requested and display capabilities expressed": *UserContext(?x) ∧ RequestedServicesPreferences(?A) ∧ MobileDeviceContext(?dm) ∧ differentFrom(?dm, ?d) ∧ sqwrl:isEmpty(?d) ∧ ConflictContext(?c)* →Causes(?A, ?c)

6.2 SWRL to Resolve Conflicts

We define five SWRL for resolving the five Conflicts, see table N°9 for the description (values) of all parameters of the following rules.

- ✓ ***Rule6: resolve the Conflict1:*** *ConflictContext(ConflictContext_1) ∧ CauseConflict(CauseConflict_1) → HasSolution(ConflictContext_1, SolutionConflict_1)∧ HasSolution(ConflictContext_1, SolutionConflict_2).*

- ✓ ***Rule7: resolve the Conflict2:*** *ConflictContext(ConflictContext_2) ∧ CauseConflict(CauseConflict_2) → HasSolution(ConflictContext_2, SolutionConflict_3)∧ HasSolution(ConflictContext_2, SolutionConflict_1)∧ HasSolution(ConflictContext_2, SolutionConflict_4)*

- ✓ ***Rule8: resolve the Conflict3:*** *ConflictContext(ConflictContext_3) ∧ CauseConflict(CauseConflict_3) → HasSolution(ConflictContext_3, SolutionConflict_5) ∧ HasSolution(ConflictContext_3, SolutionConflict_4)*

- ✓ ***Rule9: resolve the Conflict4:*** *ConflictContext(ConflictContext_4) ∧CauseConflict(CauseConflict_4) → HasSolution(ConflictContext_4, SolutionConflict_1) ∧ HasSolution(ConflictContext_4, SolutionConflict_4)*

- ✓ ***Rule10: resolve the Conflict5:*** *ConflictContext(ConflictContext_5) ∧ CauseConflict(CauseConflict_5) → HasSolution(ConflictContext_5, SolutionConflict_1)∧ HasSolution(ConflictContext_5, SolutionConflict_3)∧ HasSolution(ConflictContext_5, SolutionConflict_4)*

6.3 SWRL Rules Creation with Protégé:

We have used PROTÉGÉ 2000 to implement the precedent rules. Figure 4 show the creation of the SWRL rules under protégé

7 Ontology implementation

After the conception of the ontology "ContoLogy", we will implement our ontology. For this, we choose the editor Protégé OWL [41] and we used to formulate the ontology in the knowledge representation the language OWL. OWL represents a codification language used to implement the OWL ontology, and that, for all semantic functionalities than allows OWL which is richer than languages DAML + OIL & RDFS. In addition, we use to check the ontology the reasoner RACER (calculate the subsumption relation between concepts, and check the consistency of all concepts) [42].

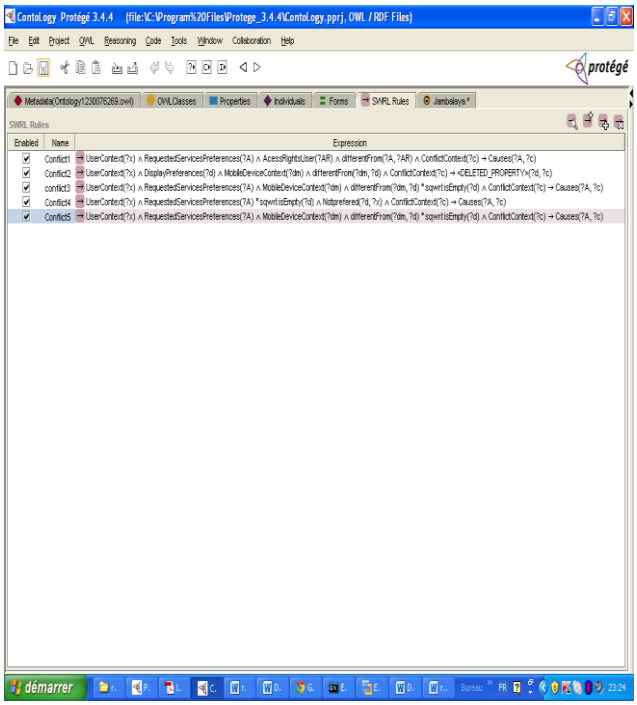

Figure 4: SWRL for Managing Conflicts.

PROTEGE OWL is a modular interface, developed at Stanford Medical Informatics, to edit, visualize, control (check constraints) ontologies [41]. PROTEGE OWL allows the definition of meta-classes which whose instances are classes, which allows you to create its own model of knowledge before building ontology. Many plugins are available or can be added by the user. The software architecture allow the insertion of plug-ins that can introduce new features (for example, the ability to import and export ontologies built in various operational representation languages such as OWL or specification of axioms) participated in the success of PROTEGE OWL, which includes a very large user community and is a reference for many other tools [43].

7.1 Implementation steps

First we start by creating concepts specified in the conceptualization step. After building classes, we create the properties for each of them see figure 5, and then we create restrictions on classes and properties see Figure6 and figure 7.

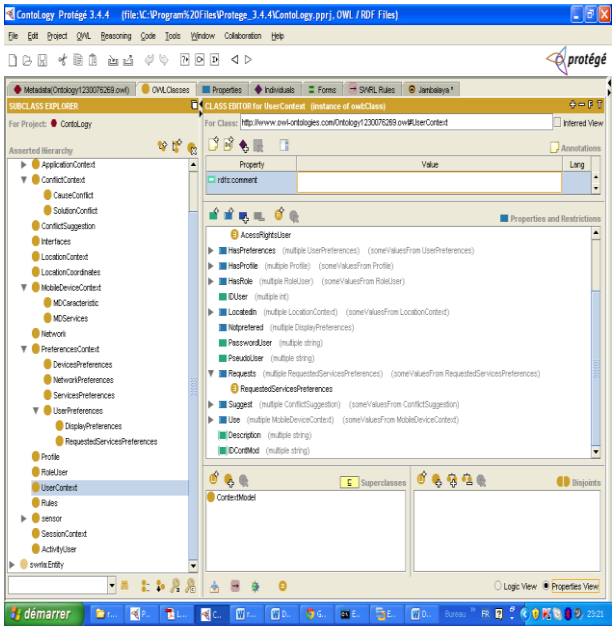

Figure 5: Contology classe and properties creation.

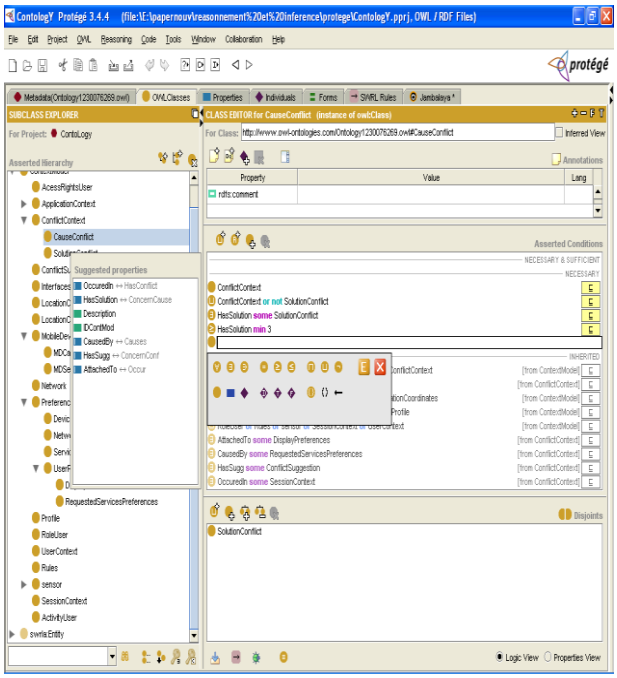

Figure 6: Contology restriction view1 with PROTÉGÉ.

