وزارة التعليم العالي و البحث العلمي

جامعة باجي مختار عنابة

BADJI MOKHTAR-ANNABA UNIVERSITY
UNIVERSITE BADJI MOKHTAR ANNABA

Faculté des Sciences de l'Ingéniorat **Année 2015-2016**

Département d'Informatique

THESE

Présentée en vue de l'obtention
du diplôme de Doctorat en Sciences

PROFILS SÉMANTIQUE DES UTILISATEURS POUR L'ADAPTATION DES SERVICES, APPLICATIONS AUX SI UBIQUITAIRES

Option
Informatique

Par
Mme Bourougaa-Tria Salima

Directeur de Thèse
Mr Farid MOKHATI Professeur Université d'Oum El Bouaghi
Co-directrice de Thèse
Mme Hassina SERIDI Professeur Université Badji Mokhtar-Annaba

DEVANT LE JURY

Président : Mr Djamel MESLATI Professeur Université Badji Mokhtar-Annaba

Examinateurs : Mme Zizette BOUFAÏDA Professeur Université de Constantine
Mr Hamid SERIDI Professeur Université de Guelma

A la mémoire de mon père.
A ma mère.
A mon époux.
A mes enfants : Houssem, Nour, Taki et Iyed
A Mon frère et mes Sœurs.
A Maram.
A toute ma famille.

REMERCIEMENTS

En préambule, je souhaite adresser ici tous mes remerciements aux personnes qui m'ont apporté leur aide et qui ont ainsi contribué à l'élaboration de cette thèse.

Tout d'abord Mr Farid Mokhati, professeur à l'Université d'Oum El Bouaghi-Encadreur de cette thèse, pour l'aide et le temps qu'il a bien voulu me consacrer et sans qui cette thèse n'aurait jamais vu le jour.

Mes remerciements s'adressent également à Mme. Hassina SERIDI-BOUCHELAGHEM, Professeur à l'Université de Annaba, pour avoir Co-encadré ce travail. J'ai apprécié ses qualités humaines, ses compétences scientifiques et son esprit d'écoute qui me permettent d'atteindre mes buts en toute liberté

Mes remerciements s'adressent vivement à Monsieurs : Pr. Djamel MESLATI de l'université de Annaba, Pr. Zizette BOUFAÏDA de l'Université de Constantine et Pr. Hamid Seridi de l'université de Guelma, Membres du jury, pour l'honneur qu'ils m'ont accordé en acceptant de juger mon travail.

Je tiens aussi à remercier vivement mon mari, Mr. Tria Youcef, doctorant à l'Université de Annaba, qui n'a cessé de me prodiguer ses conseils et ses suggestions pertinentes.

Merci à toute ma famille, à ma mère, mon mari, mes enfants, mon frère, et à mes sœurs, qui m'ont soutenu en toutes circonstances. J'espère qu'ils trouvent ici l'expression de mon éternelle reconnaissance.

Enfin, j'adresse mes plus sincères remerciements à tous mes proches et amis qui m'ont toujours soutenue et encouragée au cours de la réalisation de cette thèse.

Saji

ملخص

في العقد الماضي، أصبحت الحوسبة في كل مكان (UC) طموح لمجتمع تكنولوجيا المعلومات. اليوم هي مهمة جدا، حيث أنه من المستحيل حاليا عدم وجودها على جدول الأعمال العالمي لأبحاث في الكمبيوتر. في الحوسبة في كل مكان ، الهدف الرئيسي هو مساعدة المستخدمين على الوصول إلى الخدمات والموارد في أي وقت، في أي مكان، وخاصة باستخدام الأجهزة النقالة (MD). التطبيقات في هذا المجال هي حساسة لسياق المستخدم. و يجب أن يكونوا قادرين على إدراك هذا السياق، والتكيف مع سياق الاستخدام وتفضيلات المستخدم. في الواقع، الوصول الآمن لنظم المعلومات، لمستخدمي الهواتف المتحركة من خلال مختلف الأجهزة واستجابات التكيف لسياق مستخدم المحمول وسياق الاستخدام، هما إشكاليتين مرتبطتين. نحن نحاول في هذه الأطروحة معالجة هذه القضايا بإقتراح نهج جديد تماما، يسمح ب : (1) تمثيل السياق وتفضيلات مستخدمي الهواتف المتحركة من خلال الأنطولوجيا (2) حل النزاعات التي قد تنشأ بين تفضيلات المستخدم و، (3) تطويع هذه التطبيقات إلى سياق الإستخدام وتعريف المستخدم. ويدعم هذا النهج برمجة التي قمنا بتطويرها. وتقدم دراسة حالة لإعطاء مزيد من التوضيحات.

الكلمات المفاتيحية: سياق الاستخدام؛ نبذة عن المستخدم.نظام المعلومات على شبكة الانترنت. البيئة البدوية. الأنطولوجيا. تفضيلات المستخدم. الصراعات.الحوسبة في كل مكان. خدمة ويب.

ABSTRACT

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 thesis, 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.

KEYWORDS : Context of use; User profile; Web-based Information System; Nomadic Environment; Ontology; User Preferences; conflicts; ubiquitous computing; web service.

Résumé

Dans la dernière décennie, l'informatique ubiquitaire (IU) est devenue une aspiration de la communauté informatique. Aujourd'hui, sa vulgarisation est importante qu'il est de plus en plus impossible de la distinguer de l'ordre du jour global de la recherche informatique. Dans (IU), l'objectif principal est de fournir aux utilisateurs la possibilité d'accéder à des services et des ressources à tout moment, n'importe où, en particulier en utilisant des appareils mobiles. Dans ce domaine, les applications sont sensibles au contexte. Elles doivent être capables de percevoir ce contexte et à adapter leurs comportements en tenant compte des données qui traitent le contexte d'utilisation et les préférences des utilisateurs. En fait, assurer l'accès des utilisateurs nomades aux systèmes d'information grâce à divers dispositifs, et l'adaptation des réponses aux profils des utilisateurs et aux contextes d'utilisation sont deux problèmes inséparables. Dans cette thèse, nous tentons de répondre à ces problèmes et nous proposons une nouvelle approche permettant essentiellement: (1) la représentation du contexte et des préférences des utilisateurs nomades grâce à une ontologie, pour soutenir la représentation du contexte et le raisonnement (2) la résolution des conflits qui peuvent surgir entre les préférences de l'utilisateur et, (3) l'adaptation de ces applications au contexte d'utilisation et le profil de l'utilisateur. L'approche est soutenue par un outil visuel, que nous avons développé. Une étude de cas est présentée pour donner plus d'illustrations.

Mots-clés: Contexte d'utilisation; Profil de l'utilisateur; Système d'information sur le Web; Environnement nomade; ontologie; Préférences de l'utilisateur; conflits; informatique ubiquitaire; service Web.

TABLE DES MATIERES

Liste des figures

Liste des Tableaux

INTRODUCTION GÉNÉRALE

1. CONTEXTE GÉNÉRAL

Actuellement, les utilisateurs du Web accèdent à une grande masse de données dans diverses situations via des dispositifs distincts. Pour avoir des réponses à leurs requêtes qui sont généralement très nombreuses, provenant de plusieurs sources d'information (hétérogènes et distantes) et ne sont pas toutes aussi intéressantes et pertinentes c'est-à-dire qu'elles ne répondent pas toutes aux souhaits de l'utilisateur. Cette complexité est augmentée si l'utilisateur devient nomade (utilisateur qui change fréquemment de localisation) et fait appel à des SIW (Système d'Information sur le Web) n'importe où et n'importe quand via des dispositifs mobiles distincts *(PDA*, téléphones, ordinateurs portables). Le changement de localisation par exemple, entraîne ainsi un changement dans les conditions de travail, et par conséquent, une modification du contexte général d'utilisation. [Kosala.Y, 2013].

Dans ce cas de figure, ces applications doivent prendre en compte la situation de l'utilisateur dite : situation contextuelle qui englobe le contexte d'utilisation ainsi que les préférences de l'utilisateur, puis, adapter tout le comportement à la situation en question pour rendre à cet utilisateur une réponse pertinente de point de vue contenu et temps, c'est l'idée sous-jacente à l'informatique ubiquitaire, où les applications sont sensibles au contexte « context-aware applications ». [REBAÏ, 2012].

Dans ce cadre, garantir l'accès par des utilisateurs nomades aux Systèmes d'Information (SI) à travers divers dispositifs, ainsi que l'adaptation de l'information aux préférences de l'utilisateur nomade et au contexte d'utilisation, sont deux problèmes liés entre eux. Néanmoins, quatre aspects peuvent être approchés par les questions suivantes :

- Comment percevoir le contexte de l'utilisateur?
- Comment modéliser le contexte de l'utilisation et le profil de l'utilisateur ?
- Comment résoudre les conflits qui peuvent surgir entre les préférences de l'utilisateur?
- Comment adapter le comportement de l'application de la sensibilité au contexte pour satisfaire les besoins de ces utilisateurs nomades?

2. CONTRIBUTIONS

Afin de répondre à ces questions, nous proposons, dans le cadre de cette thèse, une nouvelle approche qui permet essentiellement:

1. La modélisation du contexte d'utilisation et les préférences de l'utilisateur en utilisant une ontologie développée "Contology", en se basant sur une nouvelle définition du contexte qui sépare les données de l'application à partir des données contextuelles. L'ontologie est utile pour supporter la représentation du contexte et le raisonnement. Aussi, Les ontologies sont les plus expressives et les plus prometteuses pour la description du contexte dans un environnement sensible au contexte.

2. La résolution des conflits qui peuvent survenir lors de la gestion des préférences de l'utilisateur, par la modélisation des conflits et leurs solutions dans l'ontologie comme des règles en utilisant le langage de règles du web sémantique (SWRL).

3. Enfin, pour assurer l'adaptation fonctionnelle et dynamique des applications sensibles au contexte, une architecture basée sur le Web Service est proposée pour montrer l'efficacité de notre proposition dans le modèle de contexte.

3. ORGANISATION DU DOCUMENT

Le document est structuré en deux parties:

- Dans la première partie nous exposerons ***un état de l'art*** sur les notions : l'informatique ubiquitaire, la notion de contexte, les approches de modélisation du contexte et l'ingénierie ontologique. Elle est structurée en trois chapitres

Dans le premier chapitre, nous présentons ***l'Informatique ubiquitaire et la notion de contexte*** en deux sections. La première section introduit la notion de l'informatique ubiquitaire en trois sous-sections. La sous-section 1: « Définitions » résume les différentes définitions de l'informatique ubiquitaire, les différentes caractéristiques sont données en sous-section 2. Quelques applications de l'informatique ubiquitaire ont fait l'objet de la sous-section 3. La deuxième section de ce chapitre met l'accent sur la notion de contexte en

présentant ces définitions, la notion de son acquisition, son importance dans le domaine informatique et ces caractéristiques. Et enfin ce chapitre se terminera avec une conclusion.

Le deuxième chapitre se focalise sur ***un état de l'art des différentes approches de modélisation du contexte***. Ce chapitre est structuré en deux parties. La première partie présente la typologie des approches de modélisation du contexte, elle contient quatre sections. Après l'introduction, les approches paires/triplets font l'objet de la section 1, les approches orientées modèles (approches basées sur les langages de balisage, les approches graphiques et les approches orientées objet) sont présentées dans la section 2. Les approches basées sur la logique sont décrites dans la section 3. La section 4 est consacrée aux approches orientées ontologie. La deuxième partie présente une évaluation des approches citées, selon le domaine de l'informatique ubiquitaire et enfin ce chapitre s'achève par une conclusion.

Le troisième chapitre est réservé à ***L'ingénierie Ontologique*** Il est structuré en 10 sections. Dans la section1, la notion d'ontologie est explicitée. La section2 présente les constituants d'une ontologie. La section 3 explique l'utilisation des ontologies : différents besoins, la classification des ontologies fait l'objet de la section4, les formalismes de représentation des ontologies sont alors présentés dans la section5. Les langages de construction des ontologies, les éditeurs d'ontologies et les systèmes de raisonnement sur les ontologies font l'objet respectivement de la section6, la section7 et la section8. Finalement, les langages d'interrogation d'ontologies, les méthodes de construction d'ontologie sont détaillées dans les sections 9 et 10. Et nous achevons par une conclusion.

- Dans la seconde partie du document nous présenterons notre contribution. Elle est composée de deux chapitres.

Le quatrième chapitre présentera notre proposition pour ***la modélisation du contexte « ContoLogy »*** et détaille notre proposition pour l'adaptation dynamique des applications sensibles au contexte en utilisant les Web services qui est une ***EXPLOITATION DE «ContoLogy ».*** Il est divisé en 4 sections. Nous commencerons par énumérer les motivations pour les ontologies dans la section1. La section2 va faire l'objet de la représentation du modèle de contexte ensuite, nous présentons plus précisément le processus de construction de notre ontologie, qui modélise le contexte d'un utilisateur nomade et présente des solutions pour résoudre les conflits qui peuvent survenir entre les préférences de ce dernier, dans la

section3. Dans la section4, nous détaillons notre contribution pour l'exploitation de « ***ContoLogy*** » et l'adaptation des applications sensibles au contexte. Nous terminons par une conclusion.

Le cinquième chapitre détaille notre approche pour l'adaptation, en utilisant une étude de cas. Tout d'abord, nous présentons l'environnent de l'implémentation, qui explique l'environnement et les outils utilisés au cours de notre implémentation. Ensuite, nous détaillons notre étude de cas « **The Travel Booking Application** » et nous terminons par une conclusion.

Enfin, nous concluons ce travail et nous évoquons nos perspectives.

CHAPITRE 1

L'INFORMATIQUE UBIQUITAIRE ET LA NOTION DE CONTEXTE

INTRODUCTION

L'informatique ubiquitaire est un paradigme récent dont le but principal est de permettre aux utilisateurs de consulter des données n'importe quand, n'importe où, en utilisant des dispositifs mobiles (DM) différents. Les applications dans ce domaine sont sensibles au contexte et par conséquent doivent avoir la possibilité de percevoir ce contexte pour adapter tout le comportement de l'application à ce dernier, et ce, dans l'objectif de satisfaire les attentes de ces utilisateurs.

Dans le cadre de cette thèse, nous nous intéressons à la modélisation du contexte pour l'adaptation des applications sensibles au contexte dans un environnement de l'informatique ubiquitaire. Ces applications sont caractérisées par le fait qu'elles s'exécutent dans un environnement hétérogène et dynamique. Cet environnement est soumis aux caractéristiques variables des ressources, des dispositifs utilisés et à la mobilité des utilisateurs, ce qui rend nécessaire l'utilisation des applications sensibles au contexte qui détectent les variations de l'environnement et adaptent leurs comportements en conséquence.

Dans ce chapitre, nous présentons tout d'abord dans la section 1, le domaine de l'informatique ubiquitaire ainsi que les caractéristiques et quelques exemples d'applications dans ce domaine. La section 2 détaille la notion de contexte. Nous commençons par la définition du contexte, puis nous abordons la notion de son acquisition, ensuite nous mettons en évidence l'importance de son utilisation pour augmenter les possibilités des applications, tout en passant en revue les différents domaines où le contexte est utilisé et enfin nous détaillons les caractéristiques des informations de contexte.

1. L'INFORMATIQUE UBIQUITAIRE

1.1 Définition

Nous présentons dans cette section, sous forme chronologique, les différentes définitions de l'informatique ubiquitaire.

- Selon le *W3C* [W3C05], « l'*informatique ubiquitaire* est un paradigme émergeant de l'*informatique personnelle* (« *Personal Computing*») ».
- *Mark Weiser* en 1988 dans le centre de recherche *Computer Science Lab* de *Xerox Parc*, et dans son article fondateur [Weiser91] dit : *« The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it»*. Pour *Weiser*, trois éléments sont nécessaires: des ordinateurs bon marché à faible consommation électrique, des logiciels d'informatique ubiquitaire et un réseau qui lie tous les appareils. *Weiser* constate que : « *chaque personne sera entourée par de nombreux ordinateurs* », comme l'illustre la figure1-1. Selon Weiser, « L'informatique qui a pour but d'utiliser tous ces ordinateurs ensemble, s'appelle *l'informatique ubiquitaire* ».

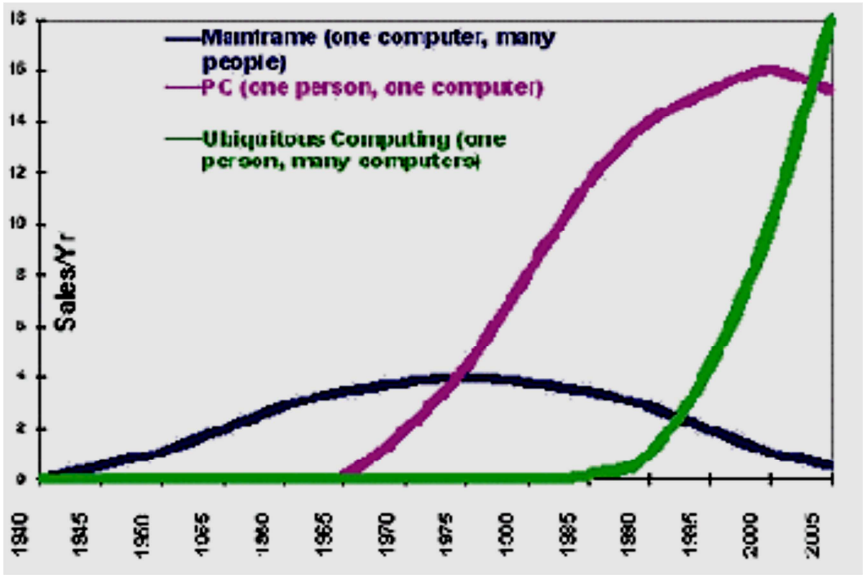


Figure 1.1 : Les grandes tendances en informatique ubiquitaire [[1]].

- Selon [Zaidenberg10], « *l'informatique sort de l'ordinateur pour s'intégrer directement dans l'environnement, qui devient donc ubiquitaire. Le but de cette idée est d'élargir les possibilités de l'informatique, de faire en sorte que l'ordinateur profite à l'utilisateur à tout moment lorsqu'il se trouve dans cet environnement. Pour aller plus loin, cet*

[1] Source : http://sandbox.xerox.com/ubicomp/.

ordinateur offre des capacités d'interaction plus naturelles, ce qui le rend transparent et utilisable sans effort pour les personnes ».

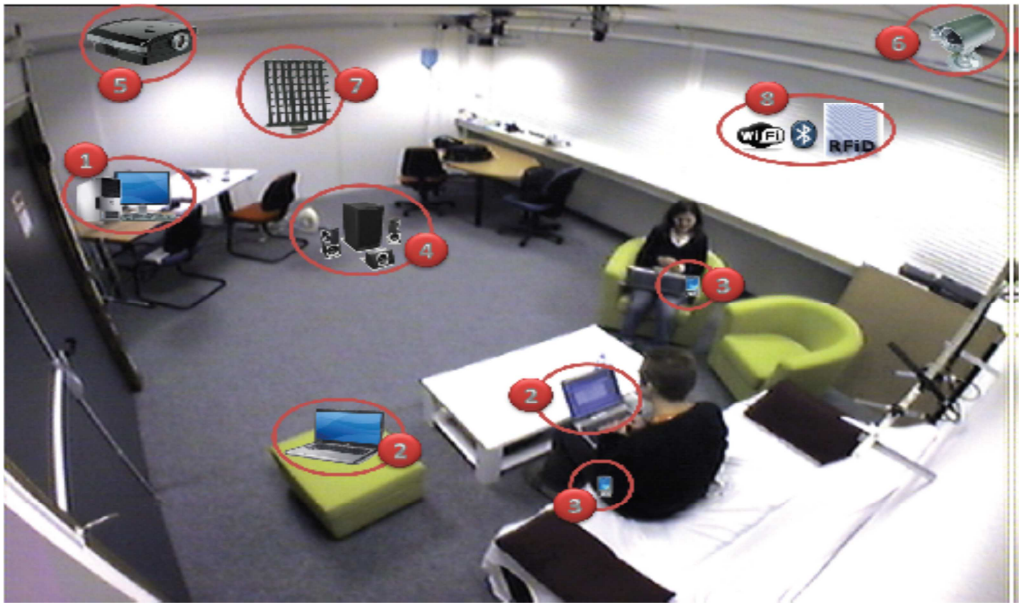


Figure 1.2 : Un exemple de bureau ubiquitaire: le Smart-Office de l'équipe prima [Zaidenberg10]

Un exemple d'un tel environnement est représenté par la figure1.2 qui nous montre un bureau ubiquitaire équipé de divers appareils :

1 Ordinateurs fixes 2 portables 3 Téléphones portables 4 hautparleurs 5 vidéoprojecteurs 6 Caméras 7 murs de microphones 8 antennes wifi, Bluetooth et RFID.

En conclusion et selon ces définitions, on peut conclure que**,** les applications ubiquitaires doivent permettre à leurs utilisateurs de consulter des données n'importe quand, n'importe où, à travers leurs dispositifs mobiles distincts.

1.2 Caractéristiques de l'Informatique Ubiquitaire

L'informatique ubiquitaire est caractérisée par plusieurs points, comme exemple :

- L'utilisation de dispositifs légers et sans fil. Pour Rahwan *et al.* [Rahwan04] ces dispositifs sont caractérisés par : « Le stockage limité, la puissance de traitement limitée, le besoin d'adjoindre des composants matériels embarqués par exemple : les capteurs de lumière, de son, de localisation, des lecteurs de code-barres, et le capteur GPS pour la détection de la localisation ».
- l'environnement ubiquitaire est caractérisé par la haute distribution, l'hétérogénéité et la mobilité.

- Koch *et al.* [Koch04] mentionnent que l'*informatique ubiquitaire* a introduit plusieurs changements :
    1. Un *changement d'infrastructure* qui sollicite la construction de technologies robustes matérielles et logicielles qui simplifient la connectivité mobile, l'identification de la localisation et la découverte de services, *etc*.
    2. Le *changement de services* qui s'intéresse à comment utiliser l'infrastructure disponible ? dont l'objectif est de fournir de nouveaux services à l'utilisateur.

1.3 Quelques Applications de l'Informatique Ubiquitaire

Parmi les applications de l'*informatique ubiquitaire* on peut citer les suivantes:

- L'infrastructure ***MyCampus de*** [Gandon04]. Elle est basée sur le Web Sémantique, sensible au contexte, en particulier aux préférences de confidentialité de l'utilisateur (« *privacy preferences*»). L'objectif de ***Mycampus*** est l'affichage des informations journalières d'un campus universitaire.
- Les applications ***WAY*** (« *Where Are You* ») [Collier03], ***Ad-Me*** (« ***Ad****vertising for* ***M****obile* ***e****-commerce user* ») [Hristova04] et ***Gulliver's Genie*** [O'Hare02] ont été réalisées pour assurer l'adaptation de l'information à la localisation de l'utilisateur, en prenant en considération les caractéristiques de son *DM (Dispositif Mobile)* (en général, des *PDA* et des téléphones portables).
    - ✓ *WAY* [Collier03] est un système destiné pour la synchronisation et la prise de rendez-vous entre utilisateurs nomades en se référant à leurs *PDA*.
    - ✓ *Ad-Me* [Hristova04] est un système qui utilise les informations sur la localisation et les préférences des utilisateurs nomades, pour leurs offrir des services de publicité. Par exemple : des magasins et des restaurants.
    - ✓ *Gulliver'sGenie* [O'Hare02] est un guide touristique qui se configure selon la localisation et l'orientation de déplacement de l'utilisateur et personnalise l'information en prenant en compte son profil (par exemple, *Gulliver's Genie* fait un plan de la ville en soulignant tous les ruines si les sites touristiques les plus visités par l'utilisateur sont des ruines).

En conclusion, les applications de L'*informatique ubiquitaire* doivent satisfaire l'utilisateur en lui fournissant une réponse appropriée coté contenu et temps. Pour cela, elles doivent prendre en considération la situation contextuelle de cet utilisateur, ensuite, adapter tout le comportement de ces applications à cette situation en cours. Dans ce

domaine, les applications sont sensibles au contexte « context-aware applications ». La section suivante est consacrée à la notion de contexte.

2. LA NOTION DE CONTEXTE

Afin d'utiliser efficacement la notion de contexte, il faut comprendre ce qu'est le contexte et comment l'utiliser. Définir ce qu'on entend par contexte est une question complexe, puisque cette notion est utilisée par plusieurs domaines de recherche.

2.1 Définition Du Contexte

Définir ce qu'on veut dire par le mot contexte, est très difficile. Dans la littérature, il existe plusieurs définitions de cette notion. On trouve que divers domaines de recherche le définissent. Chacun de ces domaines de recherche propose une ou plusieurs définitions qui ont une relation avec les recherches guidées. Dans cette section, nous allons présenter quelques définitions de la notion de contexte. Nous commençons par donner une définition littérale du terme, puis nous présentons plusieurs autres définitions utilisées dans le domaine de l'informatique.

- ***La définition littérale*** du mot contexte définit le contexte comme "*l'ensemble des unités d'un niveau d'analyse déterminé (phénomène, unité lexicale, phrase...) constituant l'entourage temporel ou spatial d'une unité "* [DictionnaireF]. Selon la même source, le terme contexte représente aussi " *un ensemble de circonstances liées à un événement qui se produit*".
- **Dans la langue française** aussi on peut trouver une autre définition du mot « contexte » comme suit: « *ce qui accompagne, précède ou suit un texte, l'éclaire* », ou encore comme « *ensemble des circonstances qui accompagnent un événement* » [Larousse92]. Selon [kirsch06], « *cette dernière définition est particulièrement intéressante pour l'informatique sensible au contexte, qui utilise la notion de contexte pour décrire les circonstances dans lesquelles un utilisateur accède à un système*. »
- **Dans le domaine informatique**, il existe plusieurs définitions, mais pour nos travaux, nous intéressons aux définitions de cette notion pour l'informatique sensible au contexte. Comme exemple on a :
  - ✓ **Selon Schilit** et **Theimer** [Schilit94a], Le contexte n'englobe que les informations sur : la localisation de l'utilisateur, les personnes qui l'accompagnent, les objets qui l'entourent, et les changements dans ces éléments. Pour Schilit *et al.* [Schilit94b] le

contexte est composé d'autres éléments, comme le niveau de bruit, le coût de communication.

- ✓ Une autre définition du contexte est donnée par **Brown** *et al*. [Brown97], qui le définissent comme un ensemble d'informations constitué de : la localisation de l'utilisateur, les personnes avec lui du moment de la journée.

Selon [Kirsch06], « *les premiers travaux restent plutôt vagues sur la définition de contexte, en accordant plus d'attention aux éléments qui peuvent décrire le contexte d'utilisation d'un système qu'à comprendre sa signification. La notion de contexte est pourtant le concept clé de la thématique de recherche dans laquelle ces travaux se situent.* »

D'autres définitions ont été proposées pour la notion de contexte, mais ces définitions restent très peu par rapport au nombre de travaux liés à l'informatique sensible au contexte. Parmi ces travaux on trouve:

- ✓ **Schmidt** et al [Schmidt99a] considèrent que le contexte est formé des informations relatives à un utilisateur et à l'état des dispositifs qu'ils utilisent.
- ✓ **Brézillon** et **Pomerol** [Brézillon99] considèrent que le contexte est formé de toutes les connaissances qui participent à la résolution d'un problème.
- ✓ **Mostéfaoui** et al [Mostéfaoui04] définissent le contexte comme toute les informations sur le centre d'intérêt de l'utilisateur et encore toutes les informations qui donnent des informations additionnelles sur ce centre d'intérêt.
- ✓ **Kirsh** [Kirsh01] à son tour définit le contexte comme : « *un amalgame très structuré des ressources informationnelles, physiques et conceptuelles qui va au-delà des simples questions de 'qui' ou 'quoi' est 'où' et 'quand'* ».