After this step, we can transform the ontology to OWL form. An excerpt from the context model ontology in OWL is illustrated below:

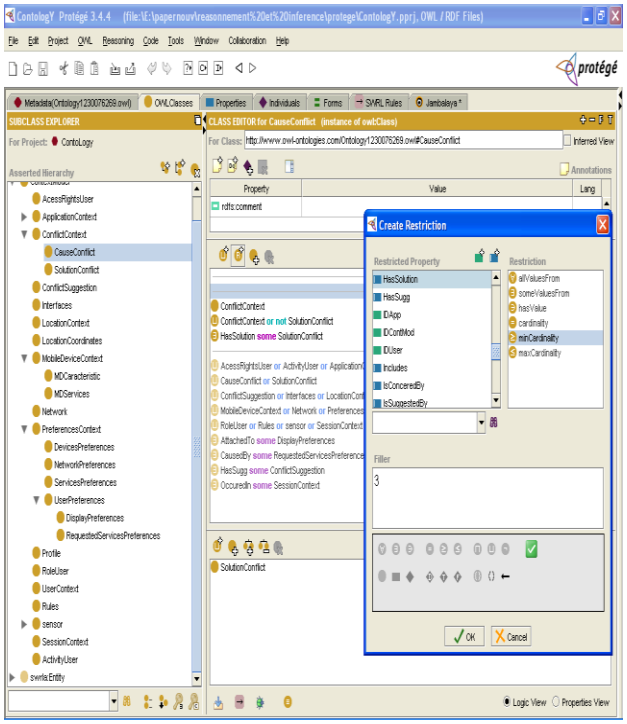

Figure 7: Contology restriction view2 with PROTÉGÉ.