Selon [Kirsch06] toujours, « *Ces définitions présentent la notion de contexte comme un ensemble d'informations qui accompagne une action, laquelle, dans le cas de l'informatique sensible au contexte, est une interaction entre l'utilisateur et le système. Cependant, même si on comprend à travers ces définitions ce qui représente la notion de contexte, celles-ci sont encore trop abstraites pour être directement exploitables par ces systèmes.* »

En conséquence, d'autres auteurs ont proposé des définitions appropriées pour la conception de systèmes sensibles au contexte.

- ✓ **Pascoe** [Pascoe97] est l'un des premiers chercheurs qui a généralisé la notion de contexte en proposant la définition : « *le contexte est un sous-ensemble des états physiques et conceptuels ayant un intérêt pour une entité particulière* »,
- ✓ **Lemlouma** [Lemlouma04] considère le contexte comme « *l'ensemble de toutes les informations de l'environnement qui peuvent influencer le processus de l'adaptation et de la transmission du contenu vers l'utilisateur final* ».
- ✓ **Chaari** *et al.* [Chaari05] définissent le contexte comme «*l'ensemble des paramètres externes à l'application (localisation, de temps, d'environnement, de terminal utilisé, de profil utilisateur…) pouvant influer sur le comportement d'une application en définissant de nouvelles vues sur ses données et ses services. Ces paramètres ont un aspect dynamique qui leur permet d'évoluer durant le temps d'exécution. Une nouvelle instance de ces paramètres caractérise une nouvelle situation contextuelle qui ne modifie pas les données de l'application mais qui peut mener à les traiter d'une façon différente* ».
- ✓ La définition la plus générale et la plus utilisée est celle donnée par **Dey** [Dey00] qui décrit le contexte comme étant " *toutes les informations pouvant être utilisées pour caractériser la situation d'une entité. Une entité est une personne, un lieu, ou un objet qui peut être pertinent pour l'interaction entre l'utilisateur et l'application, y compris l'utilisateur et l'application*".

Selon [Mostéfaoui04], « *Cette définition de Dey concerne particulièrement la conception des systèmes sensibles au contexte, puisqu'elle tient compte de la pertinence des éléments pour l'interaction entre l'utilisateur et l'application. Mais cette définition reste générale : elle permet aussi bien l'inclusion des informations explicitement fournies par l'utilisateur à travers l'interface du système, que des informations implicitement fournies, lesquelles nécessitent la réalisation de certains calculs avant d'être effectivement utilisées (par exemple, l'information sur l'état actuel du réseau estimé à partir du nombre d'utilisateurs connectés)* »

Cependant, la généralité de la définition de **Dey** [Dey00] conclut un autre problème, qui est l'énumération des éléments qui composent la notion de contexte.

Pour **Chaari** *et al.* [Chaari05], « *la définition donnée par **Dey** ne permet pas d'identifier les données appartenant au contexte, ni de les distinguer des données propres à l'application* ». Cette séparation selon **Chaari**, « *est très importante, avant de commencer la*

conception d'une application sensible au contexte. Une donnée définie comme contextuelle dans un domaine peut être une donnée de l'application dans un autre domaine. »

Comme exemple, considérons un utilisateur en déplacement, qui veut avoir la liste des restaurants dans son voisinage. Cet utilisateur a des problèmes de santé et il doit suivre un régime alimentaire précis, alors la liste des restaurants est filtrée pour ne donner que ceux qui proposent des plats adéquats. La liste des restaurants et les plats qu'ils proposent forment les données de l'application. L'information « l'utilisateur a un problème de santé » est une donnée contextuelle extraite du profil de l'utilisateur. En effet, l'application doit être conçue en faisant abstraction à des données contextuelles. La séparation entre les données contextuelles et les données de l'application est également importante pour la modélisation du contexte.

En conclusion, pour définir les éléments qui constituent le contexte, nous avons adopté la définition donnée par T.Chaari [Chaari05], parce qu'elle nous semble pertinente et générique.

2.2 Le Processus d'Acquisition de Contexte

L'acquisition du contexte représente le processus de collecte des données à partir de l'environnement qui entoure l'application. L'opération de la collecte des informations de contexte est relative à la source du contexte et du type des informations collectées.

2.2.1 Méthodes pour l'acquisition de contexte

On peut obtenir le contexte, en utilisant différentes méthodes d'acquisition. Ces méthodes sont indépendantes de l'application. Selon Mostefaoui [Mostefaoui04], ces méthodes sont classées en trois catégories:

1. **Acquisition par sonde** : dans cette méthode, on utilise des capteurs (sondes) pour acquérir les informations de contexte. Il existe deux types de capteurs [Schmidt99a]: les capteurs physiques et les capteurs logiques. Un capteur physique est un dispositif, formé par des cellules sensibles, qui transforme des grandeurs physiques ou chimiques (variables d'instrumentation) aux signaux utiles pour les systèmes de mesure ou de contrôle (grandeur électrique). Les capteurs logiques utilisent des interfaces logicielles pour percevoir des informations associées à l'hôte où s'exécute l'application comme la bande passante et la charge de la batterie. Les informations contextuelles qui peuvent être détectées par les capteurs dépendent du type de capteur et ils peuvent être la température,

la distance, la localisation, l'accélération, l'inclinaison, le déplacement, la pression, l'humidité, le pH, etc.

2. **Acquisition par dérivation** : cette méthode consiste à obtenir un nouveau contexte dit de haut niveau, en dérivant ou interprétant un ou plusieurs informations de contexte, en appliquant des méthodes d'interprétation. Par exemple, la rue où se trouve une personne est un contexte de haut niveau calculé à partir des données de localisation en latitude et longitude collectées par un capteur GPS ;
3. **Acquisition par profil** : cette acquisition consiste à acquérir les informations contextuelles soit à travers une interface graphique soit par l'intermédiaire d'un fichier de profil.

2.2.2 Difficultés dans l'acquisition de contexte

Selon [Mostéfaoui04], « *l'acquisition du contexte à travers les capteurs n'est pas toujours facile, car l'information peut être acquise à partir de différents capteurs, comporter des erreurs, ou encore exiger une interprétation supplémentaire pour être exploitée par le système* ». De plus, les informations de contexte se changent fréquemment, et ça relativement à l'utilisateur, ce qui implique des difficultés dans l'acquisition de ce contexte dynamique. [Dey00] propose trois difficultés majeures qui concernent l'acquisition de contexte :

1. Le contexte est dynamique : alors les changements dans l'environnement doivent être détectés en temps réel et les applications doivent s'adapter à ces changements continus ;
2. Le contexte est capturé à partir de multiples sources, souvent hétérogènes et réparties ;
3. Le contexte est obtenu à travers la manipulation des périphériques non conventionnels (autres que la souris ou le clavier).

A son tour, Grudin [Grudin01] a soulevé trois problèmes qui concernent aussi l'acquisition du contexte :

1. ***Le respect de la vie privée :*** Ce problème arrive lorsque les informations contextuelles d'un utilisateur nomade deviennent publiques. Par exemple, certains travaux comme [Grudin01, Schilit94a, Weiser93] ont constaté que, dans le cas général, les utilisateurs n'acceptent pas que leurs informations contextuelles, comme la localisation et leurs préférences, deviennent public, même dans le cadre d'un travail commun.
2. ***La précision des données capturées :*** L'information contextuelle est de nature dynamique, et en plus, elle est collectée de différentes sources hétérogènes, ce qui entraine l'incertitude dans cette information et encore qu'elle soit erronée, ce point est consacré à la résolution de ce problème.

3. ***L'interprétation de ces données*** : L'interprétation des informations contextuelles brutes, collectées par différents capteurs, est une émission essentielle des systèmes sensibles au contexte, pour qu'elles soient utilisables pour adapter le comportement de ces systèmes au contexte en cours de l'utilisateur.

Selon [Kirsch06] : « *La manipulation des capteurs ne comporte pas, à ce jour, de techniques aussi bien développées et maîtrisées que la manipulation des périphériques traditionnels tels qu'un clavier ou une souris, pour lesquels il existe un large support au niveau des langages de programmation. Cependant, il faut remarquer que ce support ne tardera pas à apparaître, surtout avec l'évolution des standards et techniques. Ceux-ci s'intéressent à la communication et l'interconnexion des dispositifs (pas forcément des ordinateurs) dans un réseau* ».

En conclusion, et selon Dey [Dey00] : « *l'information contextuelle est manipulée d'une façon quasiment improvisée. Les concepteurs des systèmes sensibles au contexte choisissent les méthodes d'acquisition selon leur facilité et les technologies disponibles, en dépit de quelconque considération sur la réutilisation ou l'évolution des systèmes (par exemple, l'utilisation des nouveaux capteurs non prévus auparavant). Par conséquent, ces systèmes deviennent trop dépendants des technologies de détection. Ces systèmes, qui sont conçus pour adapter leur comportement au contexte, sont, paradoxalement, incapables de s'adapter aux changements dans le processus d'acquisition du contexte* ».

2.2.3 Exemples d'infrastructures pour l'acquisition de contexte

Dans cette section, nous allons présenter deux infrastructures pour l'acquisition de contexte. L'utilisation de ces infrastructures a facilité la conception des systèmes dans le processus d'acquisition.

- *ContextToolkit* [Dey00, Dey01c] représente une des premières infrastructures pour l'acquisition des informations contextuelles. Il propose une architecture de base constituée d'un ensemble d'éléments pour la capture et l'interprétation des informations contextuelles. *ContextToolkit* n'impose pas un modèle de représentation pour les données capturées, puisqu'il a pour objectif principal de simplifier l'évolution des systèmes sensibles au contexte, en utilisant une couche d'acquisition de contexte [Dey01c]. Cet outil isole les applications des méthodes d'acquisition réellement utilisées à travers la définition de cette couche d'acquisition, qui manipule directement les technologies (Voir Figure 1.3)

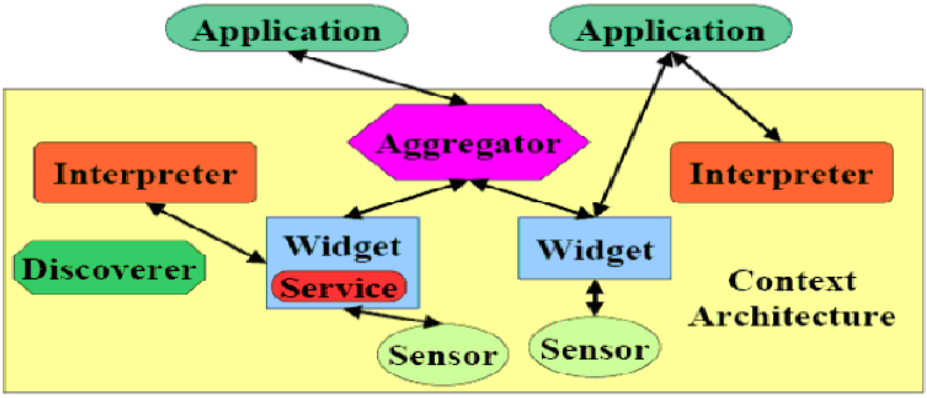


Figure 1.3 : L'architecture du ContextToolkit [Dey00].

- La deuxième infrastructure est proposée par Rey et Coutaz [Rey04]. L'idée de base pour cette infrastructure est l'assemblage des composants nommés « contexteurs». Selon ces auteurs, « *un contexteur est une abstraction logicielle qui fournit la valeur d'un élément de contexte* ». Un contexteur est généralement combiné à d'autres par différentes manières.

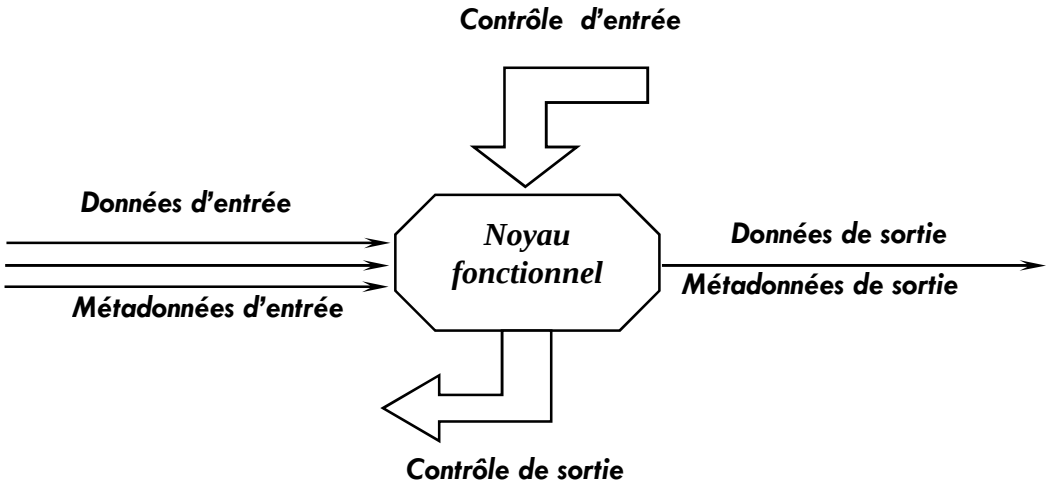


Figure 1.4 : La structure d'un contexteur [Rey04].

Chaque *contexteur* englobe trois classes d'éléments [Rey04] : les *entrées*, les *sorties* et un *corps fonctionnel* (voir Figure 1.4). Les *entrées* sont de deux types : les données d'entrée et le contrôle d'entrée. Les données d'entrée conviennent aux données que le *contexteur* a à sa charge. Le contrôle d'entrée permet à un autre *contexteur* ou à une application de modifier les paramètres internes du *contexteur*, modifiant le comportement de son corps fonctionnel.

En résumé, en utilisant ces infrastructures, on peut changer les technologies utilisées pour l'acquisition du contexte, car ce type d'infrastructure est fonctionnel en le combinant avec des représentations de contexte.

2.3 Importance du contexte dans Le domaine informatique

Actuellement la notion de contexte est utilisée par différents domaines de l'informatique. Par exemple : le traitement du langage, les systèmes de prise de décisions, l'informatique sensible au contexte et l'informatique ubiquitaire. Dans ce qui suit, nous présentons certains travaux de recherche qui s'intéressent à la notion de contexte dans les domaines cités ci-dessus.

2.3.1 Contexte dans le traitement du langage

Pour ce domaine, le contexte a un rôle très important dans l'interprétation correcte du langage humain, que soit pour un texte ou une parole. Il est utilisé spécialement dans la traduction automatique. Pour cette activité, le contexte est représenté par les phrases précédant un mot ou une autre phrase qui sélectionne la traduction correcte parmi les différentes traductions possibles [Berthouzoz99, Volk98].

2.3.2 Contexte pour l'aide à la prise de décision

A son tour, ce domaine utilise la notion de contexte dans le processus d'aide à la prise de décision, qui est lié à la situation dans laquelle il est utilisé. Plusieurs systèmes sensibles au contexte ont été utilisés pour prendre des décisions, en se basant sur cette notion. SIREN [Jiang04], un système sensible au contexte dans des situations d'urgences a pour but d'aider les pompiers pour prendre des décisions rapides dans un environnement de grande pression [belhanafi06].

2.3.3 Contexte Dans l'informatique Sensible Au Contexte

L'informatique sensible au contexte englobe les applications ou les systèmes qui utilisent le contexte pour adapter leurs comportements. En se basant sur les travaux de Dey et Abowd [Brown99], ceux de Schilit et al. [Schilit94c], et ceux de Brown et al. [Brown97], Chalmers [Chalmers04] a observé six utilisations possibles du contexte dans un environnement sensible au contexte :

1. La présentation et l'affichage des informations contextuelles par exemple: *le type du DM,*
2. L'attribution d'une information contextuelle à une donnée de l'application.

3. La configuration sensible au contexte, comme par exemple: *le fait d'effectuer une impression sur l'imprimante la plus proche, sans intervention de l'utilisateur,*
4. Lancement d'actions de réaction selon la valeur du contexte, comme *le chargement de la liste des restaurants d'une zone lorsque l'utilisateur se déplace vers cette zone,*
5. Médiation contextuelle, qui admet par exemple: *la modification d'un service offert, en utilisant le contexte*
6. Présentation sensible au contexte qui consiste *à utiliser le contexte pour modifier l'interface d'une application.*

Pour Dourish [Dourish04], « *la notion de contexte est centrale à l'informatique sensible au contexte, plus que pour d'autres thématiques de recherche en informatique qui utilisent cette notion (comme la recherche d'information), car les interactions dans un système sensible au contexte se trouvent dans des contextes d'utilisation très variés et souvent distincts de l'environnement du bureau mieux maîtrisé. C'est le contexte qui guide tout le comportement d'un tel système, ainsi que les informations qu'il fournit* ».

2.3.4 Contexte dans l'informatique ubiquitaire

Comme nous l'avons mentionné ci-dessus, les applications de l'informatique ubiquitaire doivent donner une réponse à la requête de l'utilisateur n'importe où et n'importe quand. Les applications dans ce domaine sont sensibles au contexte, donc la prise en compte des informations contextuelles par ces applications est primordiale.

2.4 Caractéristiques des informations de contexte

Les informations contextuelles sont des informations collectées à partir de différents sources hétérogènes. Pour cela, chaque contexte capturé peut être statique ou dynamique.

- ***Le contexte statique :*** est un contexte qui représente des informations qui ne changent pas au cours de l'utilisation de l'application dans toutes les sessions. Il suffit de les collecter au lancement de l'application. Parmi les informations de contexte statiques, nous citons, entre autres, le nom et le prénom d'un utilisateur.
- ***Le contexte dynamique :*** représente les informations contextuelles qui se changent fréquemment dans le temps. Par exemple, la localisation de l'utilisateur.

CONCLUSION

Nous avons présenté dans ce chapitre, tout d'abord, le domaine de l'informatique ubiquitaire ainsi que les caractéristiques et quelques exemples d'applications dans ce domaine. Ensuite, nous avons abordé la notion de contexte en commençant par la définition du contexte et son acquisition. Après, nous avons mis en évidence l'importance de son utilisation pour augmenter les possibilités des applications, tout en passant en revue les différents domaines où le contexte est utilisé. Enfin, nous avons détaillé les caractéristiques des informations contextuelles.

Pour conclure, un système sensible au contexte peut utiliser un modèle de description du contexte qui lui offre une description abstraite de ces informations de contexte afin de pouvoir les utiliser. Cette abstraction peut être fournie par des modèles de description du contexte**.** Le chapitre suivant présentera un état de l'art des approches de modélisation du contexte.

CHAPITRE 2

LES APPROCHES DE MODÉLISATION DU CONTEXTE

INTRODUCTION

La prise en compte de contexte est une action très importante pour une application sensible au contexte. Pour cela, une modélisation fiable de ce contexte est requise, parce que cette modélisation selon plusieurs chercheurs, représente l'étape préliminaire dans le processus de création des applications sensibles au contexte. Un modèle efficace permet à l'application de prendre en compte ce contexte, et changer son comportement. Dans la littérature, on trouve différents façons de modéliser le contexte d'une application sensible au contexte.

Dans ce chapitre, nous présentons les différentes approches de modélisation du contexte dans le domaine de l'informatique ubiquitaire. Il est divisé en deux parties : la première se focalise sur un état de l'art des différentes approches de modélisation du contexte. Elle est structurée en quatre sections. Après l'introduction, les approches paires/triplets font l'objet de la section 1, les approches orientées modèles (approches basées sur les langages de balisage, les approches graphiques, les approches orientées objet) sont présentées dans la section 2. Les approches basées sur la logique sont décrites dans la section 3. La section 4 est consacrée aux approches orientées ontologie.

La deuxième partie présente une évaluation des approches citées, selon des critères nécessaires pour l'informatique ubiquitaire, et selon d'autres critères qui représentent l'essence de notre approche et qui montrent aussi son apport par rapport à celles relatives à la modélisation de contexte étudiées dans ce chapitre.

1. TYPOLOGIE DES APPROCHES DE MODELISATION DU CONTEXTE

Cette partie décrit brièvement les principales approches de modélisation du contexte dans le domaine de l'informatique ubiquitaire/pervasive à savoir : les approches paires/triplets, les approches orientées modèles (les approches basées sur les langages de balisage, les approches graphiques, les approches orienté objet), les approches basées sur la logique, les approches orientées ontologie). Cette catégorisation se base sur la façon dont le contexte est modélisé, les outils utilisés à cet effet, l'expressivité du modèle et la possibilité de le réutiliser.

1.1 LES APPROCHES PAIRES/TRIPLETS

Ces approches modélisent le contexte en utilisent des structures de données assez simple pour représenter ce dernier

- *Schilit* [Schilit93] et *Theimer et Welsh* [Schilit94a] étaient les premiers qui ont modélisé le contexte en utilisant cette catégorie d'approches. Pour cela, ils proposaient d'utiliser un serveur d'environnement dynamique qui est chargé de l'observation d'un ensemble de contextes pour une application déterminée. Chaque observation du contexte est enregistrée dans une variable d'environnement constituée d'une paire clés/valeurs. La figure 2.1 représente une variable d'environnement qui permet de décrire les imprimantes les plus proches [Belhanafi06].

Nearest _ Printer=HP: HP4:HP6

Figure 2.1 : Acquisition par une variable d'environnement

- A son tour, *Schmidt* [Schmidt99b] modélise le contexte par le biais d'un triplet composé du contexte capturé, sa valeur, et un degré de certitude sur la cohérence des données capturées (voir figure 2.2).

C= (Location, Meeting Room, 95)

Figure 2.2 : Acquisition du contexte avec des triplets

- De son coté, [Samulowitz01] a proposé un framework pour gérer des services distribués « Capeus ». Il modélise le contexte en utilisant l'approche clé/valeur (Paires/Triplets).
- Aussi, [Pitoura94] a proposé un travail qui se focalise sur l'adaptation au changement du contexte, il utilise cette approche pour la modélisation de ce dernier.
- On trouve également le travail de [Maass97] qui utilise ce type d'approche pour modéliser des informations de localisation.

- Le Context toolkit de Dey [Dey01a] stocke le contexte comme un ensemble non structuré de paires clés/valeurs, par exemple, {Name="context1", User="doctor1", Location="hospital1", Time="t"}.

DISCUSSION

La modélisation paires/triplets utilise des structures de données simples à gérer, mais elle ne prend pas en compte ni les préférences de l'utilisateur ni la résolution des conflits qui peuvent survenir. Comme elle néglige l'expression des relations qui peuvent exister entre les informations de contexte ou la manière de déduire un contexte de haut niveau.

En conclusion, les paires de key-value sont faciles et de simple utilisation, mais ne sont pas assez robustes pour des modélisations de contexte efficaces.

1.2 Les Approches Orientées Modèles

1.2.1 L'Approches basées sur les langages de balisage

La structure hiérarchique des données modélisées est le point commun entre toutes les approches qui utilisent un langage de balises. Cette modélisation admet à utiliser un ensemble de balises avec des attributs et des contenus. Dans XML (eXtensible Markup Language), les balises sont déclarées dans un document appelé DTD (Document Type Définition). Le contenu des balises est défini récursivement par d'autres balises; un exemple d'utilisation de ce type de modélisation est la description des profils.

- Dans Pascoe [Pascoe97], on trouve une modélisation du contexte en utilisant un protocole d'échange de données contextuelles au format XML entre un serveur et un utilisateur nomade, pour le compte d'une application appelée Stick-e Note. Ce protocole nommé ConteXtML [Ryan06] a deux utilisations possibles. La première est de décrire le contexte acquis d'un utilisateur, la deuxième utilisation est de représenter le profil de l'application. La figure 2.3 montre un exemple d'un message qui décrit la position capturée de l'utilisateur. Cette position est représentée par les coordonnées x, y et z [Belhanafi06].

```
1  < context session= "123" action "update">
2        <spatial proj="UTM" zone="3" DATAUM="Euro 1950 (mean)"/>
3            <point x="281993" y="468790" z="205"/>
4       </spatial>
```

Figure 2.3 : Exemple de description de contexte avec conteXtML [Belhanafi06]

Malgré que cette modélisation est utile pour certaines applications, elle ne définit qu'un format d'échange de données et ne représente pas les contextes de haut niveau. Aussi, elle ne présente plus de solutions pour la résolution des conflits. De plus, elle ne décrit pas les relations entre les informations de contexte.

- Une autre approche dans cette catégorie est le CC/PP (Composit Capability/Prefrence Profile) de [Klyne04]. Il représente la proposition du W3C pour la représentation de profils. Il utilise RDF (Resource Description Framework) [Manola04] pour représenter les profils. Par l'utilisation d'une structure de profils, le CC/PP décrit les capacités d'un dispositif et les préférences de l'utilisateur. Cette structure est composée d'une hiérarchie à deux niveaux. Chaque profil a un certain nombre de composants et chaque composant a des caractéristiques. La figure 2.4 représente le profil d'un client.

```
1  < rdf : Description rdf :about= "http://www.example.com/profile#MyProfile">
2        <ccpp :component>
3           < rdf : Description rdf :about= "http://www.example.com/profile#TerminalHardware">
4              < rdf : type rdf :ressource= http://www.example.com/schema#HardWarePlatform " />
5               < ex :displayWidth>320</ex :displayWidth>
6               < ex :displayHeight>200</ex :displayHeight>
7               </rdf :Description>
8     </ccpp :component>
9   </rdf :Description>
```

Figure 2.4 : Exemple de description de contexte avec CC/PP [Belhanafi06]

Les chercheurs ont remarqué que CC/PP est pauvre en termes de vocabulaire et a besoin d'être étendu car il est destiné à la description de profil. De plus, il ne prend pas en compte la résolution des conflits qui peuvent survenir lors de la vérification des préférences. Aussi, il ne permet pas la description de relations et de contraintes complexes entre les informations de contexte [Belhanafi06].

- A leur côté**, Held et al**. [Held02] ont proposé CSCP (Compréhensive Structured Context Profile) qui utilise RDF/S pour représenter les structures naturelles d'informations de profil comme requis à l'information contextuelle. Cette proposition ne permet pas la résolution des conflits, ni de décrire des relations, des contraintes ou des dépendances entre les informations de contexte.
- Une approche semblable à CSCP est le CC / PP Contexte Extension Par **Indulska et al.** [Indulska03]**.** Les auteurs ont étendu le CC / PP et le vocabulaire de User Agent Profile UAProf [WAPF03] par un certain nombre d'arbres de composants et d'attributs reliés à

certains informations du contexte, par exemple: localisation, les caractéristiques du réseau et des informations sur la session. Cette approche est capable de permettre la sensibilisation de contexte aux applications et d'autres parties de l'infrastructure de l'informatique ubiquitaire.

- A son tour, [Chtcherbina03] a proposé PPDL (Pervasive Profile Description Language), un langage basé sur le XML. Il représente l'information contextuelle et les dépendances lors de la définition des modèles d'interaction. Ce langage présente quelques anomalies, il est limité en termes de nombre d'aspects contextuels évaluables et critères de conception. Seulement des parties du langage sont à la disposition du public. Pour cela, cette approche n'est pas pertinente pour la modélisation du contexte.
- Il y'a plusieurs autres approches de modélisation du contexte dans la catégorie des langages de balisage, par exemple, le contexte de configuration **Capra et al.** [Capra01] qui utilise le langage XML pour la modélisation du profil de l'utilisateur et le profil d'une application. Cette approche est utilisée par l'intergiciel CARISMA [Capra03] pour la reconfiguration d'une application ou l'adapter aux changements du contexte.
- Une autre approche de modélisation de contexte dans cette catégorie est le projet SECAS réalisé par T.Chaari et al [Chaari06]. Le projet SECAS (Simple Environment for Context Aware Systems) s'intéresse à l'adaptation des applications au contexte d'utilisation (préférences et environnement de l'utilisateur, terminal utilisé…). Dans le projet SECAS, les auteurs ont proposé une nouvelle définition de la notion de contexte qui sépare les données de l'application des paramètres du contexte. Ils ont utilisé une représentation basée sur RDF pour la modélisation du contexte (figure 2.5).

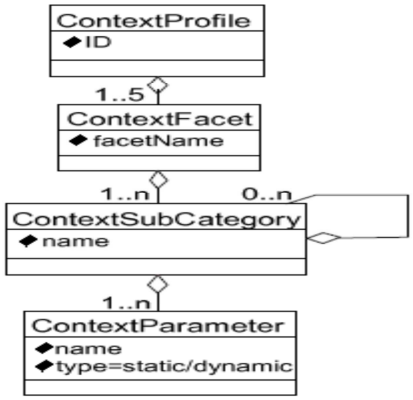


Figure 2.5 : Le modèle de contexte de SECAS [Chaari06]

- Une autre approche est celle proposée dans [Kanellopoulos08b] qui consiste en un modèle de métadonnées pour coder l'information sémantique des destinations touristiques, en

utilisant une architecture de réseau P2P basé sur RDF. Le modèle combine des structures ontologiques de l'information pour les destinations touristiques et les pairs. Dans ce travail, le modèle proposé est spécifique et très prometteur pour les destinations touristiques.

- Le travail de [Soukkarieh10] est une plateforme CA-WIS (Context Adaptation of Web Information System) qui est une démarche plus générale pour concevoir des Systèmes d'Information Web fournissant des Services Web sensibles au contexte. Ils ont intégré un modèle générique du contexte dans les deux parties de l'architecture de cette plateforme pour permettre à un utilisateur de recevoir d'une part des Services Web pertinents par rapport à son contexte, et d'autre part d'interagir avec un service sensible au contexte. Ce modèle générique concerne non seulement le contexte de l'utilisateur mais aussi le contexte du service.

DISCUSSION

Les modèles orientées balises sont réutilisables (grâce à la DTD qui permet des mises à jour au niveau des concepts utilisés) et simples à utiliser. Cependant, ils présentent quelques anomalies, comme par exemple : ces modèles ne prennent pas en considération : la résolution des conflits qui peuvent survenir lors de la vérification des préférences de l'utilisateur, la représentation des relations entre les informations de contexte et le type d'acquisition de ces informations.

1.2.2 Les approches graphiques

Un instrument de modélisation très connue est UML (Unified Modeling Language). Plusieurs approches graphiques utilisent UML pour modéliser le contexte. Nous pouvons citer entre autres :

- L'approche de **Bauer** [Bauer03] modélise les aspects contextuels relatifs à la gestion du trafic aérien sous la forme d'extensions d'UML. Mais, le modèle proposé est spécifique à l'application et ne peut pas être utilisé dans d'autres applications sans modification.
- De leur côté, Sheng et Benatallah [Benatallah05] ont proposé un méta-modèle qui se base sur une extension d'UML. Ce méta- modèle appelé *ContextUML* représente le contexte dont des services web sont sensibles. Cette modélisation est composée de plusieurs classes permettant la création des services sensibles au contexte (figure 2.6). La classe *Context* est composée de deux classes : *AtomicContext* et *CompositeContext*, elle modélise le contexte

capturé. Le contexte de bas niveau est représenté par la sous-classe *AtomicContext*, tandis que le contexte interprété est modélisé par la sous-classe *CompositeContext*. Pour ces auteurs, le contexte observé (capturé) peut être un contexte de bas niveau ou un contexte interprété. Malgré que le méta-modèle ContextUML est générique, il ne prend pas en considération : la résolution des conflits qui peuvent survenir lors de la vérification des préférences de l'utilisateur, la description des relations de dérivation et de dépendance entre les informations de contexte [belhnafi06].

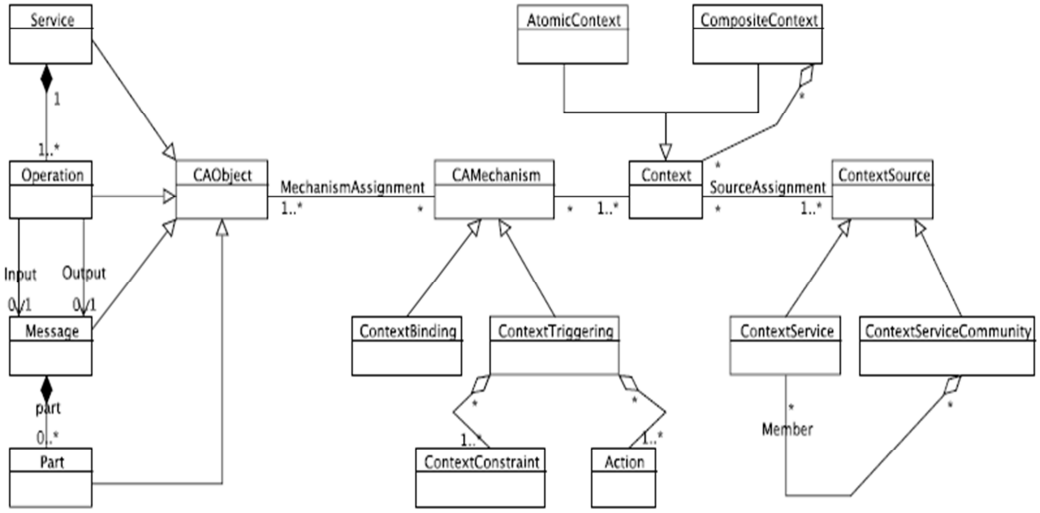


Figure 2 .6 : Le Méta modèle de ContextUML [Benatallah05]

- Un autre exemple au sein de cette catégorie, est une extension de l'approche ***Object-Role Modeling (ORM)*** [Halpin01], proposée par Henricksen et al. [Henricksen02, Henricksen03]. Dans ORM, le concept de base de modélisation est le fait (fact), et la modélisation d'un domaine en utilisant ORM implique l'identification des types appropriés des faits (facts) et les rôles joués par l'entité *type* dans ceux-ci. *Henricksen* [Henricksen02] a étendu ORM pour permettre aux types de faits d'être classés par catégorie, en fonction de leur persistance et source, soit comme statique (faits qui restent inchangés aussi longtemps que les entités qu'ils décrivent persistent), ou comme dynamique. La figure 2.7 illustre un exemple de la notation de Henricksen

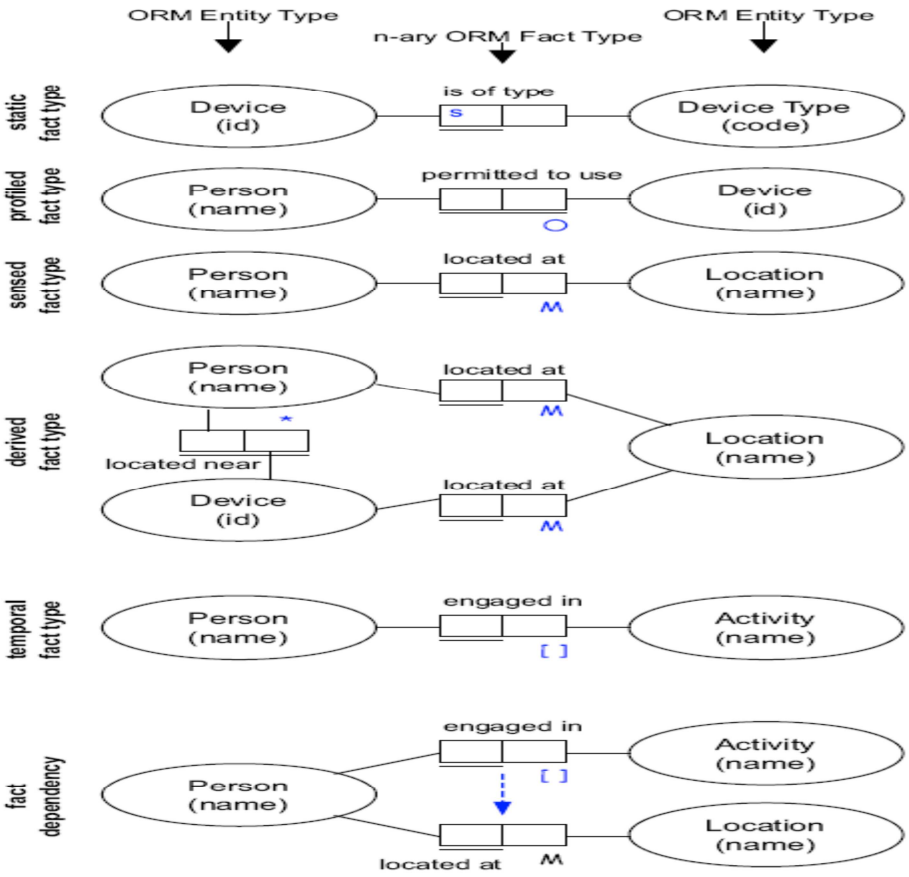


Figure 2.7 : L'extension de contexte ORM [Strang04]

DISCUSSION

Les modèles existants basées sur UML permettent de décrire des relations entre les informations de contexte mais ne prennent pas en considération la résolution des conflits qui peuvent survenir lors de la vérification des préférences de l'utilisateur. Aussi, elles ne tiennent pas en compte la description des dépendances entre ces informations, ni la qualité ou la validité temporelle des données décrites.

1.2.3 Les Approches orientée objet

Les approches orientée objet utilisent les principaux avantages de l'approche objet, à savoir, l'encapsulation et la réutilisation pour modéliser le contexte dans les environnements ubiquitaires. Les détails du traitement de contexte sont encapsulés dans un niveau d'objet et par conséquent caché à d'autres composants. L'accès à l'information contextuelle est fourni par le biais des interfaces spécifiées seulement.

- Un travail de haut niveau de ce type d'approches est celui réalisé par [Schmidt99b] dans le cadre du projet **TEA** [Schmidt01, TEA98]. Ce travail est basé sur le concept de cues qui fournit une abstraction des capteurs (sensors) physiques et logiques. Un cue est une

fonction prenant la valeur d'un seul capteur (sensor) physique ou logique comme entrée et donnant une sortie symbolique ou non symbolique. Un ensemble fini ou infini de valeurs possibles est défini pour chaque cue. La sortie de chaque cue dépend d'un seul capteur, mais différents cues peuvent être basés sur les mêmes capteurs. Les cues sont des objets qui offrent des informations contextuelles à travers leurs interfaces, mais ils cachent les détails de la détermination des valeurs de sortie [Strang04].

- Une autre approche au sein de cette catégorie est ***Active Object Model*** **du projet GUIDE** [Cheverst99]. Cette Approche a comme défi la gestion d'une grande variété de personnes et l'information contextuelle environnementale en gardant l'évolutivité.
- ***Bouzy et Cazenave*** [Bouzy97] ont proposé une approche d'utilisation des mécanismes généraux orientée objet pour représenter des connaissances contextuelles comme : le temps, l'objectif, les contextes spatiaux et globaux dans l'informatique Go (un jeu datant de 4000 ans qui est très célèbre au Japon, en Chine et en Corée). Pour ces auteurs, leur approche est appropriée à la modélisation du contexte orientée objet grâce à ses capacités d'héritage et de réutilisation, qui permettent de décrire le plus petit nombre de propriétés, fonctions et règles dont l'objectif est de simplifier la représentation de la connaissance dans des domaines et des systèmes très complexes.
- A leur tour, ***Stephen et al.*** [Yau01] ont proposé une grammaire pour définir un langage de description d'interfaces sensibles au contexte nommé CA-IDL. L'essentiel de ce langage est de rendre des applications réparties orientées objet sensibles au contexte. Pour cela, il permet, d'une part, la représentation des contextes dont l'application est sensible et, d'autre part, de définir les situations adéquates et les actions d'adaptation que l'intergiciel doit exécuter lors de leur découverte. [Yau04] a proposé une plate-forme qui fournit le langage CA-IDL. Elle prend en considération trois classes du contexte observé (capturé). Chaque catégorie est composée d'un certain nombre de types de contexte. La catégorie *DeviceSpecificContext* définit des informations de contexte propres à une machine. La catégorie *EnvironmentSpecificContext* définit des informations de contexte propres à l'environnement entourant l'application, et la catégorie *UserSpecificContext* définit des informations de contexte propres à l'utilisateur. Le langage CA-IDL est un langage qui permet de décrire non seulement les situations pertinentes de l'application mais aussi les actions d'adaptation et les conditions de leur déclenchement. Cependant, il reste limité, puisqu'il ne prend pas en considération la résolution des conflits qui peuvent survenir lors de la vérification des préférences de l'utilisateur. Par ailleurs, il ne permet pas l'ajout de

nouvelles sources de contexte sans modification de la grammaire de CA-IDL. [Belhanafi06].

CONCLUSION SUR LES APPROCHES ORIENTEES MODELE

Les approches orientées modèle sont caractérisées par l'utilisation d'un modèle formel pour décrire le contexte. Ces approches définissent le contexte d'une façon plus fiable que les approches paires/triplets. De plus, elles permettent aux concepteurs d'applications la réutilisation du modèle pour d'autres applications. La table 2.1 présente un récapitulatif des caractéristiques des approches orientées modèles selon des critères qui ont une relation directe avec ce que permettent ces modèles pour la sensibilité au contexte.

Table 2.1. Synthèse sur les approches orientées modèle

critères / Modèle	Modélisation du contexte Basé sur	Type de contexte				séparation des données contextuelles des données de l'application	La Résolution des conflits
		Profil de l'utilisateur	Le contexte de l'utilisation	Les conflits	Préférences de l'utilisateur		
ContextUML	UML	OUI	OUI	NON	NON	NON	NON
CML	ORM	OUI	OUI	NON	OUI	NON	NON
ConteXtML	XML	NON	OUI	NON	NON	NON	NON
CC/PP	RDF	OUI	OUI	NON	OUI	NON	NON
CSCP	RDF	OUI	NON	NON	OUI	NON	NON
PPDL	XML	OUI	NON	NON	NON	NON	NON
Le contexte de config. Capra	XML	OUI	OUI	NON	OUI	NON	NON
CA-IDL	**Gram-maire**	OUI	OUI	NON	NON	NON	NON
SECAS	RDF	OUI	OUI	NON	NON	OUI	NON
CA-WIS	XML	OUI	OUI	NON	NON	NON	NON
[Kanellopoulos.2008a]	RDF	NON	NON	NON	NON	NON	NON

D'après la table 2.1, nous constatons que la plupart des modèles ne permettent pas la gestion des préférences de l'utilisateur ainsi que la résolution des conflits qui peuvent survenir lors de la vérification des préférences de l'utilisateur. Aussi, aucun des modèles ne sépare entre les données contextuelles et les données de l'application sauf SECAS. La modélisation des préférences de l'utilisateur, la résolution des conflits entre les préférences et

la séparation entre les données contextuelles et les données de l'application nous semblent être des tâches très intéressantes pour satisfaire l'utilisateur.

1.3 Les Approches Basées Sur La Logique

L'utilisation de la logique permet la définition des conditions nécessitant, la déduction des faits ou des expressions en utilisant d'autres ensembles d'expressions ou de faits, ce qui est connu par le processus d'inférence. Pour ce faire, l'utilisation des systèmes formels semble inévitable. Dans un modèle de contexte basé sur la logique, le contexte est représenté sous la forme de faits, des règles et des expressions. Dans ce qui suit, nous allons présenter les approches les plus prometteuses de modélisation du contexte qui sont basées la logique

- une des premières approches de modélisation du contexte basée sur la logique a été publiée en tant que formalisme de Contexte au début de 1993 par **McCarthy** et son groupe à Stanford [McCarthy93, McCarthy94]. McCarthy définit le contexte comme une entité mathématique abstraite ayant des propriétés utiles en intelligence artificielle. Cette formalisation logique est basée sur une réification du contexte et un méta-prédicat **ist** ; ist (p, c) signifie que l'assertion p est vraie dans le contexte c. Par exemple la formalisation c : (ist (contextof ("Histoire de Sherlock Holmes"),"Sherlock Holmes est un détective") considère que le personnage Sherlock Holmes est un détective dans l'histoire de Sherlock Holmes ». [Belhanafi06]
- A leur tour, **Akman** et **Surav** [Akman97] ont proposé la Théorie de Situation Etendue (Extended Situation Theory). Cette approche étend la théorie de situation de Barwise et Perry [Barwise83] dont l'objectif étant de modéliser le contexte en utilisant des types de situation qui sont des situations ordinaires et des objets de première classe de la Théorie de situation. Ces derniers ont essayé de prendre en compte les aspects sémantiques des modèles-théoriques du langage naturel dans un système logique formel.
- L'approche de ***Giunchiglia*** [Giunchiglia93, Giunchiglia01], parfois désignée sous le nom des systèmes de Multicontext, est un travail qui a donné une grande importance au processus de raisonnement sur contexte qu'a la modélisation de contexte. Cette approche prend le contexte d'un sous-ensemble spécifique à un état complet d'une entité individuelle, et l'utilise pour le raisonnement sur un sujet donné.
- Une approche similaire est "Sended Context Model" proposée par **Gray et Salber** [Gray01]. Elle utilise des prédicats logiques du premier ordre comme représentation formelle des relations et des propositions contextuelles.

- Une autre approche au sein de cette catégorie est le système multimédia de **Bacon et** al. [Bacon97]. Dans ce système, l'emplacement (localisation) comme aspect du contexte est exprimé comme faits dans un système fondé sur les règles. Le système lui-même est mis en œuvre en Prolog.
- De son coté, [Chahuara13] a réalisé un contrôleur intelligent connecté à tous les périphériques de la maison intelligente. Ce contrôleur a pour objectif, de répondre aux demandes du résident (par commande vocale) et de découvrir les situations de risque ou de malheur. Pour ce faire, il a représenté et raisonné sur des informations temporelles et à des niveaux d'abstraction distincts. Le principal challenge est le traitement de l'incertitude, de l'imprécision, et de l'incomplétude qui caractérise les informations dans ce domaine d'applications qui sont sensibles au contexte. Le contexte pris en considération par cette approche est composé des informations de haut niveau telles que la localisation, l'activité en cours de réalisation, la période de la journée. L'auteur a divisé le travail réalisé en trois axes : l'accomplissement des méthodes d'inférence pour l'acquisition des informations contextuelles en utilisant les informations incertaines, la représentation des connaissances sur l'environnement et les situations à risque et, finalement, la prise de décision à partir des informations de contexte.
- Le travail de [Miao13] propose un modèle bayésien de préférences de l'utilisateur en fonction de l'ontologie (UPOBM) pour gérer le problème de préférences de l'utilisateur dans le domaine de recommandation traditionnelle personnalisée. Le modèle intègre la structure du réseau bayésien et la connaissance de l'ontologie pour exprimer les relations entre les contextes occasionnels, les caractéristiques de l'utilisateur et les préférences de l'utilisateur.

DISCUSSION

Les approches basées sur la logique sont caractérisées par le degré élevé de la formalité. Aussi, elles utilisent l'algèbre booléenne ou la logique du premier ordre pour la modélisation des informations contextuelles dont l'objectif est de raisonner sur les informations collectées. Cette modélisation ne décrit pas les relations qui peuvent exister entre les informations de contexte, mais elle offre beaucoup d'avantages pour le raisonnement sur le contexte et l'inférence de nouvelles actions de réaction si une situation semblable est découverte.

1.4 Les Approches Orientées Ontologie

Selon [Uschold96, Gruber93], « *une ontologie est une description sémantique, structurée et formelle des concepts d'un domaine et de leurs interrelations* ». Plusieurs approches basées sur les ontologies ont été proposées pour la modélisation du contexte. Ces approches permettent la modélisation du contexte et le raisonnement sur les données définies. Elles sont caractérisées par l'extensibilité et le partage des données. Dans cette section, nous présentons les approches de modélisation de contexte basées sur les ontologies, puis nous présentons une conclusion qui contient une synthèse des approches étudiées dans cette section.

1.4.1 Présentation des approches orientées ontologies

- **¨Otzturk" et Aamodt** [Otzturk97] étaient les premiers qui ont proposé une approche de modélisation du contexte basée sur les ontologies. Les auteurs ont mené une analyse des études psychologiques sur les points de distinction entre le rappel et la reconnaissance de plusieurs issues en utilisant des informations contextuelles. A partir de cette analyse, ils ont déduit l'obligation de la normalisation et la combinaison des connaissances de domaines distincts. L'approche consiste en un modèle de contexte basé sur des ontologies à cause de leurs forces dans le domaine de la formalité et la normalisation.
- Une autre approche prometteuse de modélisation de contexte basée sur les ontologies est le système **CoBrA-ONT** *(Context Broker Architecture ONTology)* [Chen03]. Ce système offre un ensemble de concepts ontologiques dont l'objectif était de définir des entités comme : les personnes, lieux ou autres concepts de contextes. **CoBrA-ONT** est une ontologie basée sur le langage OWL-DL. Elle est utilisée pour créer des applications multi-agents sensibles au contexte [Chen04a]. La création de cette ontologie était dans l'objectif de représenter le contexte des applications multi-agents gérant des salles de réunion intelligentes [Chen04b]. L'utilisation de **CoBrA-ONT** permet à chaque capteur (décrit comme un agent) de définir les données qu'il a capturées. Ensuite, il partage ces données avec les agents de chaque application. Cette opération de partage de données a offert la possibilité au système multi-agent CoBrA [Chen04c] d'induire de nouvelles informations de contexte et par la suite augmenter l'ontologie avec les données inférées. Par exemple, la détection de la présence d'un badge électronique dans une pièce, permet d'inférer que le possesseur du badge se trouve dans la même pièce. Cette modélisation ne prend pas en compte la gestion des préférences de l'utilisateur, ni la résolution des conflits

qui peuvent survenir entre ces derniers. Cependant, elle permet de définir les dépendances entre les contextes capturées à travers des propriétés ou par l'utilisation de la notion d'héritage. Par l'utilisation de TRIPLE [Sintek02] (un langage de règles qui permet de raisonner sur des ontologies), CoBrA-ONT permet d'inférer de nouvelles informations de contexte en raisonnant sur les données de l'ontologie.

- De sa part, [Strang03a] a proposé le modèle **Aspect-Scale-ContextInformation (ASC)**, qui a été réalisé en utilisant des langages d'ontologie appropriés. Le modèle ASC est une composition de trois ensembles : un ensemble d'aspects définissant le contexte capturé, un ensemble d'observations et un ensemble d'échelles. A son tour, l'aspect est composé d'un ensemble d'échelles, l'échelle est la combinaison d'un ensemble d'observations. Par exemple, l'aspect *Coordonnée Geographique* peut avoir deux échelles, l'échelle WGS84 ou l'échelle GaussKrueger. Une valeur capturée du contexte peut appartenir à l'une de ces deux échelles. L'objectif majeur de l'utilisation de ce langage est sa prise en considération de la sensibilité de contexte dans les frameworks de service répartis pour distinctes applications [Strang03c].
- Sur la base du modèle **ASC (Aspect-Scale-Context)**, [Strang03b] a proposé le langage d'ontologie **CoOL** (Context Ontology Language). Il est composé de deux parties, la première partie fait l'objet du cœur de **CoOL.** Elle indique une projection du modèle ASC, ce qui offre une classification des informations de contexte capturées selon l'échelle des données qu'elles définissent. Pour la deuxième partie de l'ontologie, elle définit les schémas d'intégration de cette dernière. Une collection de protocoles définit le schéma d'intégration, permettant à des systèmes ou des services web l'utilisation du cœur de l'ontologie. Ce cœur est réalisé en utilisant plusieurs langages d'ontologies : OWL [W3C04b], DAML+OIL [Pagels06] et F-Logic [Kifer89], ce qui simplifie son exploitation par différents types de plates-formes (figure 2.8). Dans **CoOL**, on trouve que les relations définies entre les informations contextuelles expriment la qualité de ces informations, mais il n'y a aucune prise en compte des préférences, ni résolution des conflits, ni la notion d'interprétation du contexte. **CoOL** est augmenté par des éléments d'intégration comme des extensions de structure pour des services Web [Strang03b, Strang03d].

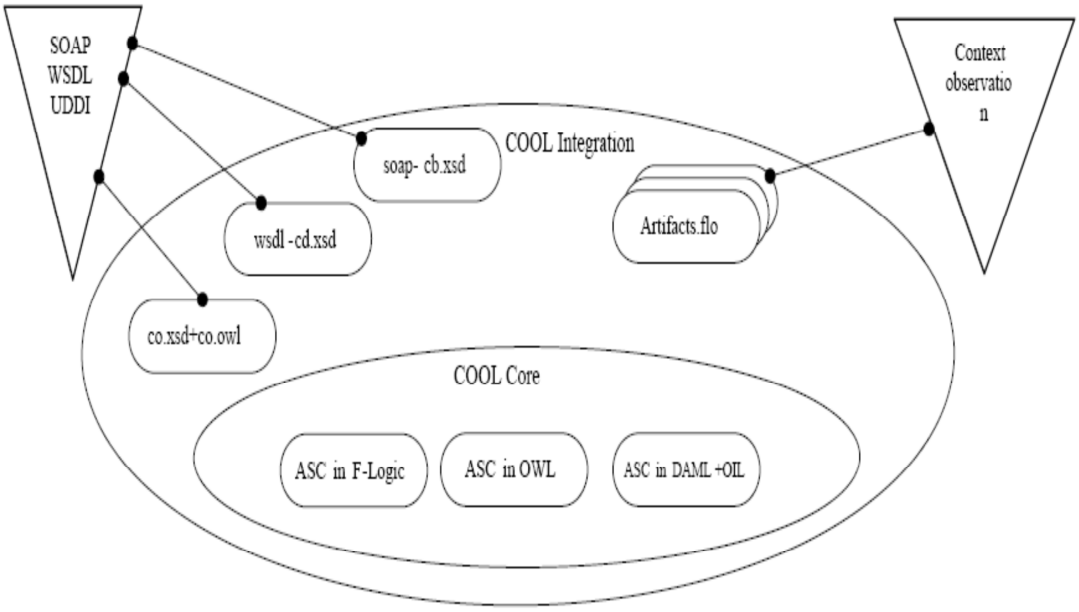


Figure 2.8 : Le langage d'ontologie CoOL [Strang03b]

- Une autre approche prometteuse de modélisation de contexte en utilisant les ontologies est ***CoNon*** (CONtext ONtology) [Wang04a]. Elle consiste en une ontologie extensible, développée à l'aide du langage OWL-DL qui est équivalant sémantiquement à la logique de description. Cette ontologie qualifiée de supérieure (an upper ontology) peut collecter les spécifications générales des entités de contexte de base et leurs caractéristiques pour chaque domaine et sous-domaine. ***CoNon*** est composée de deux niveaux. La définition des concepts généraux des informations contextuelles comme la localisation, les informations sur l'utilisateur, les informations sur des activités et des entités de calcul, qui sont en commun entre toutes les applications sensibles au contexte, est assurée par le premier niveau. Le deuxième niveau sert à la représentation des informations de contextes appropriés et spécifiques à un domaine donné [Wang04c], il est aussi extensible et peut rajouter à CoNon d'autres ontologies (figure 2.9). L'ontologie ***CoNon*** ne prend pas en considération les préférences de l'utilisateur et les conflits qui peuvent survenir lors de la vérification de ces derniers. D'après [Wang04b] la modélisation des capteurs qui collectent les informations contextuelles de contexte n'est pas aussi prise en charge. De plus, elle ne gère pas la description des règles d'interprétation. Encore, ***CoNon*** ne définit que le contexte et certaines relations existantes entre les informations contextuelles en additionnant la notion de qualité des données.