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"xmlns:protege=http://protege.stanford.edu/plugins/owl/protege#
        xmlns="http://www.owl-ontologies.com/Ontology1230076269.owl#"
    xmlns:swrl="http://www.w3.org/2003/11/swrl#"
    xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
        ……..
</owl:Ontology>
<owl:Classrdf:ID="ServicesPreferences">
<rdfs:subClassOf>
<owl:Classrdf:ID="PreferencesContext"/>
</rdfs:subClassOf>
</owl:Class>
.....<owl:Classrdf:ID="Profile">
<rdfs:subClassOf>
<owl:Restriction>
<owl:someValuesFrom>
<owl:Classrdf:ID="UserPreferences"/>
</owl:someValuesFrom>
<owl:onProperty>
<owl:ObjectPropertyrdf:ID="Includes"/>
</owl:onProperty>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty>
                ……..
<owl:Restriction>
<owl:someValuesFrom>
<owl:Classrdf:about="#ConflictContext"/>
</owl:someValuesFrom>
<owl:onProperty>
    <owl:ObjectPropertyrdf:ID="Causes"/>….
```

7.2 The “Contology” Test

We used the system Racer to test the ontology "Contology", we distinguish three types of test: Inference, Consistency test and classification test; The first consists on remove the inconsistency between concepts, and this by using the subsumption test incorporated into the Racer system, against the second allows to check the existence

of each concept instances; a concept C is satisfiable if and only if there is at least an interpretation I (instance) for the concept C. Racer is in the form of a server which can be accessed by TCP or HTTP. So we must first configure the connection to the server hosting the system Racer. We have carried all tests, and they are checked. Figure8 shows an example of inference test, figure 9 shows an example of consistency test and figure 10 shows an example of classification test.

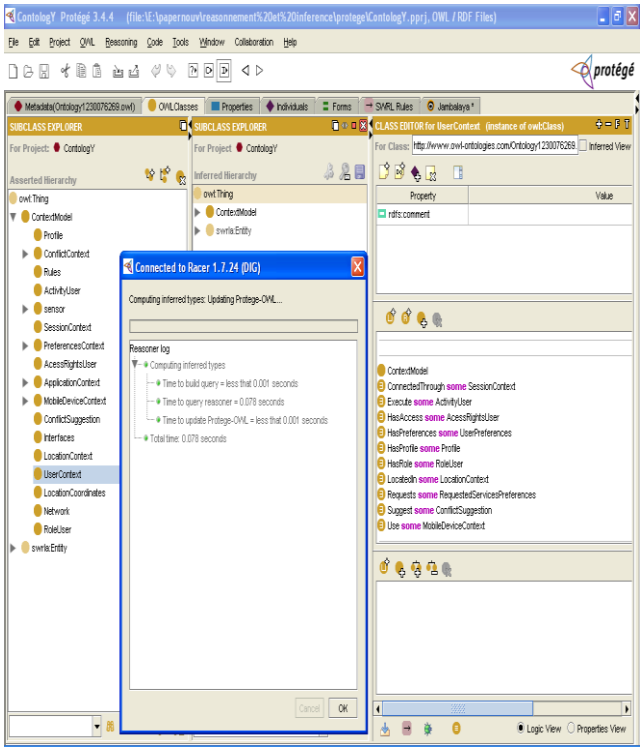

Figure 8: Test of inference.

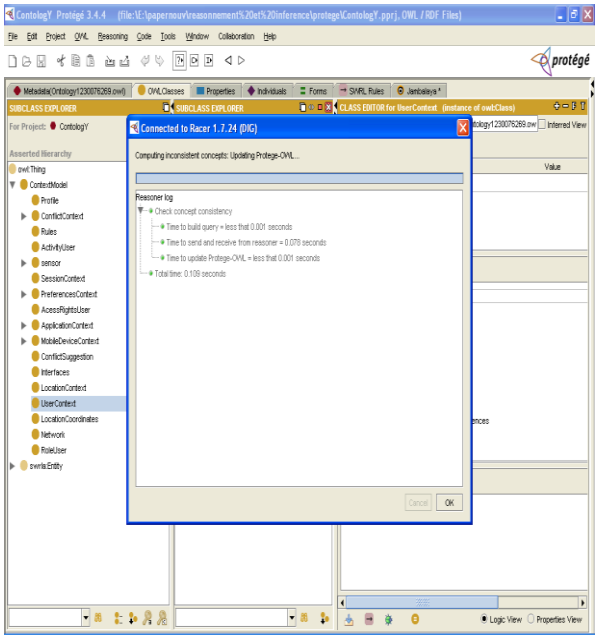

Figure 9: Test of Consistency.

According to the tests we have applied to the ontology "ContoLgy", no error is produced during the test.

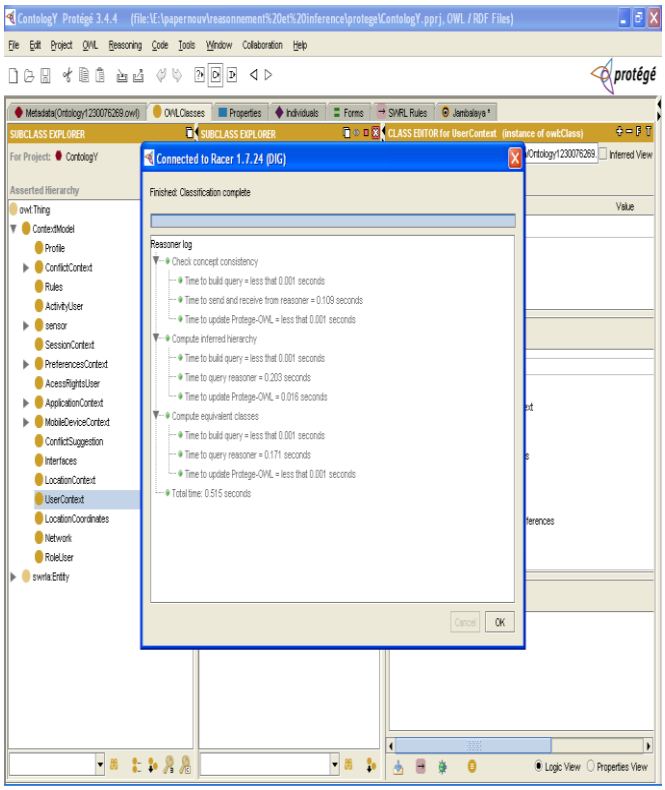

Figure 10: Test of classification.

8 The context ontology exploitation: adaptation process

We exploit and use the ontology “ContoLogy” to adapt the user's initial request to the current context. Thus, we propose a web service based architecture to ensure the adaptation process. By use of the ontology "ContoLogy", adaptation can reasoning about the user's context and adapts the user's initial request to the current context. Among the different context parameters, we focus on: the location and the used Mobile Device (MD).

After having implemented the application, it is mandatory, for many reasons, to undergo it to the adaptation process. These reasons can be classified into four categories [44]: (1) Correctional Adaptation, (2) Adaptive Adaptation (3) Scalable Adaptation and (4) Perfective Adaptation. In our approach, we are interested to the adaptive adaptation in order to adapt ubiquitous applications to their execution environment. We adopt this kind of adaptation because the application is running properly, but its execution environment, hardware components or other applications or depending data are changing (e.g. the context of user). In this case, the application is adapted in response to changes in its execution environment. Consequently, to ensure this adaptation process, we use the context ontology “contology” to the adaptation composed of two main parts: static part and the dynamic part.

(1) **Static part:** This part is described by the ontology "Contology". It focuses, on one hand, on modeling the contextual information of users and their preferences and, on the other hand, on managing the potential conflicts which may arise between the users' preferences during their checking process.

(2) **Dynamic part:** the role of this part is to ensure the functional dynamic adaptation of context-sensitive applications to various user's contextual situations. The adaptation process adopted by this part is based on " C*ontoLogy"* in order to offer a better respond to user. Also, this process is assured by the user's initial request adaptation to the context of use and user's preference using the ontology "ContoLogy". The methodology in our approach consists in three main steps: (1) the context of use modeling and the user's preferences managing, basing on a new context definition which separates the application data from the contextual data by by using "ContoLogy",, (2) the resolution of potential conflicts which may be occurred during managing of user's preferences and (3) the dynamic functional adaptive adaptation of web service-based context-aware applications. The accomplishment of the two last steps (2 and 3) is based on "ContoLogy".

In ubiquitous computing, applications are sensitive to the context (context-aware applications), user's access to various information's using different mobiles devices and in different localization, which implies, an overly dynamic, heterogeneous environment. To respond better to this challenge, we propose to use web service, for those benefits, such as:

1. The ultimate goal of the Web service approach is to transform the Web into a distributed computing system where programs (services) can interact intelligently by being able to automatically discover and negotiate with each other and consist into more complex services [45].
2. The establishment of web services facilitates the dialogue between heterogeneous environments. As web services can be implemented on different platforms and with different languages, they facilitate interoperability between heterogeneous systems and platforms, which is our case. [46]
3. Web Services [47] work with standard Web protocols (HTTP and TCP / IP) and XML. Many companies already have a Web infrastructure the staff have the knowledge and experience of maintenance. This is why the cost of access to Web services is much lower than that of previous technologies.[6]

The figure 11 shows the general architecture of the proposed approach.

As illustrated by the figure 11, the adaptation process to the context of use and the user's profile is accomplished in 16 steps explained bellow:

(1) ***Request***: the user sends his request to the platform via his Mobile Device (MD). The Module Context integration (CI) receives this request.

(2) ***Contextual information***: the module Context sensor sends contextual information of the user to the module Context integration, such us: the used MD, the localization.

(3) ***Contextual request***: in this step, the Module Context Integration increases the user request by the contextual information; the result of this step is a contextual request. The module (CI) sends this contextual request to the Preferences Management Web Service (PMWS).

(4) ***Preferences check***: In this step, the PMWS checks the contextual request using "ContoLogy". It checks the conformity between the user preferences and his access rights and the type of the used MD.

(5) ***Prefrences OK/ Conflict***: by consulting the ontology, the PMWS can detect that preferences are checked or can detect a conflict

(6) ***Soap Message: Conflit; Soap Message: Conflit:*** when a conflict arises, the PMWS sends a soap message containing the conflict to the Conflict Management Web Service (CMWP).

(7) ***Search Conflict Solution:*** using the Context Ontology, the CMWS Searches a solution for the detected conflict.

(8) ***Solution conflict/ no solution:*** this step indicates whether or not there is a solution for the Conflict.

(9) ***Ask suggestion:*** if no solution to the conflict, the CMWS asks a suggestion of solution for the conflict from the user.