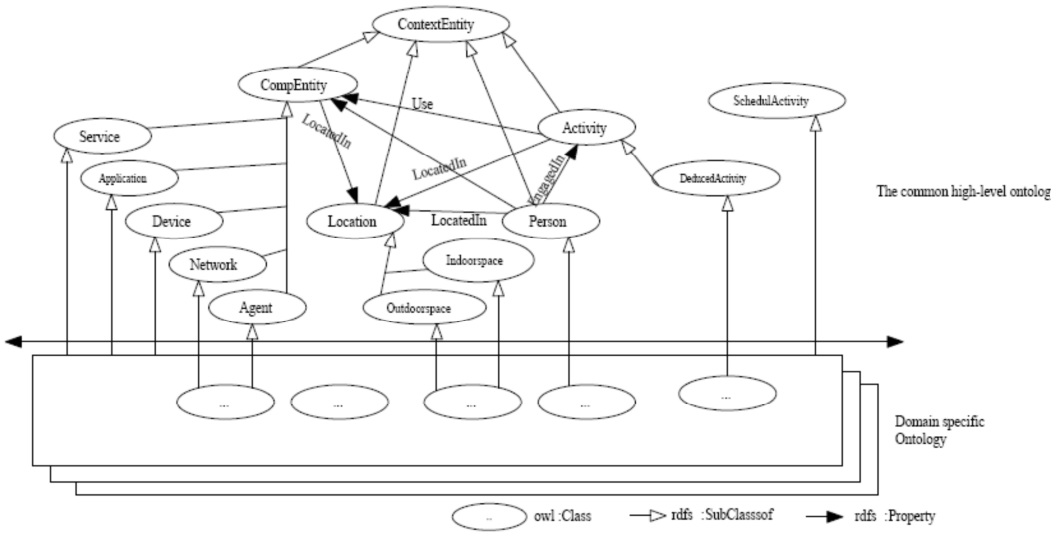


Figure 2.9 : Ontologie de contexte CONON [Wang04a]

- Toujours dans cette catégorie, on trouve l'ontologie **SOUPA** (Standard Ontology for Ubiquitous and Pervasive Applications) proposée par **H. Chen et al** [Chen04d]. L'objectif de SOUPA est la modélisation et le soutien des applications de l'informatique pervasive/ubiquitaire. SOUPA est définie en utilisant OWL (Web Ontology Language) et englobe des vocabulaires des entités modulaires pour définir les agents intelligents en prenant en compte : les intentions, les désirs, l'espace, le temps, les événements, les profils d'utilisateur, les politiques associés et les actions, pour assurer la sécurité et la confidentialité. *SOUPA* offre aux développeurs des applications de l'informatique ubiquitaire, une ontologie partagée qui combine plusieurs vocabulaires utiles à partir de distincts ontologies de consensus, ce qui aide ses développeurs ayant des insuffisances dans la définition des connaissances pour concevoir des applications basées sur les ontologies sans la définition des ontologies à partir de zéro. *SOUPA* se constitue de deux ensembles d'ontologies différentes (figure 2.10) : *SOUPA* Noyau et *SOUPA* extension. L'ontologie *SOUPA* ne prend pas en considération la gestion des préférences de l'utilisateur, ni la résolution des conflits qui peuvent survenir lors de la vérification de ces derniers. Aussi, *SOUPA* ne prend pas en compte la description des règles d'interprétation. Les ontologies qui sont mises en recommandation par SOUPA sont : *FOAF (Friend-Of-A-Friend ontology) [*Brickley03*] et [Powers03, DAML-Time&the Entry Sub-ontology of Time [Pan04], OpenCyc (Spatial Ontologies) & RCC (Regional Connection Calculus). Rei Policy Ontology.*

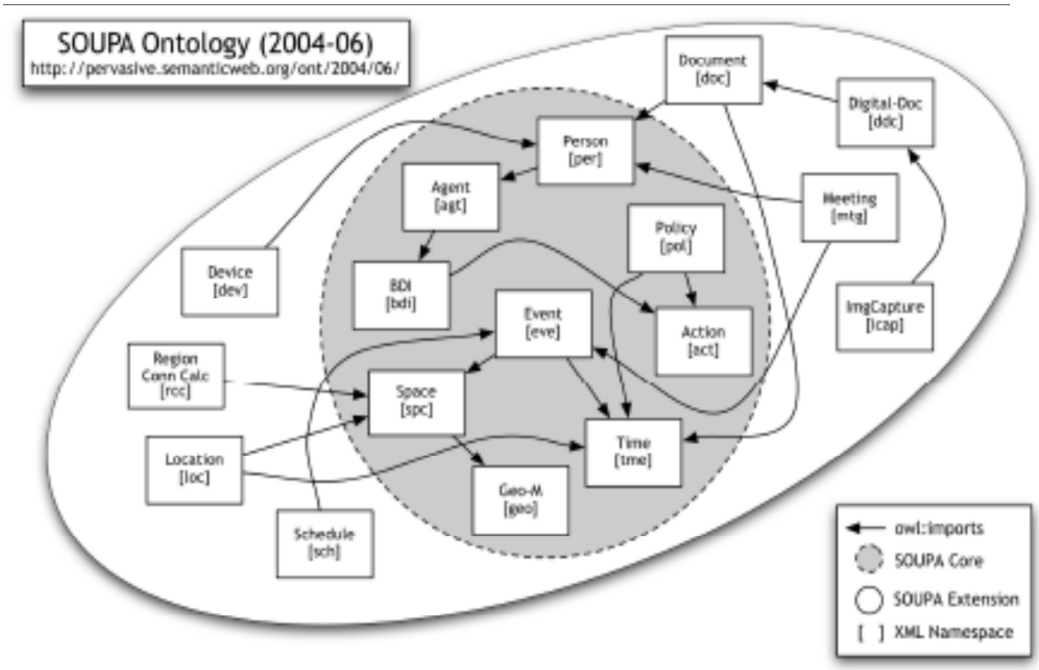


Figure 2.10 : L'Ontologie SOUPA [Chen04d]

- Une autre approche utilisant aussi les ontologies pour la modélisation du contexte est présentée dans [Gu04]. Ce travail propose un modèle formel de contexte basé sur une ontologie en utilisant OWL, pour résoudre les problèmes qui incluent : la représentation sémantique de contexte, le raisonnement sur le contexte, le partage des connaissances, la classification de contexte, les dépendances de contexte et la qualité de contexte. Le principal avantage de ce modèle est la capacité de raisonner sur les différents contextes basés sur le modèle de contexte.
- Une autre approche proposée dans [Carrillo07] utilisent aussi les ontologies pour la modélisation du contexte. Il s'agit de ***PUMAS*** (***P**eer **U**biquitous **M**ulti-**A**gent **S**ystem*), un *framework* basé sur des agents fournissant à l'utilisateur nomade une information adaptée à ses caractéristiques et à celles de son dispositif d'accès. Les connaissances de PUMAS sont définies en utilisant des ontologies, qui montrent la représentation commune de l'information partagée par les agents. Ce *framework* gère quatre ontologies qui définissent respectivement les caractéristiques liées à la session de l'utilisateur, son dispositif d'accès, ses préférences, et sa localisation. PUMAS a inspiré les concepts de chaque ontologies, des travaux d'Indulska *et al.* [Indulska03]. *PUMAS* prend en considération la gestion des préférences de l'utilisateur et propose aussi une solution pour résoudre quelques conflits qui peuvent survenir lors de la vérification des préférences. Cependant, PUMAS ne

permet pas la séparation entre les données contextuelles et les données de l'application. De plus, PUMAS est dédiée aux applications à base d'agents logiciels.

- Un autre travail est de [Kanellopoulos08a] dont l'objectif était de concevoir et de développer un portail web intelligent pour servir en tant que fournisseur de service dans les compagnies aériennes de voyage. Ce portail vise à aider les personnes vivant en Europe à trouver des places d'avion qui correspondent à leurs préférences de voyages personnels. A cet effet, les connaissances du domaine des compagnies de voyage ont été représentées en utilisant une ontologie. Ce travail porte sur la conception de l'ontologie et le développement partielle du portail web intelligent.
- Le but du travail décrit dans [Kanellopoulos09] est de fournir les éléments et les orientations pour définir et créer un profil d'utilisateur dans le domaine de multimédia. Afin de décrire le contexte multimédia, les auteurs ont utilisé une approche fondée sur les ontologies. Le modèle proposé est souple, car il fournit un Framework général qui peut être appliqué à chaque application multimédia. En outre, le modèle est extensible car il fournit des mécanismes pour ajouter de nouvelles spécifications de domaine multimédia.
- Le travail proposé dans [Miao13] est un modèle bayésien de préférences de l'utilisateur en fonction de l'ontologie UPOBM (**U**ser **P**references **O**ntology-based **B**ayesian **M**odel) pour gérer le problème de préférences de l'utilisateur dans le domaine de recommandation traditionnelle personnalisée. Le modèle intègre la structure du réseau bayésien et la connaissance de l'ontologie pour exprimer les relations entre les contextes occasionnels, les caractéristiques de l'utilisateur et les préférences de l'utilisateur.
- Kosala [Kosala13] propose un modèle de contexte basé sur les ontologies qui fournit un *framework* pour gérer les services d'approvisionnement, des aspects de consommation et des techniques pour gérer les contraintes de contexte pour le processus utilisé par le service web où le contexte dynamique peut être contrôlé.

1.4.2 Conclusion pour les approches orientées ontologie

Dans cette section, nous avons présenté les approches de modélisation de contexte basées sur les ontologies. La propriété principale de ces approches repose sur les spécifications et les avantages des ontologies. La table 2.2 résume les caractéristiques des approches orientées ontologie en fonction des informations qu'elles représentent et l'apport de chaque approche pour la résolution des conflits.

Table2.2. Caractéristiques des approches de modélisation du contexte orientées ontologie

critères / Approches	Type d'application	Type de contexte				séparation des données contextuelles des données de l'application	La Résolution des conflits	
		Profil de l'utilisateur	Le contexte de l'utilisation	Les conflits	Gestion Préférences de l'utilisateur		Prise en compte	Implication de l'utilisateur
CoBrA-ONT	à base d'agents logiciels	Oui	Oui	Non	Non	Non	Non	Non
CoOL	n'importe quel type d'appl.	Non	Oui	Non	Non	Non	Non	Non
CoNON	n'importe quel type d'appl.	Non	Oui	Non	Non	Non	Non	Non
SOUPA	n'importe quel type d'appl.	Oui	Oui	Non	Non	Non	Non	Non
PUMAS	Systèmes d'information sur le Web	Oui	Oui	Oui (quelques conflits)	Oui	Non	Oui (Quelques conflits)	Non
[Kanellopoulos 08a)]	les compagnies aériennes de voyage	Oui	Oui	non	Oui	Non	Non	Non
[Kanellopoulos 09]	le domaine de multimédia	Oui	Oui	non	Oui	Non	Non	Non
[Miao13]	recommandation traditionnelle personnalisée	Oui	Oui	non	Oui	Non	Non	Non
[Kosala13]	les services d'approvisionnement, les aspects de consommation	Oui	Oui	non	Oui	Non	Non	Non

D'après la table 2.2, nous pouvons constater que les ontologies *CoBrA-ONT, CONON, CoOL, SOUPA* étudiées dans cette section ne gèrent pas tout type de contexte. De plus, elles ne permettent pas la gestion des préférences de l'utilisateur, ni la résolution des conflits qui peuvent survenir lors de la vérification des préférences de l'utilisateur. Tandis que, les

ontologies de PUMAS prennent en considération la gestion des préférences de l'utilisateur et proposent aussi une solution pour résoudre quelques conflits qui peuvent survenir lors de la vérification des préférences. Néanmoins, PUMAS ne permet pas la séparation entre les données contextuelles et les données de l'application. De plus, PUMAS est dédiée aux applications à base d'agents logiciels.

2. EVALUATION DES APPROCHES

Dans cette partie, nous présentons une évaluation des approches de modélisation par l'utilisation de cinq critères généraux importants pour n'importe quelle approche de modélisation de contexte appliquée à un environnement de l'informatique ubiquitaire, en se référant à une étude faite par T. Strang et P. Linnhoff [Strang04].

2.1 Les critères généraux

Selon T. Strang et P. Linnhoff [Strang04] cinq critères sont notamment importantes pour n'importe quelle approche de modélisation de contexte appliquée à un environnement de l'informatique ubiquitaire.

2.1.1 Validation partielle **:** il est particulièrement important d'avoir la possibilité de valider partiellement les informations contextuelles et ça sur la structure avant le niveau d'instance, et ceci à cause de la complexité remarquée lors de la modélisation des relations contextuelles.

2.1.2 Richesse et qualité d'information **:** dans un environnement de d'informatique ubiquitaire, avec le temps, un capteur peut nous fournir des informations de qualité diminuée. De plus, la richesse des informations contextuelles caractérisant une entité qui sont offertes par des capteurs distincts peut ne pas être la même. Alors, le modèle de contexte adéquat à l'utilisation dans l'informatique ubiquitaire devrait soutenir l'indication de richesse et de la qualité.

2.1.3 Incomplétude et ambiguïté**:** la totalité des informations de contexte utilisables à n'importe quel moment qui caractérisent les entités adéquates dans les environnements de l'informatique ubiquitaire est généralement incomplet et/ou ambigu, en particulier si ces informations sont collectées des réseaux de capteurs. Pour cela, le modèle doit couvrir ce point.

2.1.4 Niveau de formalité **:** représenter des relations contextuelles et des faits d'une façon claire et précise, est toujours un grand défi. Par exemple, pour la tâche "imprimer le document sur l'imprimante près de moi", il est désirable de partager la même définition des termes utilisés dans la tâche, par exemple ce que signifie "prés" ou "moi". Alors, Il est conseillé que

toutes les parties d'un environnement de l'informatique ubiquitaire partagent la même interprétation des données échangées.

2.1.5 Applicabilité aux environnements existants : une perspective primordiale d'une implémentation est que le modèle de contexte utilisé soit accepté par les applications de l'informatique ubiquitaire existantes.

2.2 EVALUATION

Dans cette partie, et selon les travaux de T. Strang et P. Linnhoff [Strang04], nous allons présenter une évaluation des approches de modélisation de contexte selon les critères cités dans la section précédente.

2.2.1 Les approche paires/triplets

Les approches paires/triplets sont faibles sur les critères de 1 à 5 et inefficaces pour la modélisation des informations contextuelles complexes. Aussi, la simplicité des paires clés/ valeurs présente un inconvénient si le critère de la qualité de la méta-information ou celle de l'ambiguïté est pris en considération. Seulement l'applicabilité aux environnements de l'informatique ubiquitaire représente un point fort de cette approche.

2.2.2 Les approches orientées modèle

2.2.2.1 Les approches basées sur les langages de balisages

Ces approches satisfont le critère "validation partielle" grâce aux outils de validation et une définition de structure qui les supportent et qui peuvent être aussi utilisés pour contrôler les types simples et complexes. Même chose pour le critère « Incomplétude et ambiguïté ». Aussi, ces approches donnent une définition complète des structures, ce qui assure une formalité de haut niveau, qui peut être utilisée pour la détermination de l'interopérabilité. L'applicabilité aux infrastructures d'un environnement de l'informatique ubiquitaire (par exemple : les Web services) est un avantage de ce type d'approches.

2.2.2.2 Les approches graphiques

Les approches graphiques sont absolument robustes au niveau de la structure. Elles sont spécialement utilisées pour : définir la structure des informations contextuelles et inférer d'autres informations contextuelles de haut niveau (l'approche de Bauer) à partir du modèle de contexte, ce qui répond au critère « applicabilité aux environnements existants ». Le critère de " La Validation partielle" est possible, tandis que le critère "Incomplétude et ambiguïté" est négligé par Bauer, mais, il est pris en considération par Henricksen dans une version révisée de son modèle [Henricksen04].

2.2.2.3 Les approches de modélisation du contexte orienté objet

Ces approches satisfont les critères suivant : le critère de « validation partielle », le critère de « qualité de l'information » et le critère « incomplétude et l'ambiguïté ». Tandis que, l'approche TEA satisfait le critère « qualité de l'information », parce que le concept de cues contient un paramètre inférant la qualité du symbole -sortie du Cue-. Le critère « niveau de formalité » est également assuré par cette approche grâce à l'utilisation des interfaces bien conçues pour accéder au contenu de l'objet.

2.2.3 Les approches basées sur la logique

Pour cette classe d'approches, presque tous les critères ne sont pas vérifiés. Bien que leurs niveaux de formalité soient très élevés, le critère « validation partielle » est difficile à tenir par ces approches. Cela est dû à la définition des informations contextuelles qui utilisent ce type de modélisation et qui est très prédisposée aux fautes. Le critère de « la qualité d'information » n'est assuré par aucun modèle de cette catégorie. Même chose pour le critère « Incomplétude et ambiguïté ». En fin, L'applicabilité aux environnements de l'informatique ubiquitaire est un grand problème, car, en général, tous les raisonneurs de logique ne sont disponibles sur les dispositifs de l'informatique ubiquitaire.

2.2.4 Les approches basées sur les ontologies

Les approches de modélisation basées sur les ontologies satisfont largement le critère "validation partielle". On trouve que le modèle ASC permet de valider des types de données et aussi de valider les contenus des informations de contexte, en utilisant des gammes pour les informations contextuelles appelées *scales*. Tous les modèles de contexte fondés sur les ontologies sont robustes dans le domaine de la normalisation et la formalité, grâce à l'utilisation des ontologies. Le modèle ASC et le modèle de CONON prennent en considération la qualité de la méta-information et de l'ambiguïté, alors que ce n'est pas le cas pour l'approche CoBrA. Le critère de « L'incomplétude » est assuré par toutes les approches. Tandis que, le critère de « L'applicabilité à différents environnements de l'informatique ubiquitaire existants » est couvert seulement par le modèle ASC et dans le modèle CoOL.

Finalement, La table 2.3 récapitule l'évaluation des approches de modélisation du contexte pour l'informatique ubiquitaire par rapport aux critères définis précédemment.

Table 2 .3. Indication de Convenance [Strang04]

Approches- Exigences		Validation partielle	Richesse et qualité d'information	Incomplétude et ambiguïté	Niveau de formalité	Applicabilité aux environnements existants
Approches paires/triplets		-	-	-	-	+
Les approches Orientées Modèles	**Les modèles de balisages**	++	-	-	+	++
	les modèles graphiques	-	+	-	+	+
	Les modèles orienté objet	+	+	+	+	+
Approches basées sur la logique		-	-	-	++	-
Approches orientées ontologie		++	+	+	++	+

D'après la table 2.3, nous sommes parvenus à la conclusion que les approches les plus prometteuses pour la modélisation de contexte pour les environnements de l'informatique ubiquitaire en ce qui concerne les critères cités précédemment sont les approches basées sur les ontologies. Les représentants de cette catégorie ont mieux répondus aux critères.

SYNTHESE

Les informations contextuelles jouent un rôle crucial dans les différents domaines de l'ingénierie. Ces informations sont capturées à partir de distinctes sources. Dans le cas général, les informations capturées sont complexes (incorrectes, incomplètes, incohérentes ou obsolètes). Cette complexité exige l'utilisation des techniques de modélisation sophistiquées pour simplifier la manipulation de ces informations contextuelles. Comme il est avancé dans les sections précédentes, il existe différents approches pour la modélisation de ces informations de contexte. Réalisé par [Belhanafi06], La table 2.4 présente un résumé des caractéristiques de ces approches.

Table2.4. Caractéristiques des approches de modélisation du contexte [Belhanafi06]

Approches-Exigences	Caractéristiques	Richesse Sémantique	Facilité d'implantation	Raisonnement Sur le contexte	Résistance aux conflits
Approches paires/triplets	Simplicité d'utilisation Pauvreté d'expression	Non	Oui	Non	Non
Les approches Orientés Modèles	Utilisation D'un méta- modèle	Oui	Oui	Non	Non
Approches basées sur la logique	Très formelles	Non	Non	Oui	Non
Approches orientées ontologie	Utilisation d'un moteur d'inférence	Oui	Non	Oui	Oui

Finalement, l'objectif de l'état de l'art que nous avons effectué consiste à montrer l'apport de chaque approche de modélisation de contexte dans le but d'utiliser l'une d'elles pour modéliser les informations de contexte associées aux systèmes d'information ubiquitaires sensibles au contexte.

Notre choix pour l'approche de modélisation de contexte comporte plusieurs critères: l'expressivité du modèle, la richesse sémantique, la possibilité de sa réutilisation et de son extension. Selon la table 2.3 tiré de l'étude effectuée par Strang et Linnhoff-Popien [Strang04] et selon la table 2.4, résultat de l'étude effectuée par Nabiha Belhanafi [Belhanafi06], et selon les tableaux (2.1 et 2.2) effectués par nous-mêmes, nous pouvons conclure que l'approche orientée ontologie est l'approche la plus expressive et la plus prometteuse pour la description du contexte dans un environnement sensible au contexte, malgré l'inconvénient majeur de ce genre d'approches, qui est la complexité de l'exécution des ontologies et le poids de raisonnement sur leurs faits et leurs entités.

CONCLUSION

Dans la première partie de ce chapitre nous avons présenté les différentes approches de modélisation du contexte à savoir : les approches paires/triplets, les approches orientées modèles (les approches basées sur les langages de balisage, les approches graphiques, les approches orienté objet), les approches basées sur la logique, les approches orientées ontologie. L'objectif était d'étudier la possibilité d'utiliser ces approches dans les environnements de l'informatique ubiquitaire sensibles au contexte.

Ensuite, dans la deuxième partie, nous avons évalué ces approches selon la richesse et la qualité des informations qu'elles permettent de décrire, leurs degré de formalisme, et la possibilité de leurs utilisation dans différents types de systèmes.

L'évaluation des approches de modélisation de contexte a montré que les approches orientées ontologie sont les plus prometteuses pour la modélisation du contexte. De ce fait, nous avons opté pour cette approche. Nous détaillerons dans le chapitre suivant la notion d'ingénierie ontologique.

CHAPITRE 3

LES ONTOLOGIES ET LEURS CONCEPTS

INTRODUCTION

La présentation des approches de modélisation de contexte qui a fait l'objet du chapitre précédent , ainsi que l'étude critique menée sur différents critères , nous ont permis de choisir parmi ces approches une, qui est l'approche de modélisation de contexte orientés ontologies.

L'utilisation des concepts de l'ontologie, permet non seulement la représentation des informations de contexte, mais aussi, le raisonnement sur ces informations pour obtenir d'autres informations contextuelles de haut niveau, ceci est dû, aux caractéristiques des ontologies qui reposent essentiellement sur l'extensibilité, le partage et l'inférence des données.

Par conséquent, dans ce chapitre, nous allons détailler le concept des ontologies dans le but de montrer leur utilité et importance dans le domaine de la sensibilité au contexte. Tout d'abord, nous allons expliquer la notion d'ontologie dans la section1. La section 2 présente les constituants d'une ontologie. La présentation de l'utilisation des ontologies : différents besoins fait l'objet de la section3.

A son tour, la section 4 explique la classification des ontologies. Les formalismes de représentation des ontologies sont alors détaillés dans la section5. Les langages de construction des ontologies, les éditeurs d'ontologies et les systèmes de raisonnement sur les ontologies font l'objet respectivement de la section6, section7 et section8. Finalement, les langages d'interrogation d'ontologies, les méthodes de construction d'ontologie sont détaillés dans les sections 9 et 10.

1. NOTION D'ONTOLOGIE

Dans cette section, nous allons détailler la notion d'ontologie, en commençant par la présentation de l'origine et quelques définitions de cette notion.

1.1 Origine Des Ontologies

L'arrivée de l'Ingénierie des Connaissances (IC) a été l'occasion pour l'apparition des ontologies dans le domaine de l'Intelligence Artificielle (IA), et ceci, pour répondre aux problèmes rencontrés lors de la représentation et la manipulation des connaissances dans les systèmes informatiques.

La racine du mot **ontologie** est du grec :

- Onto (le participe présent du verbe être) qui est l'étude de l'être en tant qu'être ;
- Et logie qui signifie discours.

Les ontologies ont trouvé leur origine depuis le XIXième siècle en philosophie, où ils ont défini l'ontologie comme l'étude des propriétés générales de ce qui existe. Ce qui est équivalant à l'ensemble des connaissances que l'on a sur le monde [Welty01].

1.2 Définitions Informatique

Les ontologies sont nées du besoin de représenter les connaissances dans les systèmes informatiques, de ce fait, elles sont toujours définies par rapport au processus général de la représentation des connaissances [Fürst02]. Actuellement, différentes définitions sont offertes pour définir cette notion, mais, on trouve que la définition la plus adoptée par les chercheurs dans le domaine de (IC) est la définition de T. GRUBER,

- *«Une ontologie est une spécification explicite d'une conceptualisation»* [Gruber93].

Cette définition se base sur deux dimensions :

- Une ontologie est la conceptualisation d'un domaine, c'est-à-dire un choix de comment on définit un domaine.
- La spécification de cette conceptualisation, c'est-à-dire sa description formelle.

Alors, on ne construit pas l'ontologie qu'après que l'étape de conceptualisation soit achevée.

- Selon [Fürst02], N. Guarino [Guarino98] détaille la définition de T. Gruber en estimant les ontologies comme des spécifications partielles et formelles d'une conceptualisation : *"An engineering artefact, constituted by a specific vocabulary used to describe a certain reality, plus a set of explicit assumptions regarding the intended meaning of the vocabulary words. In the simplest case, an ontology describes a hierarchy of concepts*

related by subsumption relationships; in more sophisticated cases, suitable axioms are added in order to express other relationships between concepts and to constrain their intended interpretation".

- A son tour, Studer [Studer98] définit l'ontologie comme: « *An ontologyis a formal, explicit specification of a shared conceptualization* ». Ce qui veut dire « Une ontologie est une spécification formelle et explicite d'une conceptualisation partagée ». Studer l'explique comme suit : **« Formelle »** signifie que l'ontologie est explicable par une machine (machine-readable) ; **« Spécification explicite »** désigne que les concepts, les propriétés, les relations, les fonctions, les restrictions et les axiomes de l'ontologie sont définis de façon déclarative ; **« Partagé »** indique que l'ontologie prend la connaissance consensuelle (partagée par une communauté) ; **« Conceptualisation »** ce concept désigne l'abstraction d'un phénomène réel, l'ensemble des concepts importants qui caractérisent un domaine, et les points de vue d'une perception limitée du monde.
- Selon [Fürst02], on doit avoir la possibilité de construire une modélisation préliminaire semi-formelle et relativement cohérente qui correspond à une conceptualisation semi-formalisée. Donc, l'ontologie construite au cours de cette étape est une ontologie conceptuelle et semi-formelle, et le processus de spécification en question est appelé *ontologisation* [Kassel00]. Dans tous les cas, il est indispensable de traduire cette ontologie dans un langage opérationnel et formel de représentation de connaissances, pour qu'elle soit utilisable par une machine. Par conséquent, une ontologie n'est pas opérationnelle, si elle ne contient pas de mécanismes de raisonnement, n'a pas la possibilité de représenter les différents types de connaissances (connaissances terminologiques, faits, règles et contraintes) et ne peut pas manipuler ces connaissances à travers des mécanismes adaptés à l'objectif opérationnel du système conçu. Ce processus de traduction est appelé *opérationnalisation*.

En conclusion, une ontologie repose sur des définitions précises de concepts d'un domaine et de leurs relations. Ces définitions sont essentiellement acceptables et partageables par les utilisateurs et interprétables par des machines. Les constituantes de base de conception d'ontologie incluent :

- des classes ou des concepts,
- les propriétés de chaque concept définissant des caractéristiques diverses et les attributs du concept.

2. LES CONSTITUANTS D'UNE ONTOLOGIE

Cette section présente les constituants d'une ontologie, à savoir : les concepts, les relations, les axiomes et les instances. Dans ce qui suit, nous allons détailler chaque constituant.

2.1 Les Concepts

Un concept peut définir une idée, un objet matériel et une notion [Uschold95]. Un concept peut être fragmenté en trois parties :

- *Un terme* (ou plusieurs).
- *Une notion* : appelée aussi intention du concept, comprend la sémantique du concept, formulée sous la forme de *propriétés* et d'attributs, de contraintes et de règles.
- *L'ensemble d'objets*, appelé aussi extension du concept, englobe les objets maniés à travers le concept ; ce sont les *instances* du concept. Par exemple, le terme « Lion» nous fournit en même temps la notion d'objet de type « Animal » possédant un corps et quatre pieds, et à l'ensemble des objets de ce type.

Les concepts formant une ontologie sont distinctifs selon les types suivants :

- des connaissances de composition, par exemple : en médecine (catégories anatomiques), en chimie (catégories d'éléments) et en production (catégories de pièces) etc. ;
- des définitions complètes, par exemple : une personne est un client dans le cas où il passe aux moins une commande qui contient des produits proposés par l'entreprise ;
- des contraintes d'intégrité, par exemple : un étudiant possède un numéro unique dans l'ensemble des numéros de l'université ;
- des fonctions de calcul, par exemple : La glycémie normale à jeun est comprise entre à 0,70 grammes/L à 1,10 grammes/L. La glycémie après un repas oscille entre 1 et 1,4 g/L deux heures après un repas ;
- des propriétés algébriques, par exemple : la relation « est-remplacée -par» est symétrique, cela signifie que si la machine X est remplacée par la machine Y, alors le système peut aussi inférer que la machine Y peut être aussi remplacée par la machine X, et vice-versa ;
- des connaissances par défaut, par exemple : par défaut un animal a quatre pieds;

Selon [Fürst02] :

- un concept peut posséder les propriétés suivantes : la généricité, l'identité, la rigidité, l'anti-rigidité et l'unité.

- Les propriétés supportant sur deux concepts sont : l'équivalence, la disjonction et la dépendance.

2.2 Les relations

Une relation assure la liaison entre les instances d'un concept, ou des concepts génériques. Elles sont caractérisées par :

- *un terme* (voire plusieurs).
- *une signature* qui définit le nombre d'instances de concepts que la relation lie.
- *leurs types et l'ordre des concepts*, ce que veut dire la manière dont la relation doit être lue. *Par exemple*, la relation « parent-de » lie une instance du concept « parant » et une instance du concept « fils », dans cet ordre.

Selon [FÜRST02],

- Les propriétés intrinsèques à une relation sont : les propriétés algébriques, la cardinalité.
- Les propriétés liant deux relations sont : l'incompatibilité, l'inverse, et l'exclusivité.
- Les propriétés liant une relation et des concepts sont : le lien relationnel et la restriction de relation.

En conclusion et selon [Fürst02], « les propriétés des concepts et relations accomplissent la sémantique différentielle de l'ontologie, au sens où elles soutiennent la précision des liens et la différences entre les primitives cognitives du domaine de connaissance ».

2.3 Axiomes (règles)

Selon [Fürst02], Les axiomes sont utilisés pour définir les assertions de l'ontologie qui seront pris après comme vrais. Cette détermination a pour objectif de décrire les significations des composants d'ontologie, les arguments de relations et les contraintes sur les valeurs des attributs.

2.4 Instances

Représente la définition extensionnelle de l'ontologie, par exemple les individus « Houssem» et « Nour» sont des instances du concept «User».

3. UTILISATIONS DES ONTOLOGIES : DIFFERENTS BESOINS

A leur actuel, les ontologies sont utilisés presque dans tous les domaines de l'Ingénierie des Connaissances. Dans cette section, nous allons présenter quelques domaines d'utilisation des ontologies, à savoir : la communication, l'ingénierie des systèmes, le web sémantique et la sensibilité au contexte.

3.1 Communication

L'utilisation d'ontologie peut mener à bien une communication, et ceci, grâce à la spécification explicite que l'ontologie offre pour un domaine, qui est représenté par un modèle normatif. En plus de ça, les ontologies assurent la consistante et élimine l'ambiguïté dans les représentations des connaissances appartenant à un domaine spécifique. En fin, les ontologies prennent en considérations les distinctes perspectives des utilisateurs. Quand les utilisateurs (qui ont différentes perspectives d'un domaine) partagent une ontologie, ils ont une perspective standard [Driouche07].

3.2 Ingénierie Des Systèmes

Le déploiement des systèmes à base d'ontologies a défini un profit à l'ingénierie de systèmes qu'on peut le résumé comme suit : (1) *La réutilisabilité*: car l'objectif d'une ontologie est la codification des informations spécifiques à un domaine d'une manière qui assure le partage et la réutilisation. (2) *l'acquisition des connaissances*: l'ontologie mène à bien l'acquisition des connaissances. (3) *la Sûreté*: l'ontologie facilite l'automatisation du processus de vérification de la consistance. (4) *Spécification:* l'ontologie assiste le processus de définitions des exigences et la définition des spécifications des systèmes [Uschold96].

3.3 Le Web sémantique

Le Web constitue un terrain idéal d'application des ontologies considérées en tant que spécifications partagées de connaissances, les pages Web représentant une masse de connaissances aussi énorme que disparate. Cette masse augmente sans cesse ainsi que le nombre d'utilisateurs qui veulent pouvoir trouver facilement les informations qu'ils y recherchent. L'éventail des thèmes traités dans les différentes pages Web est tel qu'une recherche syntaxique par mot-clés retourne quasi systématiquement des pages qui ne portent pas toutes sur le même domaine de connaissances. L'exploitation efficace des ressources du Web suppose donc que les moteurs de recherche puissent accéder à la thématique de chaque page, et à son sens [Fürst02].

3.4 Les Ontologies Dans Les Systèmes Sensibles Au Contexte

Pour ces systèmes les ontologies sont utilisées pour la modélisation du contexte dont ils sont sensibles (Voir chapitre2- Approches de Modélisation du Contexte). L'avantage de

l'utilisation des ontologies réside dans les caractéristiques même de ces derniers, qui offrent non seulement le moyen de faire des descriptions sémantiques en modélisant le contexte, mais aussi de raisonner sur les données décrites. Et aussi, les ontologies sont caractérisées par une possibilité d'extension et de partage des données.

4. CLASSIFICATIONS DES ONTOLOGIES :

Cette section présente la manière de classifier les ontologies. Dans la littérature, on trouve beaucoup de travaux qui ont proposé différents critères pour classifier une ontologie. Dans la suite, nous allons exposer quelques exemples :

- Selon [Uschold96], les ontologies varient selon trois axes : (1) *le degré de formalisme de la représentation de connaissances*: qui est sélectionné selon les contextes d'utilisation des ontologies, et ça pour éviter la contredit qui peut survenir entre, d'une part, la nécessité de formaliser qui est lié au partage des connaissances et à la réutilisabilité, et, d'autre part, l'obligation de construire une ontologie lisible pour les utilisateurs. (2) *l'objectif opérationnel :* qui représente l'interopérabilité entre les systèmes, la communication entre les utilisateurs, la résolution de problèmes et l'utilisation par un problème d'ingénierie comme la réutilisabilité de composants. (3) *et le sujet* : qui est composé du domaine de connaissance, les connaissances d'inférence et les connaissances attachées au modèle de représentation.
- A son tour [Gómez-Pérez04] a distingué deux classes d'ontologie, les ontologies légères (*lightweight ontologies)* et les ontologies lourdes (*heavyweight ontologies*). Il les définit comme suit: **(1)** *Les ontologies légères :* qui incluent des concepts englobant des propriétés organisées en taxonomies avec des relations conceptuelles (exemple :*Yahoo! Directory*); **(2)** *Les ontologies lourdes : qui* ajoutent aux ontologies légères des règles (axiomes) et des conditions (restrictions) pour éclaircir le sens. Les ontologies lourdes modélisent un domaine de manière plus profonde avec plus de restrictions basées sur la sémantique du domaine.
- [Gómez-Pérez04] a proposé, à son tour, une classification où les ontologies sont classées selon deux critères: **(1)** la quantité et le type de structure dans la conceptualisation, **(2)** le sujet de la conceptualisation comme suit :

- *Ontologies de représentation de connaissances* : qui modélisent les représentations fondamentales utilisées pour la formalisation des connaissances sous un modèle donné.

- *Ontologies générales ou communes* : qui modélisent les connaissances de sens collectif partageables et réutilisables d'un domaine à l'autre. Elles incluent un vocabulaire correspondant aux évènements, choses, espaces, fonctions, temps, comportements causalités, etc.

- *Ontologies de niveau supérieur (top-level, upper-model)*: qui modélisent les concepts très généraux auxquels les racines des ontologies de plus bas niveaux faudrait être reliées. Elles définissent des notions générales comme les notions d'objet, d'état, de propriété, de moment, de valeur, d'action, d'événement, d'effet et de cause [Sowa84].
- *Ontologies de domaine :* ont pour but la modélisation des connaissances réutilisables dans des domaines bien définis. Elles offrent des concepts et des relations qui permettent de couvrir les activités, les vocabulaires, et les théories de ces domaines. Leurs concepts sont généralement des spécialisations de concepts décrits dans des ontologies de niveau supérieur. Il existe déjà de nombreuses ontologies de domaine comme : ENGMATH pour les mathématiques [Gruber94] et MENELAS dans le domaine médical [Zweigenbaum99].
- *Ontologies de tâches : qui* modélisent les vocabulaires liés à une activité générique ou une tâche avec la spécialisation de certains termes des ontologies de niveau supérieur.
- *Ontologies de tâches de domaine :* ce sont des ontologies de tâches réutilisables dans un domaine précis, mais pas d'un domaine à l'autre et elles sont autonomes de l'application.
- *Ontologies de méthodes* : qui modélisent les définitions des concepts et des relations adéquates pour le processus d'inférence et de raisonnement pour exécuter une tâche spécifique.
- *Ontologies d'applications* : qui modélisent les connaissances utiles pour des applications spécifiques. Elles spécialisent généralement le vocabulaire des ontologies de domaine et des ontologies de tâches.

5. FORMALISMES DE REPRESENTATION

La représentation efficace des connaissances liées à un domaine spécifique admet la définition et le codage des entités de ce domaine dont l'objectif de les rendre compréhensible par une machine, et ceci, pour faciliter le processus de raisonnement [Lekhchine09]. Pour ce faire, plusieurs formalismes ont vu le jour par L'IA. Les plus connus sont : Les frames, Les graphes conceptuels et les logiques de description.

5.1 Frames : Ce formalisme a été proposé par M. Minsky [Minsky75]. La structure de données enregistrement de ce formalisme définie une situation et un objet. Pour ce formalisme, l'idée était de collecter toutes les informations essentielles qui correspondent à une situation et de les mettre sous forme d'une structure appelée *frame*. Certains auteurs, par exemple Hayes [Hayes79], ont constaté l'inexistence d'une sémantique formelle dans ce formalisme et ont exposé qu'une fois correctement formalisé, ce formalisme est une forme syntaxique de la logique de prédicat du premier ordre. GRUBER [Gruber95] était parmi les premiers ayant considéré ce formalisme comme langage de représentation d'ontologies. Le principe de ce modèle est de diviser les connaissances d'un domaine en classes (frames) qui représentent les concepts du domaine. À un frame est lié un ensemble d'attributs (slots), chaque attribut peut prendre ses valeurs parmi un ensemble de facettes.

5.2 Graphes Conceptuels : au début des années 80, J. SOWA [Sowa84] a proposé le modèle des Graphes Conceptuels (GC). Le GC est un modèle opérationnel de représentation de connaissances qui fait partie de la famille des réseaux sémantiques. Ce dernier est développé par M. Quillian [Quillian68] pour représenter la sémantique du langage naturel. Ce formalisme possède une représentation de graphe où les nœuds définissent des concepts et des individus. Ces nœuds sont reliés par des arcs étiquetés. Il y a deux sortes d'arcs : les arcs de propriété qui attribuent les propriétés à des concepts ou à des individus et les arcs IS-A qui représentent les relations hiérarchiques entre des concepts ou entre des individus. Coté mathématique, le modèle des GCs est basé sur la logique et la théorie des graphes [Sowa84]. Il est divisé en deux parties: une partie terminologique destinée au vocabulaire conceptuel des connaissances à représenter et une partie assertionnelle destinée à la représentation des assertions du domaine de connaissance en question.

5.3 Logique De Description (LD) : la LD a pour objectif de minimiser les temps de réponse. De ce fait, plusieurs travaux ont été publiés, de nombreuses recherches qui s'intéressent à l'étude du rapport expressivité/performance des différentes LDs [Nardi03]. Dans ce qui suit, nous allons voir, un peu d'historique sur les LDs, Les deux niveaux de description et le processus d'inférence.

5.3.1 Historique :

L'évolution et l'utilisation des LDs ont agi négativement sur les travaux concernant les autres formalismes : la logique des prédicats, les schémas (frames) [Minsky81] et les réseaux sémantiques, malgré l'existence de plusieurs correspondances entre les LDs et ces formalismes [Sattler03]. Au début des années 1980, et avec l'apparition des systèmes à base de connaissances tels que KL-ONE, BACK et LOOM [Baader03b, Nardi03], les premiers travaux sur les LDs ont vu le jour. Tandis que, les années 1990 était la période de naissance d'une nouvelle classe d'algorithmes, tel que : les algorithmes de vérification de satisfiabilité à base de tableaux, qui ont pour objectif le raisonnent sur des LDs dites *expressives* ou *très expressives*, mais en temps *exponentiel*. Néanmoins, en pratique, *le comportement* des algorithmes est souvent acceptable [Baader03b]. L'expressivité augmentée a permis l'apparition de nouvelles applications telles que le Web sémantique [Baader03b, Zou04, Horrocks03].

5.3.2 Les deux niveaux de description

Pour assurer une modélisation des connaissances d'un domaine en utilisant les LDs, il faut tenir compte les deux niveaux de cette logique. Le premier niveau est le niveau terminologique ou TBox qui a pour objectif la définition des connaissances générales d'un domaine, cependant, le deuxième niveau c'est le niveau assertionnel ou ABox représentant une instanciation propre à un domaine donné. Une TBox décrit des concepts et des rôles, alors qu'une ABox définit les individus en appelant et en indiquant en termes de concepts et de rôles, des assertions portant sur ces individus nommés [Nardi03]. (Voir la table 3.1)

Table3.1 : Une base de connaissances composée d'une TBox et d'une ABox [Lekhchine09]

TBox	ABox
Femelle ⊑⊤⊓¬Mâle Mâle ⊑⊤⊓¬Femelle Animal ≡ Mâle ⊔Femelle Humain ⊑Animal Femme ≡ Humain ⊓Femelle Homme ≡ Humain ⊓¬Femelle Mère ≡ Femme ⊓∃relationParentEnfant Père ≡ Homme ⊓∃relationParentEnfant MèreSansFille≡ Mère ⊓∀relationParentEnfant. ¬Femme relationParentEnfant ⊑T_R	Humain(Anne) Femelle(Anne) Femme(Sophie) Humain(Robert) ¬Femelle(Robert) Homme(David) relationParentEnfant(Sophie, Anne) relationParentEnfant(Robert, David)

5.3.2.1 Le niveau terminologique (TBox)

Une TBox est composée des concepts atomiques et rôles atomiques des entités d'un domaine. Cette TBox est caractérisée par une notation, tel que, Les noms commençants par une lettre majuscule désignent les concepts, tandis que, les rôles commencent par une lettre minuscule (par exemple : les concepts *UserContext*, *Profile*, et le rôle *relationHasProfile)*. [Nardi03].

5.3.2.2 Le niveau assertionnel (ABox)

Les assertions possibles sur les individus composent les éléments d'une ABox, qui doit être associée à une TBox, car les assertions se formulent en se référant aux concepts et rôles de la TBox. Une ABox définit des individus déterminés par des assertions d'individus nommés. Une assertion de rôle sous la forme *R(a, b)* montre que pour cette ABox, il existe un individu nommé *a* qui est en liaison avec un individu nommé *b* par le rôle *R* (défini dans la TBox associée). [Nardi03].

5.3.3 L'inférence

Le processus d'inférence ou de raisonnement est fait sur les deux niveaux précédents, terminologiques ou assertionnel (factuel) :

- L'inférence au niveau terminologique couvre quatre principaux problèmes [Baader03a] : la Subsomption, la satisfiabilité, la disjonction et l'équivalence.
- L'inférence au niveau assertionnel couvre aussi quatre principaux problèmes [Baader03a] : la Cohérence, La vérification de rôle, la vérification d'instance et le problème de récupération.

6. LES LANGAGES DE CONSTRUCTION DES ONTOLOGIES

Cette section va faire l'objet de la présentation des différents langages existants, utilisés pour la création des ontologies.

6.1 Extended Markup Language et XML Schema

Le XML (L'eXtended Markup Language) [W3C04d] est un langage de représentation et d'échange de documents structurés. Créé par SGML (Standard Generalized Markup Language) et décrit par le consortium Web, XML offre beaucoup d'avantages : (1) il permet de définir des structures arborescentes de documents en utilisant une représentation en balises qui permet de d'écrire les éléments composant la structure et les relations entre ces éléments.

(2) il n'exige aucune contrainte sémantique sur la définition de ces informations, ce qui implique que XML seul ne peut pas modéliser les ontologies. XML Schéma [W3C04e] (XML-S) est un outil destiné à la définition de grammaires qui caractérisent des arborescences de documents (notion de validité syntaxique). Avec les schémas XML, il est possible de vérifier la validité syntaxique de la structure arborescente d'un document, mais, il ne garantit pas la vérification de la sémantique de l'ensemble des informations comprises dans ce document.

6.2 Resource Description Framework Et RDF Schéma

RDF (Resource Description Framework) [W3C04b] est un modèle pour la définition de métas donnés. RDF représente les informations en utilisant un graphe orienté sous la forme d'un triplet : Sujet, Prédicat et Objet qui peut être aussi décrit en utilisant la syntaxe XML. Les figure 3.1 et figure 3.2 montrent un exemple [W3C04b] d'utilisation de RDF. RDF ne permet pas à l'utilisateur de décrire le vocabulaire des termes à utiliser, ni de définir la sémantique des objets utilisés, malgré, qu'il fournit une capacité d'échange de connaissances.

```
<rdf :RDF>
<rdf :Description about='Toto'>
<rdf :Property about='adresse'>12 rue
des pins </rdf :Property>
<rdf :Property about='age'> 37 </rdf :Property
</rdf:Description> </rdf:RDF>
```

Figure 3.1 : Représentation RDF/XML de Toto 37 ans qui habite au 12 rue des pins

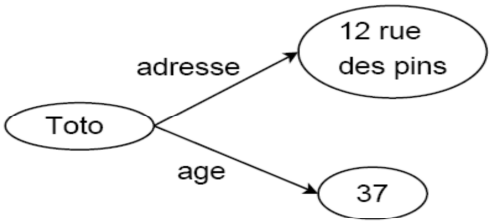


Figure3.2 : Représentation graphique de Toto 37 ans qui habite au 12 rue des pins

A son tour, RDF Schéma [W3C04c] ou RDFS est un langage qui permet de décrire des propriétés sémantiques pour les ressources utilisées dans un schéma. Ce dernier offre la possibilité de définir de nouvelles ressources sous la forme de plusieurs spécialisations de d'autres ressources.

6.3 OWL (Ontology Web Language)

OWL (Ontology Web Language [W3C04a, Zhou Mingtian03] est un langage basé sur la syntaxe RDF/XML. Il a bénéficié des travaux de DAML+OIL [Ian Horrocks01]. OWL représente une extension de RDF, en lui rajoutant l'aspect sémantique qui lui manque, tel que les moyens de comparaison de propriétés et de classes (identité, équivalence, contraire,

cardinalité, symétrie, etc.). De plus, grâce à ses primitives plus riches, il fournit une capacité d'interprétation et d'inférence plus grande à la machine que RDF et RDFS. OWL est composé de trois sous langages (voir figure 3.3) OWL Lite, OWL DL et OWL Full, qui offrent des capacités d'expression progressives alors ils sont destinés à des distinctes utilisations.

OWL Lite constitue le sous langage d'OWL le plus facile. Il est destiné aux utilisateurs qui ont, spécialement, besoin d'utiliser une hiérarchie de classifications avec des conditions simples. Comme exemple, malgré que le langage manage les contraintes de cardinalité, il n'accepte que les valeurs de cardinalité 0 ou 1, alors, il est plus simple de mettre en œuvre des outils pour OWL Lite que pour ses parents d'expression plus grande, car OWL Lite trace un chemin de migration rapide vers les thésaurus et autres taxonomies.

OWL DL est considéré plus complexe que OWL Lite, il est basé sur la logique de description (d'où son nom OWL Description Logics). Il assure l'accomplissement des inférences et leurs décidabilité, et ça malgré sa complexité. OWL DL englobe toutes les structures de langage OWL, qui ne sont disponibles cependant qu'avec certaines limitations (restrictions), par exemple, une classe peut être une sous-classe de plusieurs classes, mais elle ne peut pas être une instance d'une autre.

OWL Full représente le niveau le plus complexe d'OWL, mais aussi il assure le plus haut niveau d'expressivité. Son utilisation n'est contrainte que par le langage RDF, mais elle n'assure pas la décidabilité et la complétude des calculs reliés à l'ontologie. La syntaxe n'accepte pas de modification par rapport à OWL DL mais OWL Full admet une utilisation sans condition sur l'utilisation des constructeurs.

Les 3 niveaux d'**OWL** exposent une hiérarchie sur la validité des ontologies :

- une ontologie **OWL** Lite valide est aussi une ontologie OWL DL valide
- une ontologie **OWL DL** valide est aussi une ontologie OWL Full valide

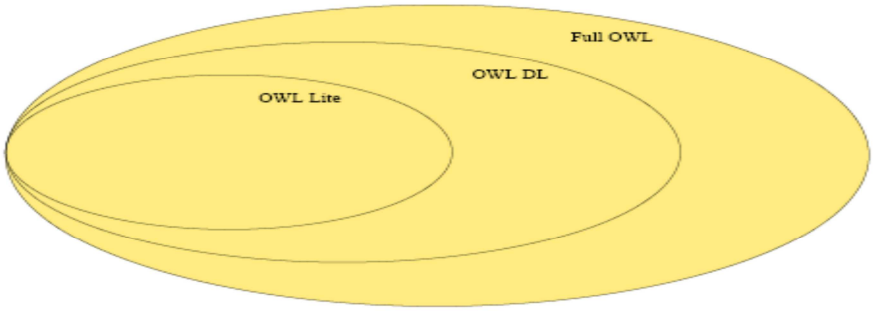


Figure 3.3 : Hiérarchies de langages OWL [Zhou Mingtian03]

En conclusion, OWL est un langage fondé sur XML, qui permet la définition riche des informations. Les logiques de description représentent la base théorique des langages d'ontologie comme OWL.

7. EDITEUR D'ONTOLOGIES

A nos jours, le nombre d'éditeurs d'ontologies ne cesse pas d'évoluer. Alors, il existe distincts éditeurs utilisant des formalismes diversifiés et permettant différentes propriétés. Dans cette partie, nous allons présenter quelques-uns, à savoir : OILed, OntoEdit, ONTOSAURUS, Protégé2000.

7.1 OILEd : OILEd [Bechhofer01] est un éditeur dont l'objectif est la création des ontologies dans le langage de représentation OIL. Cet éditeur est généralement observé comme une simple interface de la logique de description SHIQ. Il propose aussi les services d'un raisonneur. L'outil contient des techniques pour contrôler la cohérence des ontologies.

7.2 OntoEdit : OntoEdit [Sure02] est aussi un éditeur de construction d'ontologies autonome de tout formalisme, il utilise des mécanismes graphiques destinés à la visualisation d'ontologies. ONTOEDIT englobe un serveur consacré à l'édition d'une ontologie par plusieurs utilisateurs. La gestion des ordres d'édition garantit un contrôle efficace de la cohérence de l'ontologie.

7.3 ONTOSAURUS : ONTOSAURUS [Swartout97] est un éditeur constitué : d'un serveur qui utilise LOOM comme langage de définition des connaissances, et d'un serveur de navigation créant dynamiquement des pages HTML, affichant la hiérarchie de l'ontologie. Le serveur utilise des formulaires HTML dont l'objectif est d'autoriser l'usager à éditer l'ontologie. Des traducteurs du LOOM en Ontolingua, KIF, KRSS et C++, ont été déployés.

7.4 Protégé2000 : Protégé2000 [Noy01] est une interface modulaire, développée au Stanford Medical Informatics de l'Université de Stanford7. Il permet : l'édition, la visualisation, le contrôle (vérification des contraintes) d'ontologies, l'extraction d'ontologies à partir de sources textuelles et la fusion semi-automatique d'ontologies. Son modèle de connaissances est issu du modèle des frames et contient des classes (concepts), des slots (propriétés) et des facettes (valeurs des propriétés et contraintes), et aussi des instances des

classes et des propriétés. De nombreux plug-ins sont disponibles ou peuvent être ajoutés par l'utilisateur.