(10) ***Soap message: conflict solution:*** in this step, if the user sends a suggestion of solution for the conflict to the CMWS, it takes this solution and sends it to the PWSM.

(11) ***Update conflict information:*** the CMWS updates the conflict information by adding the conflict information of the current session.

(12) ***Soap message: request updated:*** The PMWS sends the request of the user, after the verification, to the adapter web service (AWS).

(13) ***Search answer:*** the AWS search an answer for the request of the user.

(14) ***Soap message: answer:*** Once the answer is found, the AWS sends it to the PMWS.

(15) ***Answer adapted to the context:*** this later sends this answer adapted to the context to the user

(16) ***Update contextual information:*** finally, the PMWS updates the contextual information by adding the contextual information of the current session to the Context Ontology.

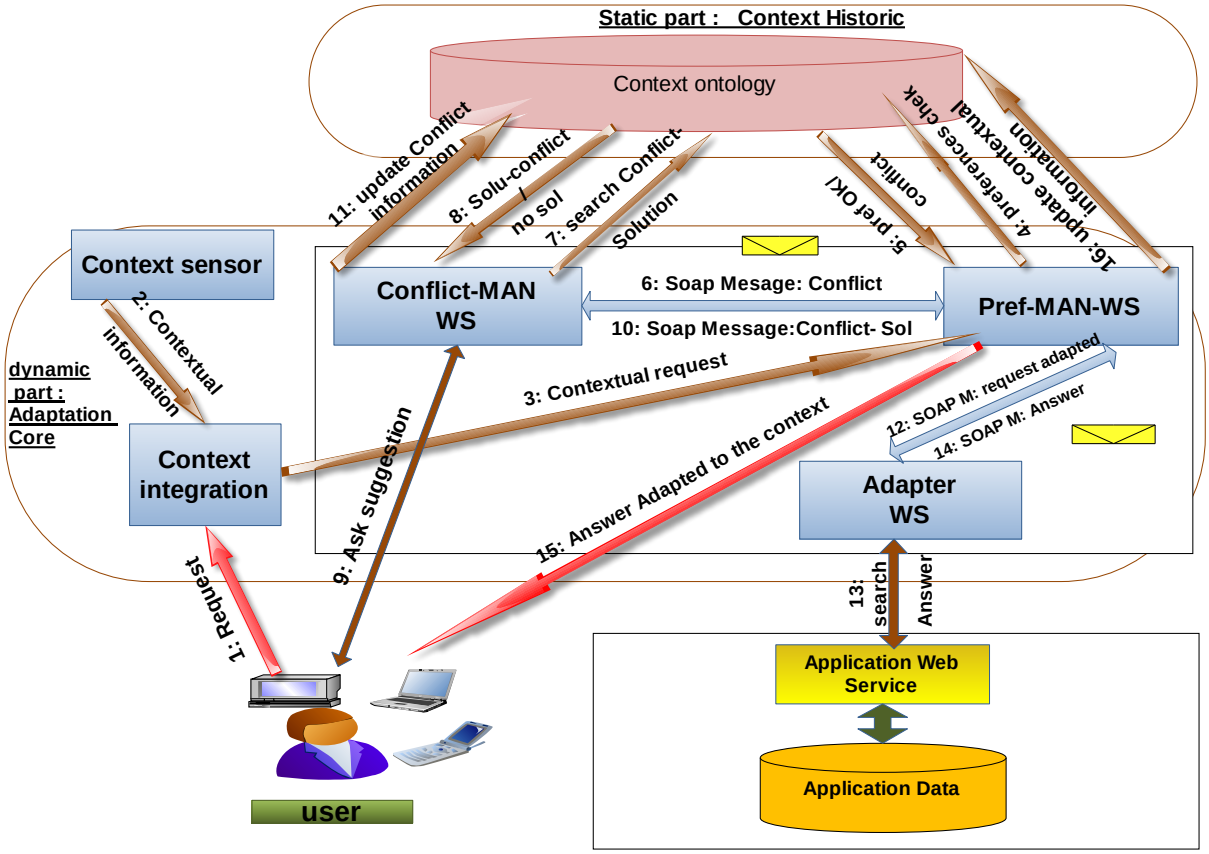

Figure 11: Architecture of our approach.

8.1 Process adaptation presentation

In this section, we present the dynamic part of our approach to adapt the ubiquitous applications to the user's context and the user's profile, using "ContoLogy".

This part assures the functional dynamic adaptive adaptation of these applications sensitive to the context of use and the user's profile, it is assured by the adaptation of the initial request of user to the context of the current session in the various contextual situations. At the end, the user can meet the best answers to their expectations.

The context of use of a user witch accedes to a ubiquitous application, in addition to be composed of multiples aspects is very variable and in constant evolution, which makes the adaptation process of the application hard to accomplish. In order to ensure this adaptation process and to be able to change the behavior of such application sensitive to the context of use, we propose to use Web Services (WS) both during the development of this type of application and in the dynamic part of the adaptation.

We opted for web service for the advantages it procures. The dynamic part of our approach is composed of three Web Services: Preferences Manager Web Service (PMWS), Conflicts manager Web Service (CMWS) and Adapter Web Service (AWS) and two modules: Context integration and context sensor. This part assures the adapting of the user request to the context, resolving the conflicts and returning an answer adapted to the user's context.

8.1.1 Preferences Manager Web Service

This web service is charged of the preferences management. Consequently, it ensures checking of the user's preferences using the initial request of the user and "ContoLogy", the PMWS can reason on the user context. The PMWS can analyze the context of the user that appears in the contextual request of the user. Consequently, it verifies the conformity between the requested preferences and the context of use, mainly the used MD, localization and his accesses rights. This step can generate conflicts which can be detected by PMWS.

Also, it reformulates the initial request of user, in the case of conflicts, by adding the new preferences. It sends to the user the adapted answer to the context, and stored the new context for using it in the next sessions, when we receive the same context and request (see Figure 12).

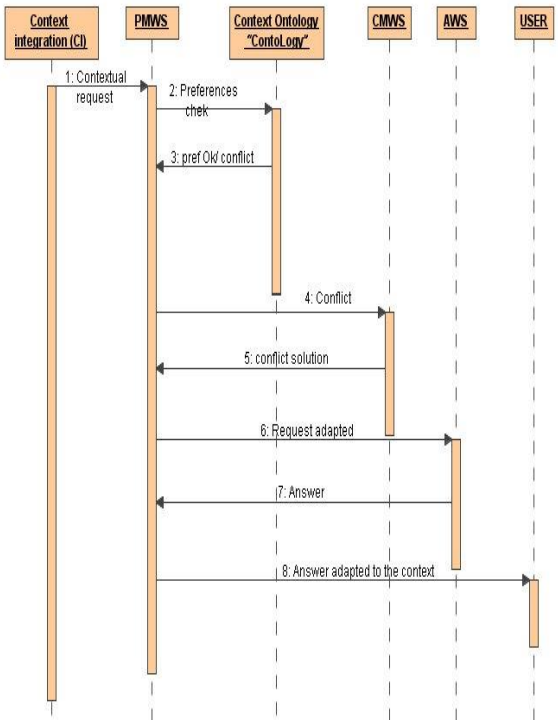

Figure 12: PMWS Sequence Diagram.

Technical Example: see section 9.1

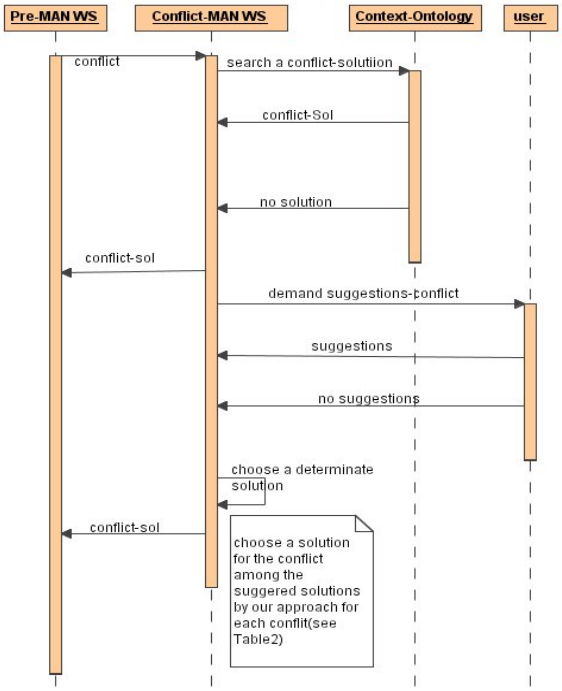

Figure 13 .Conflicts Sequence Diagram.

Technical Example: see section 9.1

8.1.2 Conflicts Manager Web Service

The role of this web service is to manage the conflicts that may arise between user preferences. The conflicts are managed, by our approach, according to the following sequence diagram (Figure 13). Specifically, our approach manages five conflicts (1) Contradiction between The Requested_Service_Preferences and access rights of the user. (2) Contradiction between the display preferences and the characteristics of used MD. (3) Various wishes of Display for the same service. (4) Absence of display preferences after checking the historic of the user. (5) Contradiction between the Display preferences requested and display capabilities expressed.) (see table 2). This web service executes the proposed solution for each conflict can be arose between the preferences of user (Table 3). After receiving a message containing the conflict which has occurred, *Conflicts Manager Web Service* reasons and infers a solution to conflict occurred by using "ContoLogy", if not; it implies the user to give his suggestions for this conflict. If there are no suggestions it takes a default solution, for each conflict (i.e. our approach proposes a determinate solution (see table 3). At the end, it sends a message which contains the solution of the conflict to the PMWS. Consequently, it updates the history of conflict information. This web service ensures: the resolution of conflicts using "Contology", and the storage of information of the occurred conflict.

8.1.3 Adapter Web Service

Its role is to return an adapted request to the user. It executes the following steps: firstly, it accedes to the Web Services of the application and researching on the WSDL of these latter, in order to extract Web Services with their interfaces, their operations and the number of interfaces specific to each Web Service. Secondly, selecting the Web Service which answers better the request of the user. Then, it reformulates and sends to PMWS the adapted answer to the context of use.

8.1.4 Context Sensor

This module is responsible of the capture of the user context at a connection time, namely: localization, MD, session. Then, it sends this contextual information to the module "Context integration". It is composed of the two following Sub-modules:

1- *Logical context sensor*: a set of interfaces used by the user to enter his context.
2- *The physical context sensor*: a set of physical dispositive used to capture the context of the use.

8.1.5 Context Integration

This module receives the initial request of the user and reformulates it by adding the contextual information. Then, it sends this contextual request to PMWS.

8.2 Utilization of "CONTOLOGY":

In this section, we explain how the user communicates with our platform to get an adapted response to its context (figure14), the adopted communication process is accomplished in four main steps:

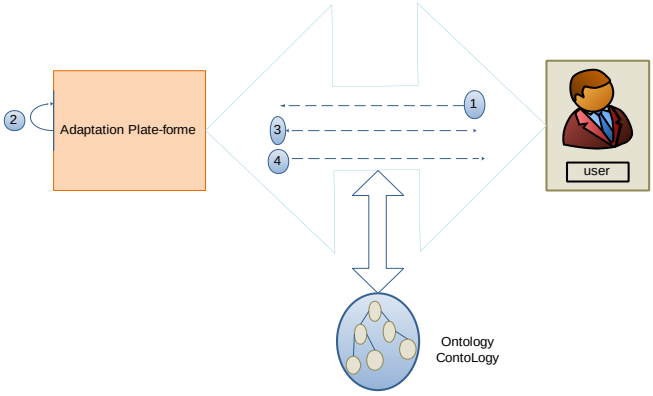

Figure 14: Communication between the user and platform using "ContoLogy".

(1) **Sending Request**: The user sends a request to the platform asking the available services and providing the necessary information (context, location, MD ....).

(2) **initial request augmentation** : The platform, using context sensor module and the module context integration, increases the initial user request by adding contextual information, this contextual request will be sent to PMWS for checking preferences using " ContoLogy".

(3) **Conflicts resolving**: in the case of conflicts when checking the preferences, the platform using the CMWS and the ontology search a solution for the conflict, or demands a suggestion from the user.

(4) A**dapting Response**: after checking preferences and taking into account the context of the user, this latter receives a response adapted to his context.

In the flow, we presents two scenarios using " ContoLogy", in order to show how our p roposed approach uses "contology" to reason and infer new information for taking into account the context.