8. SYSTEMES DE RAISONNEMENT SUR LES ONTOLOGIES

L'objectif principal d'un système de raisonnement est l'inférence et la déduction logique des informations d'ontologie qui sont implicites, par conséquent, décrire les incohérences. Inférer implique tirer une conclusion d'une série de propositions reconnues pour ‘’vraies”. Un des principaux services fournis par un raisonneur est la vérification si une classe A est une sous-classe d'une autre classe B. De plus, l'exécution des tests spécifiques sur toutes les classes dans une ontologie, permet à un raisonneur d'acquérir la hiérarchie déduite des classes d'ontologie. Le rôle d'un moteur d'inférence est alors la compilation. Il existe plusieurs moteurs comme : Racer, Pellet, jena.

8.1 Racer : Le moteur d'inférence sans ambigüité, le plus connu et le plus utilisé dans le domaine est Racer [Haarslev01], grâce à ses performances et sa stabilité. Racer prend en considération les ontologies modélisées par son langage, mais il accepte aussi des ontologies définies en RDF ou OWL. Ceci à cause de la traduction de ces ontologies vers le langage utilisé par Racer. Ce moteur d'inférence possède aussi son propre langage de requête **nRQL** (new Racer proquery Language) pour interroger et consulter les ontologies. Les principaux services d'inférences de Racer: ***(1)*** Le test de *satisfiabilité* d'un concept (vérifier qu'un concept *C* accepte des instances), ***(2)*** Le test de *subsomption* de concepts (vérifier qu'un concept *A* est subsumé par un concept *B*), ***(3)*** Le test *d'instanciation* (vérifier qu'un individu *I* est instance d'un concept *C*, si seulement si $I \in C$). **Racer** possède quelques points négatifs :**(1)** Racer n'autorise pas l'utilisation de type de données défini par l'utilisateur, car il dispose de ses propres types de données et il exécute une conversion avec les types de base. **(2)** Racer est un produit commercial, il n'existe pas de version libre d'utilisation.

8.2 Pellet : Pellet [Evren-Sirin06] est le moteur le plus récent. Il est un des projets du MINDSWAP Group, « un groupe de recherche sur le Web sémantique de l'université du Maryland ». Pellet manipule des ontologies décrites en RDF ou OWL permettant des requêtes avec RDQL et SPARQL. *Les atouts de Pellet sont :***(1)** Pellet est open-source et réalisé en Java. **(2)** Pellet est un raisonneur OWL DL complet. **(3)** Pellet présente en cas d'incohérence dans l'ontologie des rétablissements possibles. *Les points négatifs de Pellet*

sont : **(1)** Il dispose d'une documentation pauvre par rapport à celle de Racer. **(2)** Actuellement Pellet ne permet pas l'utilisation de règles SWRL. (3) Pellet n'expose pas de système de souscription à un concept.

8.3 JENA : Développée par HP, **JENA** est une bibliothèque de classes Java qui simplifie le déploiement d'applications pour le web sémantique. Il Permet de gérer des ontologies (RDF-Schema, DAML+OIL, OWL) et de raisonner en utilisant les connaissances de l'ontologie. Il permet : (1) La création d'une classe : createClass retourne une OntClass (OntClass est une spécialisation de Resource). (2) La création d'une propriété : createObjectProperty retourne une ObjectProperty (ObjectProperty est une spécialisation de Resource). (3) L'utilisation de déclarations RDF. (4) Lecture et écriture RDF/XML, Notation 3. (5) Le stockage en mémoire ou sur disque de connaissances RDF. (5) La gestion d'ontologies : RDF-Schema, DAML+OIL, OWL.

8.4 Comparaison Entre Les Systèmes De Raisonnement

Il existe beaucoup de raisonneurs, spécifiques ou non à OWL. Dans le but de choisir un moteur d'inférence, nous avons dressé cette table qui englobe les caractéristiques de chaque moteur.

Table 3.2 : Caractéristiques des moteurs d'inférence

Jena	Racer	Pellet
1. API Java le plus largement utilisées pour RDF et OWL. **2.** Représentation du modèle. **3.** L'analyse syntaxique. **4.** Les requêtes et quelques outils de visualisation. **5.** Raisonner sur les instances et les concepts.	1. Raisonner sur les instances de concepts. 2. Permet la vérification de la hiérarchie entre concepts (subsomption de concepts).	**1.** Raisonner sur les instances de concepts. **2.** Le premier raisonneur OWL-DL sûr et complet. **3.** Pas de raisonnement sur les concepts. **4.** Ne peut pas être intégré dans JAVA

Selon la table 3.2, pour l'inférence de notre ontologie on ne peut pas utiliser Pellet, puisqu'il ne peut pas être intégré dans Java. Ce critère est primordial dans notre cas parce que notre ontologie va être exploitée par une application à base de Service Web. Par conséquent, notre choix a balancé entre **Racer** et **Jena**. Parce que **Jena** étant dédié beaucoup plus à la

manipulation des ontologies par les systèmes experts, nous avons opté pour le moteur d'inférence **RACER** pour ses avantages et ses caractéristiques.

9. LANGAGES D'INTERROGATION D'ONTOLOGIES

A travers cette partie, nous allons détailler les langages d'interrogation des ontologies à savoir : RDQL, SPARQL, nRQL et SWRL, qui sont fondés principalement sur la reconnaissance de graphe RDF. Les langages utilisables sur Pellet sont : RDQL et SPARQL tandis que, nRQL et SWRL sont utilisables sur Racer.

RDQL (RDF Data Query Language) [W3C04f] est un langage d'interrogation de données décrit en RDF. Il n'est pas standardisé, car il existe de nombreuses implémentations, malgré que la soumission W3C définit une base commune. Sa syntaxe est très proche de SQL :

SELECT variable [, variable]*
FROM documents rdf [, documents rdf]*
WHERE modèle de triplets
AND restrictions booléennes
USING définition des raccourcis

SPARQL (SPARQL Protocol And RDF QueryLanguage [W3C06]) est un rétablissement de RDQL. Il est en cours de standardisation au niveau du W3C. Ce langage définit une syntaxe presque semblable à RDQL, mais, en ajoutant notamment les opérateurs UNION et OPTIONAL dans la clause WHERE. L'opérateur UNION décrit la disjonction de triplets RDF. L'opérateur OPTIONAL décrit des triplets RDF optionnels pour le résultat de la requête.

nRQL (new Racer proquery Language [Haarslev01]) est le langage d'interrogation de Racer. Il est fondé sur la recherche de graphes RDF. Sa syntaxe est proche des deux autres langages, sauf pour sa notation préfixée des opérateurs.

SWRL (Semantic Web Rule language) [Horrocks04], est une proposition visant à combiner les ontologies et les règles, avec ontologies en OWL-DL et les règles sous RuleML, donc SWRL = OWL-DL + RuleML, puisque, OWL-DL est sans variable et RuleML utilise les variables. SWRL permet la manipulation d'instances par des variables (?x, ?y, ?z), SWRL ne permet pas de créer des concepts ni des relations, SWRL permet seulement

d'ajouter des relations suivant les valeurs des variables (individus) et la satisfaction de la règle. Les règles SWRL sont construites suivant ce schéma : ***antécédent -> conséquent***. Tel que : Antécédent = conjonctions d'atomes, et Conséquent = un seul atome, un atome étant : soit une instance de concept : Ci(z) = prédicat unaire , soit une relation OWL : Ri(x, y) = prédicat binaire, soit une des 2 relations SWRL : same-as(?x, ?y) ou different-from (?x, ?y), Exemple : *R1(x, y) different-from(x, y) C1(z) … → Rn(x, z)*. [Espinasse09]

De nombreux moteurs d'inférences commencent à supporter SWRL, par exemple : Bossam, Hoolet, KAON2, Pellet, RacerPro, R2ML (REWERSE Rule Markup Language) et Sesame. Ils suivent 3 types d'approches : (1) Traduire SWRL en logique du premier ordre (Hoolet), (2) Traduire OWL-DL en règles et appliquer un algorithme de chaînage avant (Bossam), (3) Intégrer les règles SWRL dans le moteur d'inférences OWLDL fondé sur les algorithmes des tableaux sémantiques (Pellet, Racer). Cependant, les implémentations actuelles de SWRL sont gourmandes en calcul, elles sont ainsi utilisables seulement pour des ontologies de taille petites ou moyennes [Espinasse09].

10. METHODES DE CONSTRUCTION D'ONTOLOGIE

La phase de création d'ontologies est un processus compliqué et il n'y a pas également des règles ou des moyens consensuels en IC (Ingénierie des Connaissances). Malgré cela, plusieurs travaux ont proposé des méthodes et des moyens pour construire des ontologies. Dans cette section, nous allons montrer les méthodes de construction des ontologies les plus utilisées, dans l'objectif de choisir une méthode que nous voulons utiliser dans nos travaux. Parmi ces méthodes on trouve : *ENTREPRISE Ontology* [Uschold95], *TOVE* [Gruninger95], *METHONTOLOGY* [Fernandez97] et *OTK* « OnToKnowledge » [Staab01].

10.1 ENTERPRISE : Uschold et King's [Uschold95] ont proposé un prototype d'une méthode fondée sur l'expérience de construction d'ontologies dans le domaine de la gestion des entreprises. De ce fait, la méthode ENTERPRISE s'appuie sur les quatre étapes suivantes: (1) la définition du rôle et la portée de l'ontologie. (2) l'identification des concepts et des relations essentielles et des définitions temporaires de ces éléments, codage de l'ontologie dans un langage adapté, l'intégration des ontologies existantes, dans cette étape, l'ontologie est effectivement bâtie. (3) l'évaluation de l'ontologie. (4) la rédaction d'une documentation et une trace des actions accomplies lors des distinctes étapes.

10.2 TOVE : La méthode de Grüninger et Fox [Grüninger95] est basée sur l'expérience du déploiement de l'ontologie du projet TOVE (TOrento Virtual Enterprise). Elle conduit à la construction d'un modèle logique de connaissances. Le développement de l'ontologie suit ces étapes: (1) La prise des scénarios de motivation. (2) La formulation des interrogations de capacités informelles. (3) La définition de la terminologie de l'ontologie. (4) L'évaluation de la finalisation de l'ontologie. La méthode TOVE reste spécifiée de façon abstraite, les distinctes étapes et les moyens ne sont pas exposés en détail.

10.3 METHONTOLOGY : *METHONTOLOGY* [Fernandez97] a été déployée au sein du laboratoire d'IA de l'université de Madrid. Son objectif principal est de construire des ontologies au niveau de connaissances. La motivation de ce projet est justifiée par l'inexistence de méthodes ou de guides structurés, ce qui, a conduit à une difficulté remarquable dans le processus de construction des ontologies consensuelles et partagées. De plus, une difficulté pour l'obtention d'une extension pour une ontologie existante ou de la réutiliser par d'autres ontologies. L'approche METHONTOLOGY contient les étapes suivantes [Fernandez97] :

- ***Spécification:*** cette étape a pour objectif d'entourer le champ de l'étude de l'ontologie ainsi que son domaine. Pour cela, il faut poser certaines questions sur : le domaine de l'ontologie, ses objectifs, son utilisation et sa maintenance.
- ***Conceptualisation:*** cette étape a pour but l'identification et la structuration des connaissances du domaine, par l'utilisation d'un ensemble de représentations intermédiaires semi-formelles (des tables et des graphes), simples et faciles à utiliser et à comprendre par les experts du domaine.

- ***Implémentation:*** dans cette étape, nous allons formaliser la conceptualisation acquise dans l'étape précédente en utilisant un formalisme de représentation d'ontologie, par la suite il faut coder l'ontologie dans un langage d'ontologie formel. La Figure 3.4 résume ces étapes :

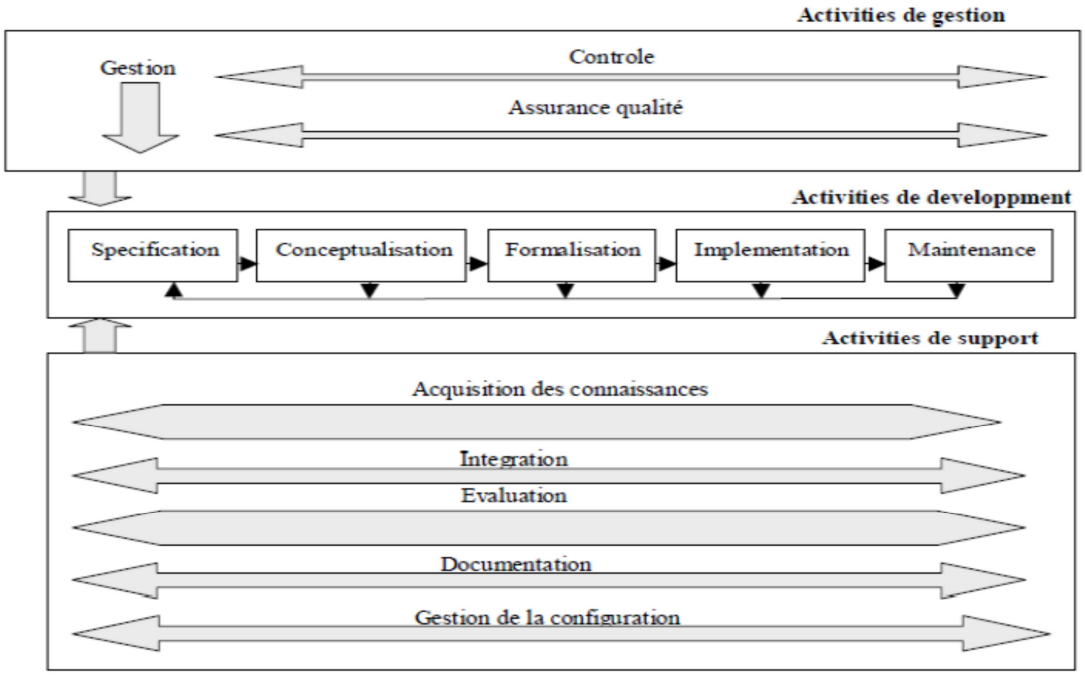


Figure 3.4 : Cycle de vie d'une ontologie dans Methontology [Fernandez97].

10.4 OTK : *OTK* (On-To-Knowledge) [Staab01] est un projet qui a proposé l'application des ontologies aux informations et ressources textuelles disponibles sur internet, extranet et en intranet. Le but était d'améliorer la qualité de la gestion de la connaissance dans les organisations distantes. Elle englobe cinq étapes : (1) l'étude de faisabilité, (2) la description des spécifications des besoins, (3) l'affinement, (4) l'évaluation, (5) et la maintenance.

En fin, dans le but de choisir une méthode de construction pour notre ontologie, nous nous sommes référés à une étude comparative faite dans [Keita07]. La table 3.3 donne un résumé de cette étude comparatif accomplie sur toutes les méthodologies précédentes.

Table3.3 : Comparaison des méthodes de développement des ontologies [Keita07].

Critères de comparaison	**TOVE**	**ENTERPRISE**	**METHONTOLOGY**	**OTK**
Spécification	++	+	++	++
Acquisition de connaissances	+	+	++	++
Conceptualisation	++	-	++	+
Formalisation	++	-	++	++
Evaluation	+	+	++	+
Outils support	Pas d'outils spécifiques	Pas d'outils spécifiques	ODE, WebODE, OntoEdit, Protégé-2000	OntoEdit avec ses plug-ins

D'après la table précédente, nous remarquons que METHONTOLOGY est l'approche qui satisfait tous les critères de comparaison. La plupart des approches ont donné une importance aux activités de développement et l'implémentation de l'ontologie, elles ont négligé d'autres activités importantes liées à la gestion, à l'évolution et à l'évaluation des ontologies. Par ailleurs, la plupart des approches sont très spécifiques et leurs utilisations sont réduites chacune, à un domaine unique.

Aussi, la plupart des approches ne disposent pas d'outil spécifique qui leur donne le support technique. De plus, aucun des outils disponibles ne couvre toutes les activités nécessaires dans la construction d'ontologie. Toutefois, METHONTOGOLY semble la plus efficace parmi toutes les méthodes proposées. Pour cela, nous avons opté, dans le cadre de construction de notre ontologie, pour l'utilisation de cette méthodologie.

CONCLUSION

Dans ce chapitre, nous avons fait un survol sur les ontologies en introduisant quelques définitions des ontologies suivies des constituants, classification, les méthodes de représentation et les raisons de créations des ontologies.

En plus, ce chapitre a présenté la notion de construction des ontologies, à travers la présentation des méthodes, outils et langage de construction des ontologies. Puis, il a introduit les systèmes de raisonnement et les langages d'interrogation des ontologies. Finalement, quelques applications des ontologies sont abordées.

Nous allons présenter dans le prochain chapitre notre contribution, à savoir, *« ContoLogy » : une ontologie pour la modélisation de contexte et la gestion des conflits*. Et l'exploitation de cette ontologie dans un processus d'adaptation des applications ubiquitaires.

CHAPITRE 4

« CONTOLOGY » : UNE ONTOLOGIE POUR LA MODELISATION DE CONTEXTE ET LA GESTION DES CONFLITS

INTRODUCTION

La modélisation du contexte est une étape très importante dans le processus de sensibilité au contexte. Elle permet aux applications de simplifier l'interaction avec le contexte en donnant une description abstraite des informations qui constituent ce dernier. La dite modélisation permet d'offrir les fondations pour une représentation expressive du contexte et une simplification de son utilisation.

Les approches de modélisation du contexte se différent selon le type des informations qu'elles permettent de définir et de représenter avec expressivité. Parmi ces approches, nous avons étudié les approches fondées sur les ontologies, qui sont les plus prometteuses et les plus expressives pour la modélisation du contexte dans un environnement sensible à ce dernier. Nous avons opté pour cette approche pour modéliser le contexte dans nos travaux.

Dans ce chapitre, nous allons détailler notre proposition de modélisation du contexte. Nous commencerons par énumérer les motivations pour les ontologies dans la section1. La section2 va faire l'objet de la représentation du modèle de contexte. Nous présentons dans la section3 le processus de construction de l'ontologie proposée qui modélise le contexte d'un utilisateur nomade et présente des solutions pour résoudre les conflits qui peuvent survenir entre les préférences de ce dernier. Dans la section4, nous allons détailler notre contribution pour l'exploitation de « ContoLogy » et l'adaptation des applications sensibles au contexte.

1. MOTIVATIONS POUR L'UTILISATION DES ONTOLOGIES

L'objectif principal de l'approche proposée est de modéliser le contexte de l'utilisateur par l'utilisation d'une représentation sémantique et de résoudre les conflits qui peuvent survenir lors de la vérification des préférences de l'utilisateur. Cette modélisation a pour but l'adaptation de la requête initiale de l'utilisateur à ce contexte, pour avoir une requête contextuelle et l'utiliser pour donner à l'utilisateur une réponse adaptée à son contexte

Nous avons opté, dans le cadre de ce travail, pour l'utilisation des ontologies pour le consensus et l'expressivité sémantique qu'elles procurent. Elles fournissent les moyens pour décrire les informations sémantiquement, partager des données décrites facilement pour être utilisées par d'autres applications et étendre la description initiale lorsque de nouveaux besoins apparaissent. Les langages d'ontologie peuvent créer des modèles expressifs, évolutifs, réutilisables et partageables sur lesquels on peut raisonner en utilisant des moteurs d'inférence. OWL [W3C04a] est un langage recommandé par W3C pour décrire des ontologies. Il fournit un moyen simple et efficace basé sur une description d'un modèle de XML pour : partager des données décrites, raisonner sur ces données et ajouter des axiomes pour décrire les relations spécifiques entre les informations. Finalement, les ontologies sont les plus expressives et les plus prometteuses pour la description du contexte dans un environnement sensible au contexte [Belhanafi06, Kosala13].

Pour les systèmes sensibles au contexte existants, des notations comme XML, XMbased CC/PP [Doulkeridis06], UML [Kapitsaki09], TopicMaps [Goslar04], RDF [Medjahed07] et OWL [Farrar10, Wang04a] sont utilisées pour la modélisation du contexte. L'utilisation d'OWL dans le présent travail a pour objectif la formalisation des relations de contexte en se basant sur une représentation de la logique de description *(description logicDL)*. Le choix du langage OWL est motivé par son support de raisonnement. Il fournit un langage logique d'inférence (OWL-DL) et un langage SWRL (Semantic Web RuleLanguage) pour permettre le raisonnement à base de règles [Kosala13]. Le langage logique (DL) supporte la composition du contexte et les améliorations des contraintes du contexte. OWL facilite le partage des conceptualisations. Afin d'encoder les aspects du contexte avec OWL-DL, une introduction des constructeurs de SHOIN (D) est nécessaire. Les constructeurs sont illustrés dans la Table4.1 [Farrar10]. Leur sémantique est basée sur les interprétations habituelles de la logique du premier ordre. « C » désigne le concept, et « R » désigne la relation. La spécification de DL peut être construite comme un ensemble d'axiomes. Les constructeurs de base de SHOIN (D) peuvent être utilisés avec la subsomption ou les

symboles d'équivalence ≡ pour créer des déclarations de DL. Les axiomes peuvent être des axiomes terminologiques (TBox) ou des axiomes Assertionnelle (ABox). Des axiomes terminologiques (des déclarations d'entités comme des concepts et des rôles, mais pas des individus) peuvent être des axiomes d'équivalence ou la sous-supposition. Les axiomes Assertionnels (appartiennent seulement aux individus) peuvent être des affirmations de concept ou des axiomes d'affirmation de rôle. Un axiome de subsomption donne des conditions nécessaires à un concept d'être inclus (sous-classe) dans un autre, par exemple, A ⊑ B où « A », « B » sont des concepts. Un axiome d'égalité est de la forme A≡ B. Un concept affirmation est de la forme « C (i) » où « C » un concept à partir d'un TBox et « i » est un individu. Le rôle affirmation est de la forme R(a, b), où « R » est un rôle d'une TBox et « a » et « b » sont des individus.

Table 4 .1 : La notation SHOIN (D) pour l'ontologie contexte [Kosala13]

constructeur	***SHOIN(D)***	**OWL-DL**
conjonction	C1 ∏ C2	intersectionOf(C1,C2)
Disjonction	C1 ∪ C2	unionOf(C1,C2)
Négation	¬C1	ComplementOf(C1)
La restrictionExists	∃R.C	SomeValuesFrom(C)on(R)
La Valeur derestriction	∀R.C	allValuesFrom(c)on(R)

2. LA REPRESENTATION DU MODELE DU CONTEXTE

Dans cette section, nous allons décrire comment nous avons défini les concepts de contexte. Pour le développement de notre ontologie de contexte "Contology", nous avons utilisé "METHONTOLOGY" [Fernandez97]. Selon [Fernandez97] Il est important de savoir que l'acquisition des connaissances est une activité indépendante dans le processus de développement de l'ontologie bien qu'elle puisse être confondue avec d'autres activités. La plupart des processus d'acquisition se font simultanément avec la phase de spécification des exigences, et diminuent lorsque le processus de développement de l'ontologie avance. Des experts, des livres, des manuels, des figures, des tableaux et même d'autres ontologies sont des sources d'acquisition à partir desquels le contexte peut être spécifié en utilisant des techniques de conjonction telles que: la réflexion, des interviews, l'analyse formelle et informelle des textes, et des outils d'acquisition de connaissances. Dans notre approche, la connaissance représente toutes les connaissances sur le contexte de l'utilisateur. Les techniques utilisées dans l'acquisition de contexte sont : - Les entretiens avec des experts, pour construire des spécifications des exigences. - L'analyse informelle du texte, l'étude des principaux concepts donnés dans les livres et les manuels. Cette étude permet de remplir l'ensemble des représentations intermédiaires de la conceptualisation. - L'analyse formelle du texte, la première chose à faire est d'identifier les structures à détecter (définition, affirmation, etc.) et le type des connaissances apportées par chacun (concepts, les attributs, les valeurs et les relations). - Les entretiens structurés avec des experts pour obtenir des connaissances précises et détaillées sur les concepts, leurs propriétés et leurs relations. - Toutes les définitions de contexte données par des chercheurs et des experts du domaine de sensibilité au contexte.

2.1 La Définition du Contexte

Les recherches dans le domaine de la sensibilité au contexte n'ont pas encore abouti à une définition générique et pragmatique du contexte. Plusieurs définitions du contexte ont été avancées [Zacarias05, Belotti04, Pittarello05, chaari06, Soukkarieh10, Kosala13]. Les définitions émises à ce jour sont très abstraites ou très spécifiques à un domaine particulier, faisant la formalisation du contexte très difficile. La définition du [Dey01b] est largement acceptée comme une «bonne» définition. Selon [Chaari06], cette définition ne permet pas la séparation entre les données contextuelles et les données de l'application, et le noyau de l'application doit être conçu de façon dépendante. Cette séparation selon [Chaari06] est très importante pour toute conception d'une application sensible au contexte. Une donnée définie

comme contextuelle dans un domaine peut être une donnée de l'application dans un autre domaine. Par exemple, la localisation GPS fait partie des données d'application dans un système de régulation du trafic, mais fait partie des données du contexte dans une application de télémédecine. La séparation entre les données contextuelles et les données de l'application est également importante dans la modélisation du contexte. [Chaari06] définit le contexte comme: «*l'ensemble des paramètres externes qui peuvent influencer le comportement de l'application en définissant de nouveaux points de vue sur les données et les services disponibles*". Par conséquent, dans la détermination des concepts les plus descriptifs de l'information qui constitue le contexte, nous avons choisi la séparation des données contextuelles des données de l'application selon la définition de [Chaari06] du contexte, car elle nous semble très pertinente et générique. Selon cette définition, nous pouvons diviser les concepts du contexte en deux parties : les concepts qui représentent le contexte d'utilisation d'un utilisateur et les concepts qui représentent le profil de l'utilisateur. Le contexte d'utilisation dans notre approche présente l'ensemble des données et permet d'indiquer la situation de l'utilisateur lorsqu'il se connecte à l'application ubiquitaire. Par exemple, il est représenté par les notions suivantes: L'utilisateur; la session; le dispositif mobiles utilisé (DM) et la localisation de l'utilisateur. Le profil de l'utilisateur est représenté par un ensemble de préférences de l'utilisateur. Nous allons détailler ces concepts dans les sections suivantes.

2.2 La Représentation Du Contexte: Préférences, Conflits

Parmi les concepts du contexte de l'utilisateur, nous trouvons les préférences. Dans cette partie, nous allons définir le concept de préférence de l'utilisateur et nous allons détailler une classification des différents types de préférences. Nous allons expliquer le concept de conflit et nous allons présenter ses causes et ses solutions.

2.2.1 Préférences

Par le concept de préférence de l'utilisateur, nous nous référons à un ensemble de descriptions couvrant ce que l'utilisateur aime recevoir comme services et le choix de l'affichage des résultats. Nous définissons deux types de préférences : Requested Service Preferences (les préférences des services demandées) et Display Preferences (les préférences d'affichage) et cinq conflits.

a) Requested Service Preferences

Elles décrivent comment l'utilisateur choisit ses services dans le système. Nous définissons ce type de préférence comme suit: Au cours de son premier contact avec notre système, l'utilisateur peut définir le contenu de chacun de ses services préférés. L'utilisateur

peut définir dès le début quand il demande le service «S» ce qui implique automatiquement les contenus: C1, C2 .... Etc.