a- Scenario1: Preferences Checking:

The preferences checking process (figure 15) is accomplished in three main steps:

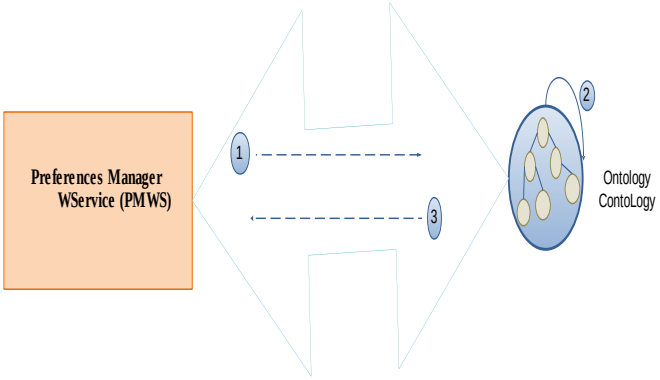

Figure 15: Scenario1: Checking of the preferences.

(1) Contextual request: PMWS uses "ContoLogy" to verify the contextual request of the user that contains the user's context namely the type of the used MD and the location. The PMWS checks the conformity of services requested by the user with their access rights, and display preferences with display capabilities offered by the used MD, and that using the information of the previous sessions stored in the ontology.

(2) Reasoning and inference: according to the contextual information that exists at "ContoLogy" we can check the user preferences, reasoning on the current context with the available information and also infer new user preferences in the case of conflict.

(3) Chek result: in this case, "Contology" can refer two answers. The first answer is: preferences OK, where preferences are checked. The second one is, a conflict has been arisen between user preferences, which must be resolved by the CMWS (see next scenario).

b- *Scenario2*: Conflicts Resolution:

Figure 16 illustrates the conflicts resolving process we propose. It is accomplished in seven steps:

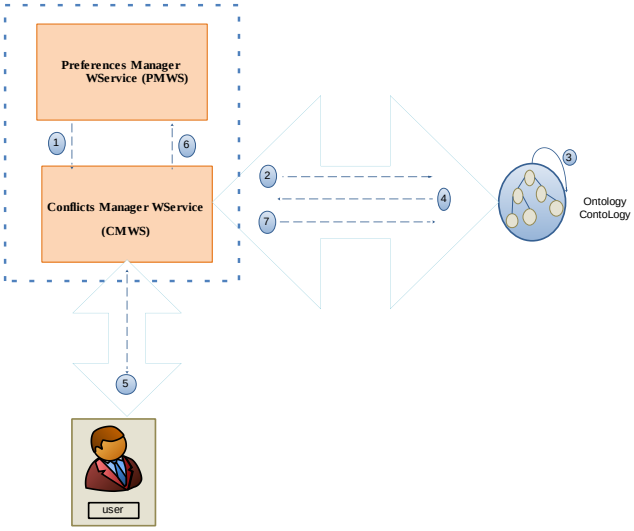

Figure 16: Scenario2: Conflicts resolution.

(1) **Conflict**: the previous scenario can cause a conflict, so it will be sent to the CMWS by the PMWS.

(2) **Searching for a solution to the conflict**: using the Context Ontology, the CMWS Searches a solution for the detected conflict.

(3) **Reasoning and inference about the conflict**: using "ContoLogy", the CMWS can reason about the conflict information of previous sessions and infer a solution to the current conflict.

(4) **Solution / no solution:** this step indicates whether or not there is a solution for the Conflict.

(5) **Conflicts suggestions**: if CMWS does not find a solution to the conflict in the ontology, it asks a suggestion of solution from the user. This latter can give a solution, change the request or does not responds.

(6) **Soap message: conflict solution:** in this step, if the user sends a suggestion of solution of the conflict to the CMWS, he takes this solution and sends it to the PWSM.
(7) **Update conflict information:** the CMWS updates the conflict information by adding the conflict information of the current session.

9 The case study: the travel booking application

In this section, we present using a case study, how we exploit the Ontology "ContoLogy" for the adaptation of the user request. For this, we have created a travel booking application to be used in the process adaptation, and we have implemented the dynamic part of our approach. We will present the different steps we followed during the implementation. Firstly, we present the environment and the tools that we used in the implementation. Secondly, we will present the application we have developed; finally, we detail the implementation steps, by a detailed example, from the reception of the request of the user passing through the resolution of conflicts, until reception of the adapted response by the user.

The environment and tools we used to implement the system Such: Microsoft visual studio(Visual Web Developer, Smart Device Applications, Web Forms, Windows Forms, XML Web Services, XML Support, C#) [48] , Protégé [49], OWL [50].

Travel booking is a web service-based application to manage a travel agency and Online reservation (see figure17).

It offers to user to make flight reservation and hotels reservation. This application is adapted by our architecture to the context and the profile of the user. Using this application the user can search for a flight, a hotel and car, and he can receive an answer adapted to his context, for example: adapted to: his location, the used MD, his city and the location of the airport. For example: the user can receive a list of hotel situated near the airport. For designing the agency services, we distinguished three web services:

(1) *Airline Service*: It offers services responsible for online managing of the flights reservations of customers.

(2) *Hotel Service:* It offers services which have like function, the online control of the hotels and reservations of the customers.

(3) *Location Car service:* It classifies all services responsible for online managing of cars and location.

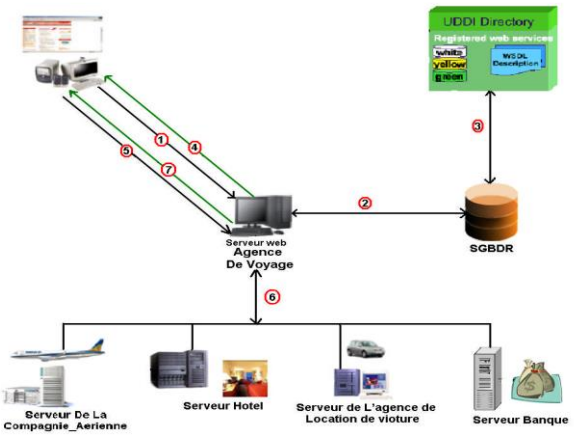

Figure 17: Global architecture of the application.

We have created a service portal that serves as a gateway to various web services. This portal does not store any data on its physical basis, but acts as a service provider. The application we have developed allows to a customer to avoid making several research on the web (airlines, hotel, car ...), to plan his travel. The portal we have implemented provides the interfaces necessary to planning travel through the use of web service technology. This application will be used by our system for the adapting to the context of use and the profile of user. The dynamic part of our approach ensures the process of adaptation, which will be the subject of the following section.

All web services related to the dynamic part which are necessary to validate our approach are created using Microsoft visual studio. More precisely, three web services have been created to handle the interaction and the messages between the user and the application. After the web service creation, a C# page will pop on which named service1.asmx.cs. The page contains the library that we need and the web service code behind. To create a web service method in *.net* environment, simply we write the [WebMthode] and after that we write the method .

9.1 Process Adaptation Unfolding

In this section, we detailed our approach to manage preferences and conflicts, and detail the process adaptation unfolding, by using an example which explain the interactions between web services of our architecture, the ubiquitous application (Travel booking application); the context ontology "ContoLogy" and the user. For this, we present an example which includes basically the following points: (1) Interaction between user and the dynamic part and the context ontology "ContoLogy". (2) The receipt and the check of the user request. (3) Resolution of conflicts. For this, we take a conflict that can occur and we explain how the system will handle this conflict and we will see how the system resolves this conflict step by step. (4) Adaptation of the answer of the application to the context information. In this case we will take as example:

- *The ConflictContext(ConflictContext_2)* ="Contradiction between the display preferences and the MD characteristics"
- causes by *CauseConflict(CauseConflict_2)*= "The user requests a display which is not supported by his used MD"
- With:
  - The solution *SolutionConflict_3*= "ContoLogy" witch means: reasons and infers a better solution from "ContoLogy".
  - If no, then *SolutionConflict_1="suggestion"*witch means, demands a suggestion from user.
  - If no, then *SolutionConflict_4=" default_display_preference"*

a. Interaction between user, our dynamic part and context ontology:

At the first time when the user login to the system, the system asks him to be registered on it, by giving his personal information such as name, username and address, email and choose his services and preferences that he prefer. The system will get automatically the MD (Mobile Device) characteristics from the MD information files. The MD characteristics in the ontology will be look like:

```
default:MD_i0435    MD:MDid "MD_i0435"
                     MD:Class "MD" ;
                     MD:Type "Nokia";
                    MD:ImageD "0" ;
                    MD:TextD "1" .
```

- *User:*

```
Default: i0435    profil:id "profile_i0435" ;
                 profile:Class "USERPROFILE"
                 profil:FName "MM1" ;
                 profil:LName "TT1" ;
                 profil:UserName "us11" ;
                 profil:Password "pass1" ;
                 profil:address "adress AD" ;
                 profil:email "AD@hotmail.com".
```

- *Service Preference "Show flight":*

```
default:preser_i043501
                  preser:Num_Ser "preser_i043501" ;
                  preser:Class "ServicePreferences"
        preser:ser "Show flights" ;
                 preser:serAso1 "preser_i043502" ;
                 preser:serAso2 "0" ;
               preser:dispser disser_i043501_pre01" .
```

As we see here, this service has an associated service "preser_i043502"which is "Show hotel" service

- *Service Preference "Show hotel":*

```
default:preser_i043502
      preser:Num_Ser "preser_i043502" ;
      preser:Class "RequestedServicePreferences "
      preser:ser "Show Hotels" ;
       preser:serAso1 "0" ;
       preser:serAso2 "0" ;
preser:dispser "disser_i043501_pre01".
```

- *Display Preference for: "Show flight" and "Show hotel":*