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

Comme exemple pour illustrer notre proposition, nous considérons un utilisateur en voyage qui veut avoir la liste des restaurants dans son entourage. Il préfère que cette liste soit affichée sous forme de carte. Son profil utilisateur peut, par exemple, spécifier que lorsqu'il exécute le service "Consultation de la liste des restaurants", cet utilisateur est seulement intéressé par des restaurants offrant les plats qui respectent son régime, parce qu'il a des problèmes de santé. Alors, la préférence stipule que l'utilisateur veut exécuter le service "S" = "Consultation de la liste des restaurants ", dont le contenu est C = " Restaurants qui offrent la nourriture adéquate" La forme d'affichage est « image ». Donc, *Requested Service Preferences* est représentée comme suit :

Requested Service Preferences(S, {contenu}, {Services_Associés}).

S: est le service que l'utilisateur souhaite réaliser dans le système. ***{Contenu}***:est une liste des contenus définit par l'utilisateur dès son premier contact avec notre système. ***{*Services_Associés*}*** est une liste des services associés, que l'utilisateur veut exécuter s'il demande le service ***S.***

Comme exemple pour illustrer la représentation des ***Requested Service Preferences,*** nous considérons qu'un utilisateur souhaite demander un service ***S*** composé d'un ou plusieurs ***contenus*** et possède un ou plusieurs ***Services_ Associés***.

Par exemple, à chaque fois qu'un enseignant consulte «la liste réunions prévues», il souhaite savoir les réunions de la semaine en cours. Aussi, il demande le service_ associé «possibilité réunion» pour voir les possibilités de fixer une réunion entre des enseignants en précisant le jour, l'heure et la liste des enseignants concernés et le service_ associé «autres dates possibles» pour savoir toutes les dates possibles de réunion d'un ou de plusieurs enseignants donnés (jours et heures libres).

On peut représenter la ***Requested Service Preferences*** comme suit:

S1= Possibilité réunion (liste des enseignant concernés, jour, heure).

S2= autres dates possibles (jour libre, heures libres, liste des enseignants).

C1=réunions de la semaine en cours.

Requested Service Preferences (S: "la listeréunionsprévues", {S1, S2}, {C1})

b) Display Preferences

Display Preferences décrivent la manière dont l'utilisateur souhaite que son DM affiche l'information (par exemple, l'utilisateur désire seulement l'information au format Texte). A chaque Activité est associé un display preferences, elle est représentée comme suit:

Display Preferences (format, caractéristiques)

Où **format** peut prendre pour valeur : « vidéo », « texte », « image », « son» et chaque format repose sur un ensemble de caractéristiques.

La section suivante, détaille les conflits dans notre approche, présente leurs causes et explique leurs solutions.

2.2.2 Les Conflits

Par conflits, nous faisons référence aux problèmes qui peuvent survenir lors de la vérification des préférences de l'utilisateur. Par exemple, "Contradiction entre les display preferences et les caractéristiques du DM utilisé". Ce conflit peut survenir quand l'utilisateur demande un affichage non supporté par son DM utilisé. Pour ce genre de problèmes (conflits) -que nous allons définir par la suite-, notre proposition donne quelques solutions pour les résoudre. Pour chaque conflit est associée une solution, représenté comme suit:

Conflit (Type, Solution, Suggestion)

Type*:* représente le conflit qui peut survenir. ***Solution*** *:* permet de définir comment on peut réagir pour résoudre le conflit survenu. ***Suggestion*** *:* représente la proposition de l'utilisateur, si le système ne trouve pas une solution pour le conflit survenu.

Notre approche gère cinq conflits qui peuvent survenir lors de la vérification des préférences de l'utilisateur. Les deux tableaux suivants présentent notre proposition pour la résolution des conflits. Table 4.2 présente les conflits avec leurs causes. Table 4.3 présente les conflits et leurs solutions dans notre proposition.

Table 4.2: Conflits et Causes

N° Conflit	Conflit	Cause
1	***a. Contradiction entre les Requested_Service_Preferences ET les droits d'accés de l'utilisateur.***	• ce conflit peut survenir dans le cas où l'utilisateur demande un Service qui ne convient pas avec ses droits d'accès.
2	***b. Contradiction entre les Display preferences et les caractéristiques du DM utilisé***	• Ce conflit peut survenir quand l'utilisateur demande un affichage qui ne convient pas avec les types d'affichage de son DM utilisé. Par exemple : un utilisateur demande un affichage "Vidéo" sur un Sagem X1.
3	**c.** ***Différents souhaits d'Affichage pour le même service***	• Ce conflit peut survenir dans deux cas: **a)** L'utilisateur n'a pas spécifié les préférences d'affichage. **b)** Les préférences d'affichage ne conviennent pas avec les caractéristiques du DM. Dans les 2 cas, le système utilise l'ontologie de contexte pour résoudre les conflits survenus.
4	**d.** ***L'absence des Display preferences après la vérification de l'historique de l'utilisateur***	• l'utilisateur peut ne pas spécifier de préférences d'affichage, le système dans ce cas va revenir à l'historique de l'utilisateur, et il peut ne pas trouver de préférences d'affichage pour le service préféré.
5	**e.** ***Contradiction entre les Display preferences demandés et les possibilités d'affichage exprimés.***	• l'utilisateur peut demander le service sous un format non offert par le système, par exemple: l'utilisateur veut la liste des restaurants sous format carte, tandis que le système possède cette information sous format texte seulement.

Table 4.3 : Conflits et Solutions

Conflit	Solution
1	· le système doit revenir à l'utilisateur pour l'informer qu'il n'a pas le droit d'accéder à ces services et demande par conséquent, des suggestions pour ce problème. Si l'utilisateur ne donne pas de suggestions, le système arrête.
2	· le système exécute un des cas suivants: **a.** Utilise l'ontologie pour chercher et résonner sur une solution pour le conflit,en utilisant les informations des sessions précédentes, pour extraire Display Preferences qui convient avec les caractéristiques du DM utilisé. **b**. Sinon, il revient à l'utilisateur et demande des suggestions. **c**. Sinon dans les 2 cas précédents, il prend un Display Preferences par défaut, qui convient avec les caractéristiques du DM utilisé.
3	· Dans ce cas, nous proposons d'utiliser une *opération de calcul* qui nous donne le nombre de spécification de chaque préférence rencontrée. Ensuite, le système va effectuer une *opération de comparaison*, et va retenir la préférence qui a le nombre max de spécification par l'utilisateur. Dans le cas *d'égalité* entre les préférences, nous proposons d'utiliser une préférence par défaut qui convient avec les caractéristiques du DM utilisé.
4	· Dans ce cas, le système soit: · Il revient à l'utilisateur et demande ses suggestions. · Ou, il utilise une préférence par défaut.
5	· Dans ce cas, le système soit: **a**. Utilise l'ontologie pour chercher et résonner sur une solution pour le conflit, en

	utilisant les informations de sessions précédentes, pour extraire Display Preferences qui convient avec les caractéristiques du DM utilisé. **b**. Sinon, il revient à l'utilisateur et demande des suggestions. **c**. Sinon dans les 2 cas précédents, il prend un Display Preferences par défaut, qui convient avec les caractéristiques du DM utilisé.

Après avoir détaillé l'acquisition du contexte, la définition de ce que signifie contexte dans notre travail, et l'explication de la représentation du contexte. Dans la section suivante, nous allons présenter le processus de construction de l'ontologie basée sur la méthode "METHONDOLOGY".

3. PROCESSUS DE CONSTRUCTION DE L'ONTOLOGIE « ContoLogy », POUR LE DOMAINE DE LA SENSIBILITE AU CONTEXTE DANS LES ENVIRONNEMENTS UBIQUITAIRE.

Cette section présente les étapes suivies pour la construction de l'ontologie de contexte « ContoLogy » dédiée à la modélisation du contexte de l'utilisateur qui accède à une application ubiquitaire. Pour ce faire, nous utilisons un processus de construction dans le développement de l'ontologie partant des connaissances brutes et arrivant à une ontologie d'application opérationnelle représentée par le langage OWL. Les grandes étapes de ce processus sont inspirées de la méthodologie de construction d'ontologies «METHONTOLOGY» [Fernandez97] qui est le support de base pour la conceptualisation de l'ontologie à créer, à travers un ensemble de représentations intermédiaires semi-formelles. L'application de chacune des étapes de ce processus est basée sur l'exploitation du travail de HEMMAM [Hemam04]. Ce processus est composé de cinq étapes :

- ✓ Spécification des besoins.
- ✓ Conceptualisation.
- ✓ Formalisation.
- ✓ Implémentation.
- ✓ Test & évolution de l'ontologie.

La logique de description est le formalisme adopté pour l'expression de l'ontologie semi-formelle. OWL, le langage de définition d'ontologies, est choisi afin de codifier l'ontologie en utilisant l'éditeur d'ontologies Protégé OWL. Finalement, le système d'inférences RACER (*RenamedAbox and Concept Expression Reasoner*), est utilisé afin de tester la consistance de l'ontologie tout au long du processus de développement.

3.1 Spécification

Le développement de l'ontologie est débuté par la phase de spécification qui consiste à établir un document de spécification des besoins. Au sein de ce document, nous décrivons l'ontologie à construire à travers les cinq aspects suivants :

1. **Le domaine de connaissance :** l'ontologie, que nous venons de construire, s'inscrit dans le cadre de la modélisation du contexte pour l'adaptation des applications ubiquitaires sensibles au contexte. En effet, elle peut prendre ses concepts depuis les définitions du contexte dans le domaine de la sensibilité au contexte.
2. **L'objectif :** l'objectif majeur de notre ontologie est la représentation sémantique des concepts liés au contexte d'un utilisateur nomade qui accède à une application ubiquitaire. Cette représentation qui se base sur les ontologies permet l'inférence et le raisonnement sur le contexte explicite. Cela, va permettre à l'application ubiquitaire de percevoir le contexte de l'utilisateur de son environnement et par conséquent, adapter tout le comportement à ce dernier.
3. **Les utilisateurs :** cet aspect présente l'ensemble des utilisateurs pouvant exploiter l'ontologie. Dans notre cas, les utilisateurs de l'ontologie représentent la plateforme proposée pour l'adaptation et l'application ubiquitaire.
4. **Les sources d'informations :** les sources d'informations sur lesquelles nous nous sommes basés pour arriver à la construction de l'ontologie sont des définitions du contexte pour le domaine de la sensibilité au contexte « context-aware applications»
5. **La portée de l'ontologie :** Cet aspect consiste à déterminer à priori la liste des termes de l'ontologie (les plus importants), parmi ces termes, nous pouvons citer : ContextModel, ApplicationContext, ServicesApplication, DataBase, RulesApplication, ConflictContext, CauseConflict, SolutionConflict, ConflictSuggestion, LocationContext, PreferencesContext, UserPreferences, DevicePreferences, NetworkPreferences, ServicePreferences, ……..etc

Nous résumons cette phase dans un document RDF présenté dans la Figure 4.1. Il peut inclure aussi d'autres aspects tels que : la date de création de l'ontologie, ses créateurs, son niveau de formalité …etc.

```
<rdf :RDF>
<rdf : Description about=’URI Of Ontology’>
<Domaine> la modélisation du contexte pour l’adaptation des applications
ubiquitaires sensibles au contexte</Domaine>
<Date> 24/01/2014 </Date>
<Développé-par>
<rdf :Sequence>
<rdf :_1 SalimaBourougaa>
<rdf :_2 HassinaSeridi>
<rdf :_3 FaridMokhati>
</rdf:Sequence>
</Développé-par>
<Objectif>l’objectif majeur de notre ontologie est la représentation sémantique
des Concepts liées au contexte d’un utilisateur nomade qui accède à une
application ubiquitaire.</Objectif>
<Niveau de formalité> formel </Niveau de formalité>
<Termes>
<rdf :Sequence>
<rdf :_1 ContextModel><rdf :_2 , ConflictContext><rdf :_3 UserPreferences>…..
</rdf:Sequence>
</Termes>
<Sources>
<rdf :Sequence>…….
</rdf:Sequence>
</Sources>
</rdf : Description>
</rdf:RDF>
```

Figure 4.1 -Un document RDF de spécification de l’ontologie.

3.2 Conceptualisation :

Une fois la majorité des connaissances acquises, nous devons les organiser et les structurer en utilisant des représentations intermédiaires semi-formelles qui sont faciles à comprendre et sont indépendantes de tout langage d’implémentation. Cette phase comporte plusieurs étapes qui sont :

- Construction de glossaire de termes.
- Construction de diagramme de classification de concepts.
- Construction de diagramme de relations binaires.
- Dictionnaire de concepts.
- Tableaux des relations binaires.
- Tableaux des attributs.
- Tableaux des instances.
- Tableaux des règles : Les règles de la gestion des conflits

3.2.1 Construction de glossaire de termes :

Ce glossaire contient la définition de tous les termes relatifs au domaine (concepts, instances, attributs, relations) qui seront représentés dans l'ontologie finale, par exemple, dans notre cas les termes UserContext et ContextModel sont des concepts, PreferredBy et CoveredBy représentent des relations,…etc. La Table 4.4 fournit une liste détaillée des différents termes utilisés dans l'ontologie.

Table 4.4 – Glossaire de termes

Nom du terme	Synonymes	Description
ContextModel	Le modèle de contexte	Modélise tous les concepts du contexte liés à l'environnement ubiquitaire.
ApplicationContext	-	Représente l'application ubiquitaire
ServicesApplication	-	Représente les services offerts par l'application en question.
DataBase	-	Présente des informations sur la base de données de l'application.
RulesApplication	-	Regroupent toutes les règles liées au fonctionnement de l'application ubiquitaire, ainsi liées à l'utilisation de ses services.
ConflictContext	Les causes des conflits	Représente les conflits qui peuvent survenir entre les préférences de l'utilisateur
CauseConflict		Désigne les causes des conflits
SolutionConflict	Les solutions des conflits	Désigne les solutions proposées pour la résolution des conflits.
ConflictSuggestion	-	Représente les suggestions de l'utilisateur en cas de conflit non résolu.
LocationContext	-	Informations liées à la localisation de l'utilisateur.
PreferencesContext	-	Toutes les préférences liées au contexte de l'utilisateur.
UserPreferences	-	Informations sur les préférences de l'utilisateur
DevicePreferences	-	Informations liées aux préférences du dispositif mobile utilisé.
Network Preferences	-	Informations liées aux préférences du réseau.
ServicePreferences	-	Informations liées aux préférences des services proposés.
RequestedServicePreferences	-	Représente les services que l'utilisateur souhaite utilisé.
DisplayPreferences	-	Désigne la manière d'affichage des résultats préférés par l'utilisateur.
MobileDeviceContext	Le dispositif mobile utilisé.	Cette information définie le dispositif mobile utilisé par l'utilisateur durant une

		session.
MDCharacteristic	-	Informations sur les caractéristiques du DM.
MDServices	-	Informations sur les services offertes par le DM.
Profile	-	représente les caractéristiques statiques de l'utilisateur (nom, prénom, etc…) et ses préférences.
Interfaces	-	
Network	-	Expose des informations sur le type de réseau, ses caractéristiques, etc.
Sensor	-	-
LogicalSensor	-	-
PhysicalSensor	-	-
LocationCoordinates	-	Représente les coordonnées de la localisation de l'utilisateur.
RoleUser	-	Représente le rôle des utilisateurs dans l'application ubiquitaire.
UserContext	Le contexte de l'utilisateur	Les informations essentielles sur l'utilisateur qui utilise l'application ubiquitaire.
Rules	-	Regroupent les différentes règles liées aux: activités, réseaux, utilisateurs de l'application ubiquitaire.
ActivityUser	-	-
AccessRights	-	-
SessionContext	-	Présente la connexion de l'utilisateur au système (la durée de la connexion, la date de connexion, etc.)
IDContMod	-	Identifiant du modèle de contexte
Description	-	Description du modèle de contexte.
IDApp	-	Identifiant de l'application ubiquitaire
DescriptApp	-	Description de l'application ubiquitaire
IDServ	-	Identifiant du service.
DescripSer	-	Description du service.
IDConf	-	Identifiant du conflit.
DescripConfl	-	Description du conflit.
IDCause	-	Identifiant de la cause du conflit
DescripCause	-	Description de la cause du conflit.
IDSolution	-	Identifiant de la solution du conflit
DescripSolution	-	Description de la solution du conflit.
IDConfSugg	-	Identifiant de la suggestion du conflit
DescripConfSugg	-	Description de la suggestion du conflit
AltitudeLoc	-	L'altitude de la localisation
LatitudeLoc	-	Latitude de la localisation
CountryLoc	-	Pays de la localisation
CityLoc	-	Ville de la localisation
LongitudeLoc	-	Longitude de la localisation

IDPreferences	-	Identifiant des préférences
DescipDevPref	-	Description des préférences du DM
DescipNetPref	-	Description des préférences du réseau
DescipSerPref	-	Description des préférences des services
IDReqSerPref	-	Identifiant de la préférence du service demandé.
DescripSerPref	-	Description de la préférence du service demandé.
IDDispPref	-	Identifiant de la préférence d'affichage demandée.
DescripDispPref	-	Description de la préférence d'affichage demandée.
IDMD	-	Identifiant DM utilisé
TypeMD	-	Type du DM utilisé
StatusMD	-	Status DM utilisé
BatteryType	-	Type de batterie du DM utilisé
ConversationTime	-	Le temps de conversation
Weight	-	Poids du DM utilisé
display types	-	Les types d'affichage du DM utilisé
DescripMdSev	-	Description des services du DM utilisé
IDUser	-	Identifiant de l'utilisateur
PasswordUser	-	Mot de passe de l'utilisateur
PseudoUser	-	Pseudo de l'utilisateur
IDProfile	-	Identifiant du profil de l'utilisateur
NameUser	-	Le nom de l'utilisateur
first name	-	Le prénom de l'utilisateur
Address	-	Les adresses de l'utilisateur
birth date	-	La date de naissance de l'utilisateur
Email	-	Les e-mails de l'utilisateur
Name	-	Le rôle de l'utilisateur
DescripLocCoord	-	Description des coordonnées de la localisation
TypeSen	-	Le type du capteur
NameNet	-	Le nom du réseau
TypeNet	-	Le type du réseau
NameAct	-	L'activité de l'utilisateur
TypeAct	-	Le type de l'activité de l'utilisateur
DescriptionAccR	-	Description des droits d'accès de l'utilisateur
Start time	-	Le temps de début de la session
IsConceredBy	-	Les services de l'application sont concernés par la préférence de service demandé par l'utilisateur
HasSugg	-	Relie le conflit à sa suggestion
AttachedTo	-	Le conflit est attaché à une ou plusieurs préférences d'affichage
CausedBy	-	Un conflit est causé par une ou plusieurs

		préférences de service.
OccuredIn	-	Un conflit peut survenir dans une ou plusieurs sessions
ConcernConf	-	La suggestion pour un conflit concerne un et un seul conflit
IsSuggestedBy	-	Une suggestion pour un conflit est suggérée par un et un seul utilisateur
HasCoordinate	-	Une localisation a des coordonnés
PreferredBy	-	Les préférences de l'utilisateur concernent un utilisateur
CoveredBy	-	Les préférences de l'utilisateur sont couvertes par le profil de l'utilisateur
Distinguish	-	Les préférences de réseau distinguent un et un seul réseau.
Concern	-	La préférence de service concerne un ou plusieurs services
AttachedToDisp	-	Chaque préférence de service est attachée à une préférence d'affichage
Causes	-	Une préférence de service peut causer un conflit
RequestedBy	-	La préférence de service est demandée par un et un seul utilisateur.
Associated	-	La préférence de service est associée à une préférence de service
Occur	-	Une préférence d'affichage peut causer un conflit
PreferredByUser	-	La préférence d'affichage est demandée par un et un seul utilisateur
UsedBy	-	Un DM est utilisé par un utilisateur ou plusieurs.
Connect	-	Un DM est connecté à un ou plusieurs réseaux.
RelatedToSen	-	Un DM peut être lié à un capteur.
HasDevPref	-	Un DM a des préférences.
Characterize	-	Un profil caractérise un et un seul utilisateur
Includes	-	Un profil inclue les préférences de l'utilisateur.
AttachedToNet	-	Un réseau est attaché à des préférences
ExecutedBy	-	Une activité est exécutée par un et un seul utilisateur.
ConcernUser	-	Les droits d'accès concernent un utilisateur.
HasSolution	-	Chaque cause de conflit a un ou plusieurs solutions
ConcernCause	-	Une solution concerne une et une seule cause
LocatedIn	-	Un utilisateur peut avoir une ou plusieurs

		localisations
HasRole	-	Un utilisateur peut avoir un ou plusieurs rôles.
HasProfile	-	Un utilisateur peut avoir un ou plusieurs profils.
HasPreferences	-	Un utilisateur peut avoir une ou plusieurs préférences.
Use	-	Un utilisateur peut utiliser un ou plusieurs DM.
Requests	-	Un utilisateur peut demander un ou plusieurs services
Suggests	-	Un utilisateur peut suggérer un ou plusieurs suggestions pour des conflits.
Execute	-	Un utilisateur peut avoir une ou plusieurs préférences
HasAccess	-	Un utilisateur peut avoir un ou plusieurs droits d'accès.
ConnectedThrough	-	Un utilisateur peut connecter via une ou plusieurs session
HasConflict	-	Une session peut avoir un ou plusieurs conflits.

3.2.2 Construction Du Diagramme De Classification Des Concepts :

Dans cette étape, nous construisons le diagramme de classification des concepts. La hiérarchie de classification des concepts démontre l'organisation des concepts de l'ontologie en un ordre hiérarchique qui exprime les relations sous classe. Un concept universel « **Thing**», qui généralise tous les concepts racines des différentes hiérarchies de concepts est utilisé pour former une seule hiérarchie globale.

Pour construire la taxonomie des concepts, METHONTOLOGY propose d'utiliser les quatre relations : *Subclass-Of*, *Disjoint-Decomposition*, *Exhaustive-Decomposition*, et *Partition*.

- Un concept « C1 » est une *sous-classe* de concept « C2 » si et seulement si toute instance de « C1 » est une instance de « C2 ». Par exemple, CauseConflict est une sous-classe de ConflictContext.

- Une *Disjoint-Decomposition*d'un concept « C » est un ensemble de sous-classes de « C » qui ne couvrent pas « C » et n'ont pas des instances communes. Par exemple, Les conceptsDevicesPreferences et NetworkPreferences forment une *Disjoint-Decomposition*de concept PreferencesContext.

- Une *Exhaustive-Decomposition* d'un concept « C » est un ensemble de sous-classes de « C » qui couvrent « C » et peuvent avoir des instances communes.

- Une *Partition* d'un concept « C » est un ensemble de sous-classes de « C » qui couvrent « C » et n'ont aucune instance commune. Par exemple, Les concept CauseConflict et SolutionConflictformentune *Partition* de concept ConflictContext.