```
default:disser_i043301_pre01
disser:Num_Dis "disser_i0433_pre01" ;
        preser:Class "DisplayPreferences "
        disser:default "disText" ;
        disser:disText "1" ;
        disser:disImage "1".
```

b. Check of The User Request

After user login, the next figure presents flight searching form will be displayed.

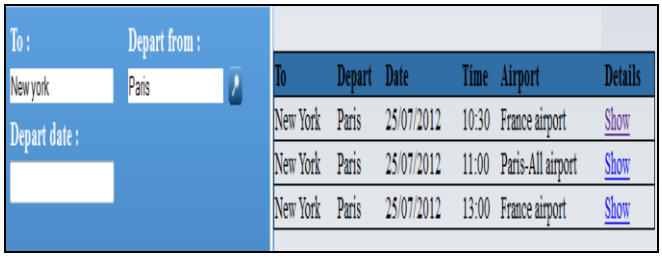

Figure 18: Flight searching result form.

After clicking on show details link, the PMWS receives the query and the contextual information for the user, and checks it with the user preferences and services on the ontology "ContoLogy" by the following steps:(1) PMWS receives the service ID and the contextual information (localization and used MD) by the method "Service_check". This method returns the associated_services and the display preference (figure 19).

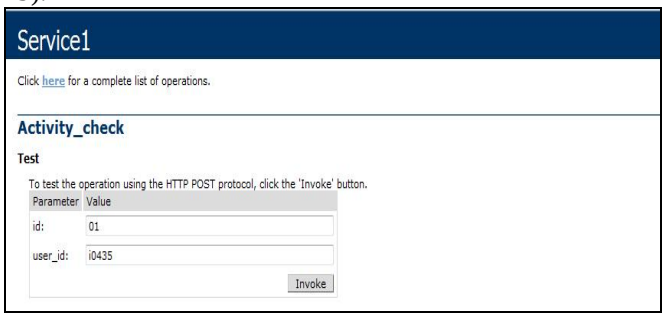

Figure 19: Service_check method call.

2- Next figure presents the soap message receive by the PMWS

Figure 20: Service_check method SOAP 1.1.

3- In figure21, we find the result receive by the PMWS after checking the request of the user using "ContoLogy"

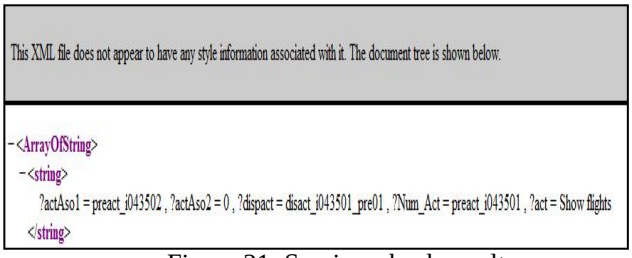

Figure 21: Service_check result.

4- In the next step, the PMWS compare the values that return from "MD_check" method, and the "display_check" method. In our example, the values will be not the same because:

- User MD does not support image display which its valu

(figure22).

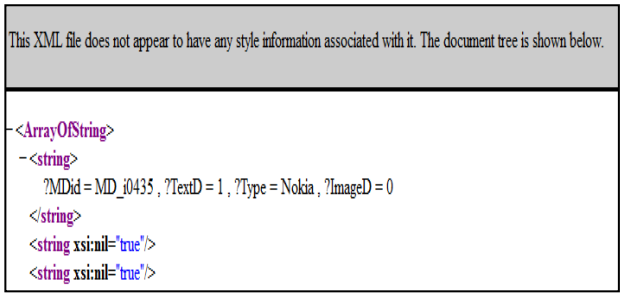

Figure 22: MD_check result.

-Text and image forms in display preference have both the value 1 (figure 23).

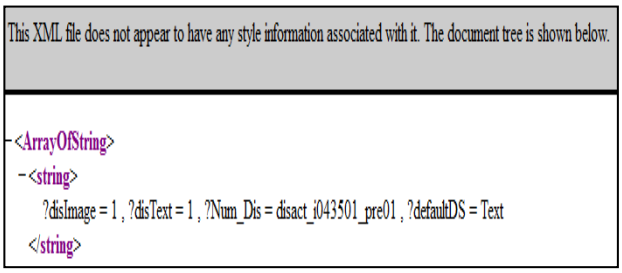

Figure23:Display_check result

c. Conflict of md characteristics and display preferences

In this step the PMWS will detect the conflict between the display preference and the MD characteristics see figure23 and figure24. PMWS send the conflict to the CMWS, which it will consult the conflict and the solution will take to resolve it from the ontology "ContoLogy". The system will check the user history by History_check" method for similar service, and the preferences of that service. If there is not result from the user history, the system will demand the suggestion to the user. The suggestion will aim to change the display preference to this service to be appropriate with user MD (figure 24).

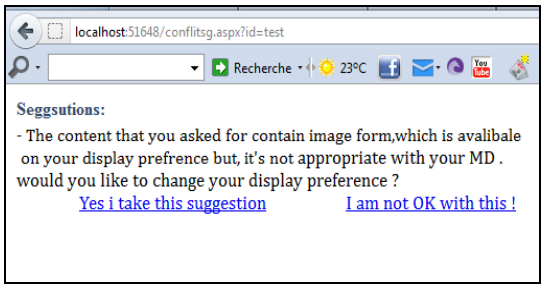

Figure 24: Conflict suggestion.

If the user chooses to take the suggestion, the CMWS sends to the PMWS the suggestion with method "change_cont_info" to update the display preference and change the display image to 0 values (figure 25).

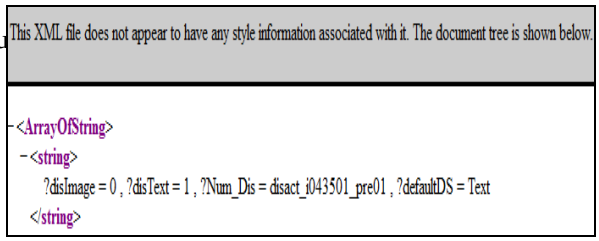

Figure 25: Check display result after the update.

d. The User Request Adaptation

After updating display preference, the PMWS reformulates the user request by adding the contextual information and sends it to the AWS and gets the result from the travel-booking application (see figure26)

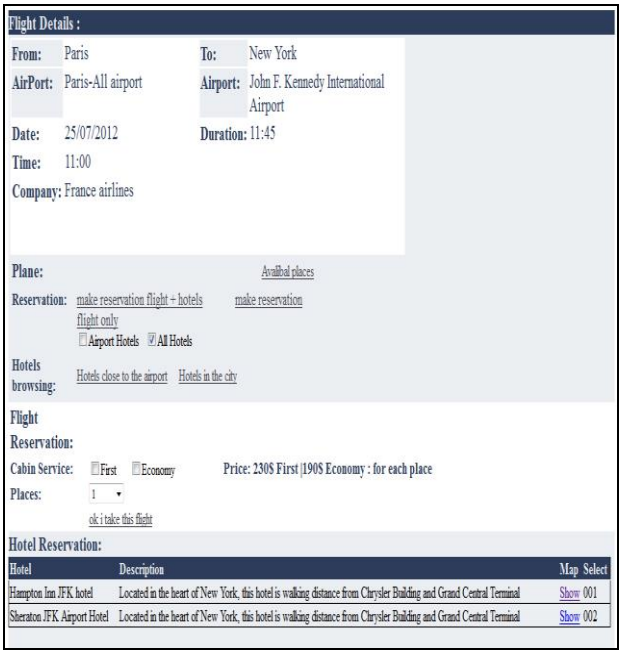

Figure 26: Result after the adaptation.

Figure 26 shows the result of the user request that it adapted to the user context and preferences. Our adaptation process is assured by the adaptation of the request of the user to their preferences.

According to all steps of this section, we can see the use of the ontology of the context "ContoLogy" for managing preferences and resolving Conflicts, in order to adapt the initial request of the user to his context of use and his profile, which includes his preferences.

10 Conclusion

The ubiquitous computing focuses on the use of two essential notions: user profile and context of use in order to satisfy better demands of nomadic users. Furthermore, a reliable modeling of such two notions and an adaptation of the application behavior to them are two required processes. In this paper, firstly, we presented a novel approach allowing, on one hand, modeling the context of use and the user profiles using an ontology, to support context representation and reasoning, and, on the other hand, resolving the conflicts using some proposed solutions. An architecture illustrating the dynamic adaptation of web service-based ubiquitous applications is also proposed. Secondly; we detailed a prototype implementation and system performance. Through this part in this paper, we tried to explain how we implement the web services, the ontology and shown up the adaptation process to resolve the conflicts by a detailed example.

As future directions to this work, we plan to:

1. Complete the implementation of the context acquisition module composed of two sub-modules: context sensor and context integration.

2. Use a probabilistic approach to represent the users' preferences. Because, it is a very complex challenge to represent the users' preferences with its contexts and the ambiguity posed by these ubiquitous applications. One of the considerations which generate abstraction data sources of information are cited for example: temporality, uncertainty, heterogeneity, online processing, and conflicting information. In the literature, several probabilistic (SVM, CPnet, HMM, HHMM, etc) are studied and we decide on the Hierarchical Hidden Markov Model (HHMM). HHMM is legible, easy for the preferences representation and does not require expertise in prior.
3. The cloud computing provides the next generation of Internet based, highly scalable ubiquitous computing systems in which computing resources are provided as a service. A new computing model that allows convenient access and on-demand network to a shared pool of configurable computing resources (eg, networks, servers, storage, applications and services) that can be rapidly provisioned. However, ubiquitous computing refers to a scenario in which computing is ubiquitous, particularly where devices that do not look like computers have computational capabilities. The idea is how to use cloud computing resources efficiently and earn maximum profits with ubiquitous systems?

Acknowledgement

This work summarizes doctoral thesis research, supported by the University of Annaba and the University of Tebessa- Algeria.

References

[1] Kosala.Y, MingXue.W, & Claus. P.2013. An extended ontology-based context model and manipulation calculus for dynamic Web service processes. Journal of Service Oriented Computing and Applications, ISSN 1863-2386. Springer-Verlag London.

[2] Rebei.I. 2012. Informatique ubiquitaire et pervasive . F2B506, Telecom Bretagne, 22 février.

[3] Dey A. K., Abowd. G. D., & Salber.D. 2001. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context- Aware Applications. Human-computer Interaction, 16 : 97–166,.

[4] Schmidt. A., Aidoo. K. A., Takaluoma. A., U. Tuomela, K. V. Laerhoven, & W. V.de Velde. 1999. Advanced Interaction in Context. In HUC '99 : Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing, pages 89–101, London, UK, Springer- Verlag

[5] Held.A, Buchholz.S, & Schill .A. 2002. A Modeling of Context Information for Pervasive Computing Applications. In: Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI), Orlando, FL, USA, Jul 14-18

[6] Soukkarieh. 2010. SOUKKARIEH Bouchra "Technique de l'internet et ses langages : vers un système d'information Web restituant des services Web sensibles au contexte. thèse Doctorat, Université de Toulouse III, France, 30 avril.

[7] Ryan. N. 2006. ConteXtML: Exchanging contextual information between a mobile client and the fieldnoteserver. Httpwww.cs.kent.ac.uk/projects/mobicomp/fnc/ConteXtML.html.

[8] Sheng .Q. Z & Benatallah. B .2005.ContextUML: A UML Based Modeling Language for Model- Driven Development of Context-Aware Web Services. In The 4th International Conference on Mobile Business(ICMB'05), IEEE Computer Society. Sydney, Australia. July 11-13.

[9] Henricksen.K and Indulska. J. 2004.Modelling and Using Imperfect Context Information. In PerCom Workshops, pp 33–37.

[10] Chevert, K., Mitchell, K., & Davies, N. 1999.Design of an object model for a context sensitive tourist GUIDE. Computers and Graphics 23, 6 883–891.

[11] Schmidt, A., Beigl, M., & Gellersen, H.-W. 1999. There is more to context than location. Computers and Graphics 23, 6, 893–901.

[12] Akman, V., & Surav, M. 1997. The use of situation theory in context modeling. Computational Intelligence 13, 3 427–438.

[13] Chahuara P. 2013. Contrôle intelligent de la domotique à partir d'informations temporelles multi-sources imprécises. Thèse doctorale. s.l., France : Université de Grenoble, 27 mars.

[14] Miao. LV, Chun.JIN, Yoshiyuki.H, & Jim. C. 2013. Ontology-based User Preferences Bayesian Model for Personalized Recommendation. Dalian University of Technology. China, Fukushima University. Japan, Florida Atlantic University .USA, Journal of Computational Information Systems 9: 16 6579–6586.

[15] Strang.T, Linnhoff-Popien.C, & Frank. K. 2003. CoOL: A Context Ontology Language to enable Contextual Interoperability. In J.-B. Stefani, I. Dameure, and D. Hagimon, editors, LNCS 2893 : Proceedings of 4th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems (DAIS2003), volume 2893of Lecture Notes in Computer Science (LNCS), pp 236–247, Paris/France, November. Springer Verlag

[16] Chen, H., Perich, F., Finin, T., & Joshi, A. 2004. SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications. International Conference on Mobile and Ubiquitous Systems: Networking and Services, Boston, 22-25 August

[17] Gu, T. et al. (2004) "An ontology-based context model in intelligent environments". Proceedings of Communication Networks and Distributed Systems Modelling and Simulation Conference, San Diego (CA), USA.

[18] Chen, H., Finin, T. and Joshi, A. 2003.Using OWL in a Pervasive Computing Broker. In Proceedings of Workshop on Ontologies in Open Agent Systems (AAMAS) .

[19] Chen, H., Finin, T. and Joshi, A. (2004) "An ontology for context aware pervasive computing environments", Knowledge Engineering Review, Vol. 18, No. 3, pp.197–207.

[20] Kanellopoulos D. (2008) "An ontology-based system for intelligent matching of travellers' needs for airlines seats", International Journal of Computer Applications in Technology, Vol. 32, No.3, pp. 194-205.

[21] Kanellopoulos D., Panagopoulos A. (2008) "Exploiting tourism destinations' knowledge in an RDF-based P2P network". Journal of Network and Computer Applications (Elsevier Science), Vol. 31, No. 2, pp.179-200.

[22] Kanellopoulos D. (2009) "Adaptive multimedia systems based on intelligent context management", International Journal of Adaptive and Innovative Systems, Vol. 1, No.1, pp.30-43.

[23] Carrillo R.A. 2007.Agents ubiquitaires pour un accès adapté aux systèmes d'information : Le Framework PUMAS. Thèse pour obtenir le grade de docteur de l'université joseph fourier Spécialité : Informatique, préparée au Laboratoire l'Informatique de Grenoble présentée et soutenue publiquement le 5 mars.

[24] Belhanafi N. 2006. Ajout de mécanismes de réactivité au contexte dans les intergiciels pour composants dans le cadre d'utilisateurs nomades. Thèse présentée pour l'obtention du grade de Docteur de l'Institut National des Télécommunications Soutenue le 27 Novembre.

[25] Chaari.Tand Laforest. F. 2006. Adaptation in Context-Aware Pervasive Information Systems : the secas project, journal of pervasive computing and communications, vol.2, no. 2, june 2006. received: august 2 2005; revised: january 27.

[26] W3C. 2004. Recommendation W3COWL, 2004. http://www.w3.org/TR/owl-ref/

[27] Doulkeridis.C, Loutas.N, &Vazirgiannis.M .2006. A system architecture for context aware service discovery. J Electron Notes Theoretic ComputSci, pp 101–116.

[28] Kapitsaki G, Kateros D, Prezerakos G, & Venierris I. 2009. Model driven development of composite context-aware web applications. J InformSoftwTechnol 51:1244–1260

[29] Goslar K, & Schill A. 2004. modelling contextual information using active data structures. In: Proceedings of the EDBT workshops.Lecture notes in computer science, vol 3268. Springer

[30] Farrar S, & Langendoen DT.2010. An owl-dl implementation of gold- an ontology for the semantic web. Journal of Linguistic modeling of Information and Markup Languages 40:45–66

[31] Wang X, Zhang DQ, Gu T, & Pung H. 2004. Ontology based context modelling and reasoning using owl. In: Proceedings of the 2ndannual conference on pervasive computing and communications workshops. IEEE

[32] Horrocks I, Patel-Schneider F. 2003. Reducing owl entailment to description logic satisfiability. The Semantic Web—ISWC 2003.Lect Notes ComputSci 2870:17–29

[33] Fernandez-Lopez. M & al. 1997. Methontology: from ontological art towards ontological engineering. In Proceedings of the AAAI97 Spring Symposium. Series on ontological engineering. Stanford, CA, (pp, 33-40)

[34] Zacarias, M., Caetano, A., Pinto, S., & Tribolet, J. 2005.Modeling Contexts for Business Process Oriented Knowledge Support. In :Althoff, K.D., Dengel, A., Bergmann, R., Nick, M., Roth-Berghofer, T. (eds.) : Proceedings of the 3rd Conference on Professional Knowledge Management - Experiences and Visions (WM 2005) (Kaiserslautern, Germany, April 10-13,2005), DFKI, pp. 389-396

[35] Belotti, R., Decurtins, C., Grossniklaus, M., Norrie, M.C., & Palinginis, A. 2004. Interplay of Content and Context. In : Koch, N., Fraternali, P., Wirsing, M. (eds.) : Proceedings of the 4th International Conference on Web Engineering (ICWE 2004)(Munich, Germany, July 26-30, 2004), Lecture

Notes in Computer Science, vol. 3140, Springer-Verlag, Berlin Heidelberg, pp. 187-200.
[36] Pittarello, F. 2005. Context-Based Management of Multimedia Documents in 3D Navigational Environments. Proc. of the 11th International Workshop on Multimedia Information Systems (MIS 2005). Lecture Notes in Computer Science 3665, Springer Verlag, 2005, pp. 146-162
[37] Sowa.j. 1984. Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley. A , Reading, MA
[38] Baader.F et al,. 2003. The description logic handbook theory, Implementation and Applications, Cambridge University Press.
[39] Driouch.R. 2007. Proposition d'une architecture d'intégration des applications d'entreprise basée sur l'interopérabilité sémantique de l'EbXML et la mobilité des agents. Thèse présentée pour obtenir le diplôme de Doctorat en science.
[40] Keita.A. 2007. Conception coopérative d'ontologies pré-consensuelles : application au domaine de l'urbanisme. Thèse pour l'obtention du diplôme de Doctorat à l'institut national des sciences appliquées, Lyon, Ecole Doctorale Informatique et Information pour la Société 209 pages.
[41] N. Noy, R. W. Fergerson M. & A.Musen. 2000 The knowledge model of Protégé2000: combining interoperability and flexibility. In: Dieng R, Corby O (eds) 12th International Conference in Knowledge Engineering and Knowledge Management (EKAW'00). JuanLes-Pins, France. Springer-Verlag, LNAI 1937, Berlin, Germany.
[42] Bachimont, B., J. Charlet& R. Troncy. 2004. Ontologies pour le Web Sémantique. Action spécifique 32 CNRS / STIC Web sémantique Rapport final.
[43] Troncy. Bachimont, B., J. Charlet& R.. 2004. Ontologies pour le Web Sémantique. Action spécifique 32 CNRS / STIC Web sémantique Rapport final.
[44] Ketfi. A., Belkhatir. N., P-Y Cunin. 2002. Adaptation Dynamique Concepts et Expérimentations. , In Proceedings of ICSSEA. In French.
[45] Kouadri Mostefaoui. S. & Hirsbrunner.B.2003. Vers une approche orientée contexte pour la découverte et la composition des services dans des environnements mobiles.
[46] Kadima. H. &Montfort.V. 2003. Les services Web: Techniques, démarches et outils XML, WSDL, SOAP, UDDI, Rosetta, UML, Dunod. Paris
[47] Ponge, J. 2004. comptabilité et substitution dynamique des web services. Mémoire de fin d'études, université Blaise Pascal Clermont II, juillet.
[48] Msdn. 2012 :. msdn.microsoft.com
[49] Protege 2000 Ontology Editor Home Page, http: //protege.stanford.edu
[50] w3c. 2012. www.w3.org.