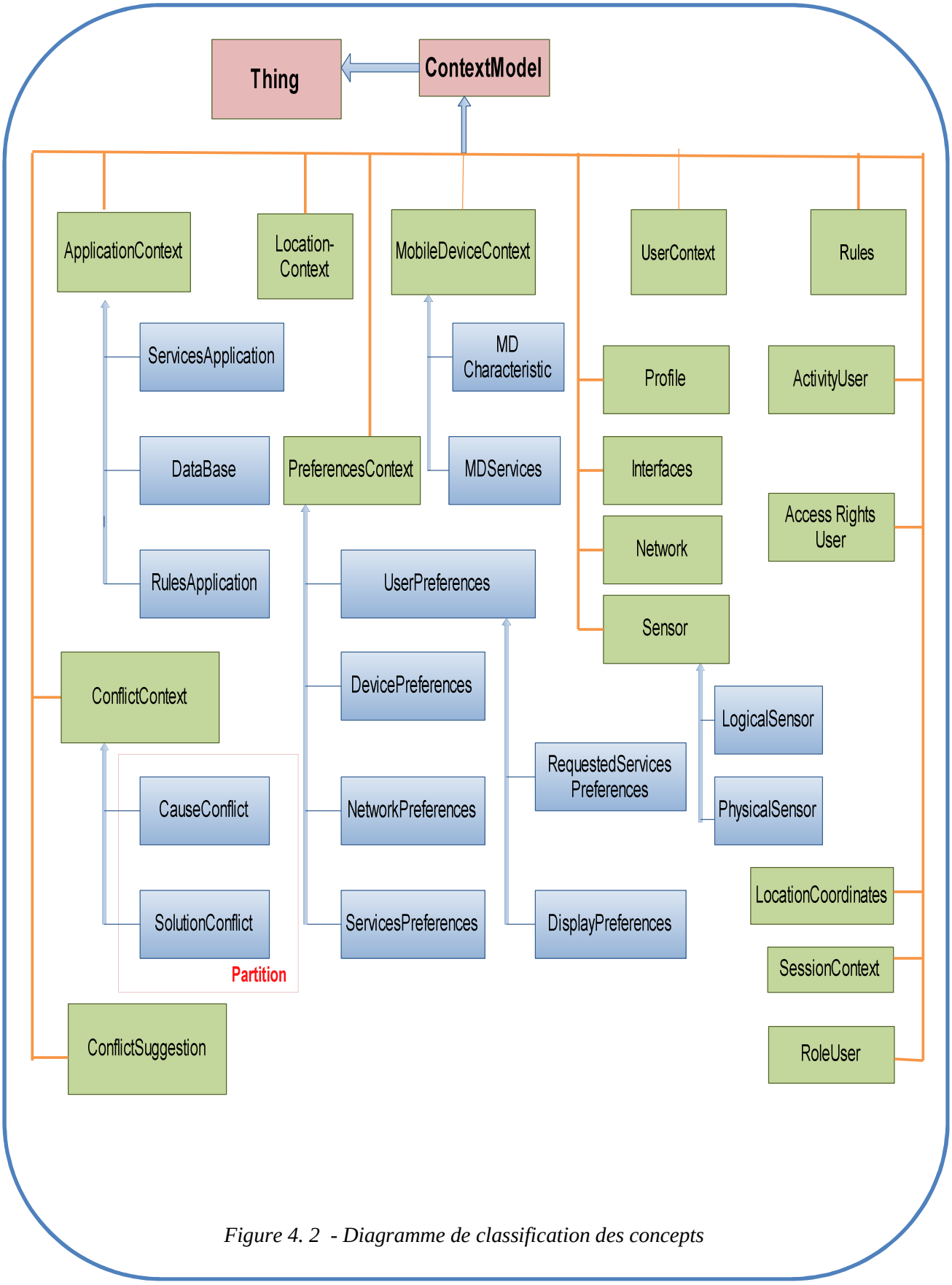


Figure 4. 2 - Diagramme de classification des concepts

3.2.3 Construction Du Diagramme Des Relations Binaires

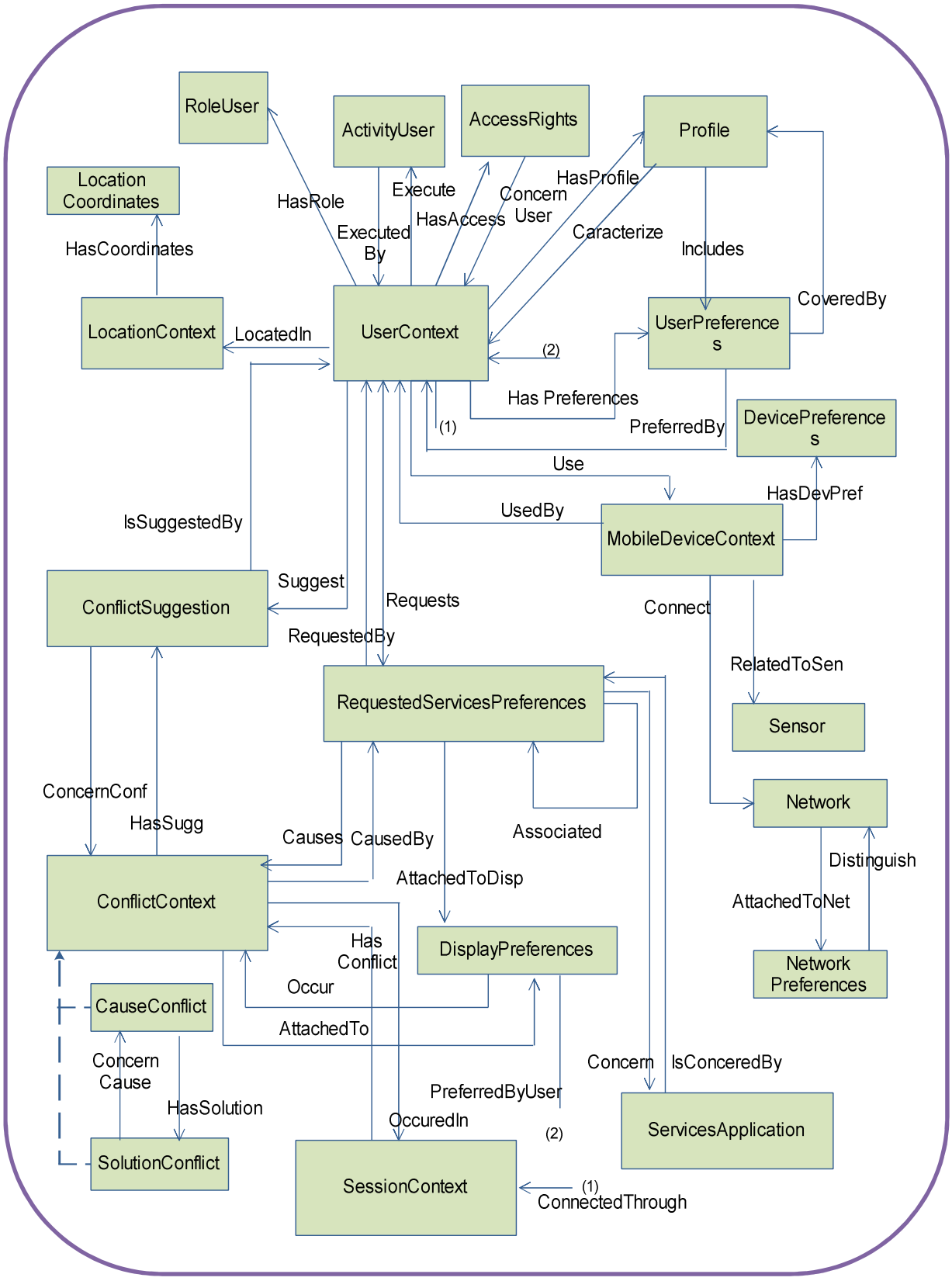


Figure 4.3 - Diagramme relations binaires

3.2.4 Dictionnaire De Concepts

Table 4.5 – Dictionnaire de concepts

Nom de concept	Instances	Attributs d'instance	Relations
ContextModel	-	IDContMod Description	-
ApplicationContext	-	IDApp DescriptApp	-
ServicesApplication	-	IDServ DescripSer	IsConceredBy
DataBase	-	-	-
RulesApplication	-	-	-
ConflictContext	Conflict1, conflict2 Conflict3, conflict4 Conflict5	IDConf DescripConfl	HasSugg AttachedTo CausedBy OccuredIn
CauseConflict	C1,C2,C3,C4,C5	IDCause DescripCause	HasSolution
SolutionConflict	S1,S2,S3,S4,S5	IDSolutionDescrip Solution	ConcernCause
ConflictSuggestion	-	IDConfSugg DescripConfSugg	ConcernConf IsSuggestedBy
LocationContext	-	AltitudeLoc LatitudeLoc CountryLoc CityLoc LongitudeLoc	HasCoordinate
PreferencesContext	-	-	-
UserPreferences	-	IDPreferences	PreferredBy CoveredBy
DevicePreferences	-	DescipDevPref	-
Network Preferences		DescipNetPref	Distinguish
ServicePreferences	-	DescipSerPref	/////
RequestedServicePreferences	-	IDReqSerPref DescripSerPref	Concern AttachedToDisp Causes RequestedBy Associated
DisplayPreferences	-	IDDispPref DescripDispPref	Occur PreferredByUser
MobileDeviceContext	-	IDMD TypeMD StatusMD	UsedBy Connect RelatedToSen HasDevPref
MDCharacteristic	-	BatteryType	-

		ConversationTime Weight display types	
MDServices	-	DescripMdSev	-
Profile	-	IDProfile NameUser first name Address birth date Email	Characterize Includes
Interfaces	-	-	//////
Network	-	NameNet TypeNet	AttachedToNet
Sensor	-	TypeSen	-
LogicalSensor	-	-	////
PhysicalSensor	-	-	/////
LocationCoordinates	-	DescripLocCoord	-
RoleUser	-	Name	
UserContext	-	IDUser PasswordUser PseudoUser	LocatedIn HasRole Execute HasAccess HasProfile HasPreferences Use Requests Suggests ConnectedThrough
Rules	-	-	-
ActivityUser	-	NameAct TypeAct	ExecutedBy
AccessRights	-	DescriptionAccR	ConcernUser
SessionContext	-	Start time	HasConflict

3.2.5 Tableaux Des Relations Binaires :

Table 4.6 – Tableau des relations binaires

Nom de relation	Concept source	Cardinalité source (max)	Concept cible	Relation inverse
IsConceredBy	ServicesApplication	N	RequestedServicePreferences	Concern
HasSugg	ConflictContext	N	ConflictSuggestion	ConcernConf
AttachedTo	ConflictContext	N	DisplayPreferences	Occur
CausedBy	ConflictContext	N	RequestedServicePreferences	Causes
OccuredIn	ConflictContext	N	SessionContext	HasConflict
ConcernConf	ConflictSuggestion	1	ConflictContext	HasSugg

IsSuggestedBy	ConflictSuggestion	N	UserContext	Suggest
HasCoordinate	LocationContext	1	LocationCoordinates	//
PreferredBy	UserPreferences	N	UserContext	HasPreferences
CoveredBy	UserPreferences	N	Profile	Includes
Distinguish	NetworkPreference	N	Network	AttachedToNet
Concern	RequestedService-Preferences	N	ServicesApplication	IsConceredBy
AttachedToDisp	RequestedService-Preferences	N	DisplayPreferences	//
Causes	RequestedService-Preferences	N	ConflictContext	CausedBy
RequestedBy	RequestedService-Preferences	N	UserContext	Requests
Associated	RequestedService-Preferences	N	RequestedServicePreferences	////
Occur	DisplayPreferences	N	ConflictContext	AttachedTo
PreferredByUser	DisplayPreferences	N	UserContext	/////
UsedBy	MobileDevice-Context	N	UserContext	Use
Connect	MobileDevice-Context	N	Network	//////
RelatedToSen	MobileDevice-Context	N	Sensor	///////////
HasDevPref	MobileDevice-Context	N	DevicePreferences	/////////
Characterize	Profile	N	UserContext	HasProfile
Includes	Profile	N	UserPreferences	CoveredBy
AttachedToNet	Network	N	NetworkPreference	Distinguish
ExecutedBy	ActivityUser	N	UserContext	Execute
ConcernUser	AccessRights	N	UserContext	HasAccess
HasSolution	Cause	5	Solution	ConcernCause
ConcernCause	Solution	5	Cause	HasSolution
LocatedIn	UserContext	N	LocationContext	///
HasRole	UserContext	N	RoleUser	///
HasProfile	UserContext	N	Profile	Characterize
HasPreferences	UserContext	N	UserPreferences	PreferredBy
Use	UserContext	N	MobileDeviceContext	UsedBy
Requests	UserContext	N	RequestedServicePreferences	RequestedBy
Suggests	UserContext	N	ConflictSuggestion	IsSuggestedBy
Execute	UserContext	N	ActivityUser	ExecutedBy
HasAccess	UserContext	N	AccessRights	UserContext
ConnectedThrough	UserContext	N	SessionContext	/////
HasConflict	SessionContext	N	ConflictContext	OccuredIn

3.2.6 Tableaux Des Attributs

Table 4.7 – Tableau des attributs

Nom d'attribut d'instance	Nom de concept	Type de valeur	Intervalle de valeur	Cardinalité
IDContMod	ContextModel	String	-	(1,1)
Description	ContextModel	String	-	(1,1)
IDApp	ApplicationContext	String		(1,1)
DescriptApp	ApplicationContext	String	-	(1,1)
IDServ	ServicesApplication	Integer	-	(1,1)
DescripSer	ServicesApplication	String	-	(1,1)
IDConf	ConflictContext	String	-	(1,1)
DescripConfl	ConflictContext	String		(1,1)
IDCause	CauseConflict	String	-	(1,1)
DescripCause	CauseConflict	String	-	(1,N)
IDSolution	SolutionConflict	string	-	(1,1)
DescripSolution	SolutionConflict	String	-	(1,N)
IDConfSugg	ConflictSuggestion	Integer	-	(1,1)
DescripConfSugg	ConflictSuggestion	String	-	(1,N)
AltitudeLoc	LocationContext	Real	>0	(1,1)
LatitudeLoc	LocationContext	Real	>0	(1,1)
CountryLoc	LocationContext	String	-	(1,1)
CityLoc	LocationContext	String	-	(1,1)
LongitudeLoc	LocationContext	String	-	(1,1)
IDPreferences	UserPreferences	String	-	(1,1)
DescipDevPref	DevicePreferences	List	-	(1,N)
DescipNetPref	Network Preferences	List	-	(1,N)
DescipSerPref	ServicePreferences	List	-	(1,N)
IDReqSerPref	RequestedServicePreferences	Integer	-	(1,1)
DescripSerPref	RequestedServicePreferences	String	-	(1,1)
IDDispPref	DisplayPreferences	Integer	-	(1,1)
DescripDispPref	DisplayPreferences	String	-	(1,1)
IDMD	MobileDeviceContext	String	-	(1,1)
TypeMD	MobileDeviceContext	String	-	(1,1)
StatusMD	MobileDeviceContext	String	-	(1,1)
BatteryType	MDCharacteristic	String	-	(1,1)
ConversationTime	MDCharacteristic	Real	-	(1,1)
Weight	MDCharacteristic	Real	-	(1,1)
display types	MDCharacteristic	List	-	(1,N)
DescripMdSev	MDServices	List	-	(1,N)
IDUser	UserContext	Integer	-	(1,1)
PasswordUser	UserContext	String	-	(1,1)
PseudoUser	UserContext	String	-	(1,1)
IDProfile	Profile	Integer	>0	(1,1)

NameUser	Profile	String	-	(1,1)
first name	Profile	List	-	(1,N)
Address	Profile	List	-	(1,N)
birth date	Profile	Date	-	(1,1)
Email	Profile	String	-	(1,N)
Name	RoleUser	String	-	(1,1)
DescripLocCoord	LocationCoordinates	String	-	(1,1)
TypeSen	Sensor	String	-	(1,1)
NameNet	Network	String	-	(1,1)
TypeNet	Network	String	-	(1,1)
NameAct	ActivityUser	String	-	(1,1)
TypeAct	ActivityUser	String	-	(1,1)
DescriptionAccR	AccessRights	List	-	(1,N)
Start time	SessionContext	Time	-	(1,1)

3.2.7 Tableaux Des Instances :

Table 4.8 – Tableau des instances

Nom de l'instance	Nom du concept	Attributs	Valeurs
Conflict1	ConflictContext	IDConf DescripConfl	"Contradiction between the RequestedServicePreferences and access rights of the user "
Conflict2	ConflictContext	IDConf DescripConfl	"Contradiction between the display preferences and the characteristics of used MD"
Conflict3	ConflictContext	IDConf DescripConfl	"Various wishes of Display for the same Requested Service"
Conflict4	ConflictContext	IDConf DescripConfl	"Absence of display preferences after checking the historic of the user"
Conflict5	ConflictContext	IDConf DescripConfl	"Contradiction between the Display preferences requested and display capabilities expressed"
C1	CauseConflict	IDCause DescripCause	"The user requests a Service which does not suit with these access rights".
C2	CauseConflict	IDCause DescripCause	"The user requests a display which is not supported by his used MD"
C3	CauseConflict	IDCause DescripCause	"The user did not specify Display preferences. OR Display preferences are not suitable to the characteristics of MD"
C4	CauseConflict	IDCause DescripCause	"The user does not specified display preferences"
C5	CauseConflict	IDCause DescripCause	"The user requests the Service in a format not offered by the system"
S1	SolutionConflict	IDSolutionDescripSolution	"The system returns to the user to inform him that he has not the right to access these activities and asks consequently, suggestions for this problem. If the user does not give suggestions, the system stops"
S2	SolutionConflict	IDSolutionDescripSolution	"The system execute one of the following cases: a. Returns to the historic of the user in the system to extract the display preferences that agrees with the characteristics of the used MD. In this case: it

			can meet the third conflict, and it will execute the solution. b. Returns to the user and demand suggestions. c. In the case where the system does not find a solution in the historic of the user, or suggestions he can take a preference of display by default which suits of course with the characteristics of the used MD, and it to satisfy the user"
S3	SolutionConflict	IDSolutionDescripSolution	"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."
S4	SolutionConflict	IDSolutionDescripSolution	"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."
S5	SolutionConflict	IDSolutionDescripSolution	"In this case the system executes one of the following cases: a. it returns to the historic of the user to extract another display preference for the activity requested which suits with the characteristics of the MD. b. it returns to the user and asks these suggestions, c. It takes a preference by default"

3.2.8 Description Des Règles Du Contexte : Les Règles De La Gestion Des Conflits :

En utilisant l'ontologie "ContoLogy", nous pouvons dériver un nouveau contexte. Le contexte dérivé est un contexte implicite dérivé d'un contexte explicite. Dans notre ontologie de contexte, la dérivation est basée sur des règles de la forme **antécédent → conséquent**. Antécédentet et conséquent sont composés d'un ou plusieurs concepts du contexte et la description des relations. Le contexte dérivé peut affecter d'autres aspects contextuels, par exemple: ConflictContext est un contexte dérivé de MobileDeviceContext, UserContext et UserPreferences. Dans notre travail, nous avons prévu de résoudre tous les conflits qui peuvent survenir lors de la vérification des préférences de l'utilisateur. Dans les sections précédentes, nous avons défini cinq conflits qui peuvent survenir lors de la vérification des préférences de l'utilisateur. Pour gérer ces conflits, nous avons utilisé SWRL (Semantic Web RuleLanguage), nous avons défini cinq règles SWRL pour dériver les conflits, cinq SWRL pour résoudre ces conflits et nous avons créé ces règles sous Protége 2000 [Noy00].

a) SWRL Pour Dériver Les Conflits

On a défini cinq règles pour dériver les cinq conflits (voir table 4.2)

- ✓ ***Régle1****: dérive le conflit 1 : "****Contradiction entre les Requested_Service_Preferences et les droits d'accès de l'utilisateur.****" :*

UserContext(?x) ∧ RequestedServicesPreferences(?A) ∧ AcessRightsUser(?AR) ∧differentFrom(?A, ?AR) ∧ConflictContext(?c) → Causes(?A, ?c)

- ✓ ***Rule2****: dérive le conflit 2: "****Contradiction entre les Display preferences et les caractéristiques du DM utilisé*** *":*

UserContext(?x) ∧DisplayPreferences(?d) ∧MobileDeviceContext(?dm) ∧differentFrom(?dm, ?d) ∧ConflictContext(?c) → Occur(?d, ?c)

- ✓ ***Rule3****: dérivele conflit 3: "****Différents souhaits d'affichage pour le même service****":*

UserContext(?x) ∧RequestedServicesPreferences(?A) ∧MobileDeviceContext(?dm) ∧differentFrom(?dm, ?d) ∧sqwrl:isEmpty(?d) ∧ConflictContext(?c) → Causes(?A, ?c)

- ✓ ***Rule4****: dérive le conflit 4: "****L'absence des display preferences après la vérification de l'historique de l'utilisateur****":*

UserContext(?x) ∧RequestedServicesPreferences(?A) ∧sqwrl:isEmpty(?d)∧Notprefered(?d, ?x) ∧ConflictContext(?c) → Causes(?A, ?c)

- ✓ ***Rule5****: dérive le conflit 5: "****Contradiction entre les Display preferences demandés et les possibilités d'affichage exprimés****":*

UserContext(?x) ∧RequestedServicesPreferences(?A) ∧MobileDeviceContext(?dm) ∧differentFrom(?dm, ?d) ∧sqwrl:isEmpty(?d) ∧ConflictContext(?c) →Causes(?A, ?c)

b) SWRL pour résoudre les conflits

Nous avons défini cinq SWRL pour résoudre les cinq conflits, voir table 4.3 pour la description des valeurs de tous les paramètres des règles suivantes :

- ✓ ***Rule6****: résout le conflit1: ConflictContext(ConflictContext_1) ∧CauseConflict(CauseConflict_1) → HasSolution(ConflictContext_1, SolutionConflict_1)∧HasSolution(ConflictContext_1, SolutionConflict_2).*

- ✓ ***Rule7**: résout le conflit2: ConflictContext(ConflictContext_2) ∧CauseConflict(CauseConflict_2) → HasSolution(ConflictContext_2, SolutionConflict_3)∧HasSolution(ConflictContext_2, SolutionConflict_1)∧HasSolution(ConflictContext_2, SolutionConflict_4)*
- ✓ ***Rule8**: résout le conflit 3: ConflictContext(ConflictContext_3) ∧CauseConflict(CauseConflict_3) → HasSolution(ConflictContext_3, SolutionConflict_5) ∧HasSolution(ConflictContext_3, SolutionConflict_4)*
- ✓ ***Rule9**: résout le conflit4: ConflictContext(ConflictContext_4) ∧CauseConflict(CauseConflict_4) → HasSolution(ConflictContext_4, SolutionConflict_1) ∧HasSolution(ConflictContext_4, SolutionConflict_4)*
- ✓ ***Rule10**: résout le conflit5: ConflictContext(ConflictContext_5) ∧CauseConflict(CauseConflict_5) → HasSolution(ConflictContext_5, SolutionConflict_1)∧HasSolution(ConflictContext_5, SolutionConflict_3)∧HasSolution(ConflictContext_5, SolutionConflict_4)*

3.3 Formalisation

Dans cette étape, nous allons utiliser le formalisme de la logique de description afin de formaliser le modèle conceptuel que nous avons obtenu dans l'étape précédente de conceptualisation. Nous Définissons le ContextModel comme suit **:**

ContextModel =(T ,A)

Avec T= (Tbox) et A=(Abox)

3.3.1 Construction De Tbox :

Nous construisons la TBox en définissant les concepts et les rôles et en utilisant les constructeurs fournis par les logiques de descriptions. Par exemple, la définition « un 'ActivityUser' doit être au moins exécuté par un 'user' » peut être écrite en logique de description par: $ActivityUser \equiv \exists ExecutedBy$

De plus, nous construisons la TBox par la spécification des relations de **subsomption** qui existent entre les différents concepts/rôles ; par exemple pour spécifier que la classe 'User Context' est subsumée par la classe 'ContextModel' on écrit :

$$UserContext \sqsubseteq ContextModel$$

Les définitions de différents concepts sont illustrées dans le tableau ci-dessous.

Table 4.9 – définition de la TBox

Concept	Définition	Relations de subsomption
ContextModel	≡ (UserContext ⊔MobileDeviseContext⊔ LocationContext ⊔ ApplicationContext ⊔ ConflictContext ⊔ ConflictSuggestion ⊔ PreferencesContext ⊔ Profile ⊔ Intrefaces ⊔ Network ⊔ Sensor ⊔ Rules ⊔ ActivityUser ⊔ AcessRights ⊔ Location Coordinates ⊔ RoleUser)	ContextModel ⊑ ⊤
ApplicationContext	≡ (ServicesApplication ⊔ DataBase ⊔ RulesApplication)	ApplicationContext ⊑ ContextModel
ServicesApplication	≡∃IsConceredBy.RequestedServicesPreferences	ServicesApplication ⊑ ApplicationContext
DataBase	-	DataBase ⊑ ApplicationContext
RulesApplication	-	Rules Application ⊑ Application- Context
ConflictContext	≡ (CauseConflict ⊔ SolutionConflict) ⊓ ∃HasSugg.ConflictSuggestion ⊓ ∃CausedBy.RequestedServicesPreferences ⊓ ∃ AttachedTo.DisplayPreferences ⊓ ∃OccoredIn.SesionContext	ConflictContext ⊑ ContextModel
CauseConflict	≡ (ConflictContext ⊔¬SolutionConflict)⊓ ∃HasSolution.SolutionConflict	CauseConflict ⊑ ConflictContext
SolutionConflict	≡ (ConflictContext ⊔¬CauseConflict)⊓∃ConcernCause.CauseConflict	Solution Conflict ⊑ Conflict Context
ConflictSuggestion	≡ ∃ConcernConf.ConflictContext⊓ ∃IsSuggestedBy.UserContext	ConflictSuggestion ⊑ ContextModel
LocationContext	≡ ∃HasCoordinate.LocationCoordinates	LocationContext ⊑ ContextModel
PreferencesContext	≡(UserPreferences⊔DevicePreferencesNetworkPreferences⊔ServicesPreferences)	PreferencesContext ⊑ ContextModel
UserPreferences	≡(RequestedServicesPreferences ⊔ DisplayPreferences) ⊓∃CoveredBy.profile⊓∃Preferred.UserContext	UserPreferences ⊑ PreferencesContext
DevicePreferences	-	DevicePreferences ⊑ PreferencesContext
NetworkPreferences	≡ ∃Distinguish.Network	NetworkPreferences ⊑ PreferencesContext
ServicePreferences	-	ServicePreferences ⊑ PreferencesContext
RequestedService Preferences	≡∃AttachedToDisp.DisplayPreferences⊓ ∃Causes.ConflictContext⊓ ∃RequestedBy.UserContext⊓ ∃Concern.ServicesApplication ⊓ ∃Associated.RequetedServicePreferences	RequestedServicePreferences ⊑ UserPreferences
DisplayPreferences	≡ ∃Occur.ConflictContext⊓	DisplayPreferences ⊑

	∃PreferredByUser.UserContext	UserPreferences
MobileDeviceContext	≡(MDCharacteristic ⊔ MD Services) ⊓∃UsedBy.UserContext⊓∃Connect.Network⊓∃Has.DevicesPrefences⊓∃RelatedTo.Sensor	MobileDeviceContext ⊑ ContextModel
MDCharacteristic	-	MD Characteristic ⊑ MobileDeviceContext
MDServices	-	MD Services ⊑MobileDeviceContext
Profile	≡ ∃Caracterize.UserContext⊓ ∃Includes.UserPreferences	Profile ⊑ ContextModel
Interfaces	-	Interfaces ⊑ContextModel
Network	≡AttachedToNet.NetworkPreferences	Network ⊑ ContextModel
Sensor	-	Sensor ⊑ ContextModel
LogicalSensor	-	LogicalSensor ⊑ Sensor
PhysicalSensor	-	PhysicalSensor ⊑ Sensor
LocationCoordinates	-	LocationCoordinates⊑ ContextModel
RoleUser	-	RoleUser⊑ ContextModel
UserContext	≡ ∃Execute.Activity⊓ ∃HasProfile.profile⊓ ∃HasAccess.AccessRights⊓ ∃HasRole.Role⊓ ∃LocatedIn.LocationContext⊓ ∃Suggest.ConflictSuggestion⊓ ∃Use.MobileDevice⊓ ∃HasPreferences.UserPreferences⊓ ∃ConnectedThrough.SessionContext⊓ ∃Request.RequestedServicePreferences	UserContext⊑ ContextModel
Rules	-	Rules ⊑ ContextModel
ActivityUser	≡ ∃ExecutedBy.UserContext	ActivityUser⊑ ContextModel
AccessRights	≡ ∃Concern.UserContext	AccessRights⊑ ContextModel
SessionContext	≡ ∃HasConflict.ConflictContext	SessionContext⊑ ContextModel

3.3.2 Construction De Abox :

Nous décrivons les faits en utilisant le langage assertionnel, de la manière suivante :

A(C) : Pour spécifier que « A » est une instance de la classe « C » ; Par exemple : C1(CauseConflict) / C1= cause1

R(A1, A2) : Pour spécifier que les deux individus « A1 » et « A2 » sont reliés par la relation « R ». Par exemple : HasSolution(C1, S1). / C1= cause1, S1= Solution1

Dans les deux tableaux Table 4.10, Table 4.11, nous définissons quelques assertions :

Table 4.10 – Partie assertionnelle des concepts

Concept	Définition
ConflictContext	Conflict1(ConflictContext) Conflict2(ConflictContext) Conflict3(ConflictContext) Conflict4(ConflictContext) Conflict5(ConflictContext)
CauseConflict	C1(CauseConflict) C2(CauseConflict) C3(CauseConflict) C4(CauseConflict) C5(CauseConflict)
SolutionConflict	S1(SolutionConflict) S2(SolutionConflict) S3(SolutionConflict) S4(SolutionConflict) S5(SolutionConflict)

Table 4.11– Partie assertionnelle des relations

Relation	Définition
HasSolution	HasSolution(C1,S1)
HasSolution	HasSolution(C2,S2)
HasSolution	HasSolution(C3,S3)
HasSolution	HasSolution(C4,S4)
HasSolution	HasSolution(C5,S5)

3.4 Implémentation

Après la conception, nous allons implémenter notre ontologie. Notre choix porte sur OWL qui représente un langage de codification utilisé pour implémenter l'ontologie en OWL, et cela pour toutes les fonctionnalités sémantiques que permet OWL et qui sont plus riches que celles des langages RDFS & DAML+OIL.

3.4.1 Présentation De L'éditeur PROTEGE 2000 :

PROTEGE 2000 est une interface modulaire, développée au Stanford Medical Informatics de l'Université de Stanford7, permettant l'édition, la visualisation, le contrôle, l'extraction à partir de sources textuelles, et la fusion semi-automatique d'ontologies [Noy00]. PROTEGE 2000 autorise la définition de méta-classes, dont les instances sont des classes, ce qui permet de créer son propre modèle de connaissances avant de bâtir une ontologie. De nombreux plugins sont disponibles ou peuvent être ajoutés par l'utilisateur. L'interface, très bien

conçue, et l'architecture logicielle permettant l'insertion de plug-ins pouvant apporter de nouvelles fonctionnalités (par exemple, la possibilité d'importer et d'exporter les ontologies construites dans divers langages opérationnels de représentation tels que OWL ou encore la spécification d'axiomes) ont participé au succès de PROTEGE 2000, qui regroupe une communauté d'utilisateurs très importantes et constitue une référence pour beaucoup d'autres outils [Troncy04].

3.4.2 Définition De La Hiérarchie Des Classes :

Nous commencerons tout d'abord par la création des concepts spécifiés dans l'étape de conceptualisation. PROTEGE 2000 nous offre également un moyen de construire la hiérarchie de concepts, la Figure 4.4 présente les concepts de « ContoLogy », crées sous PROTEGE 2000.

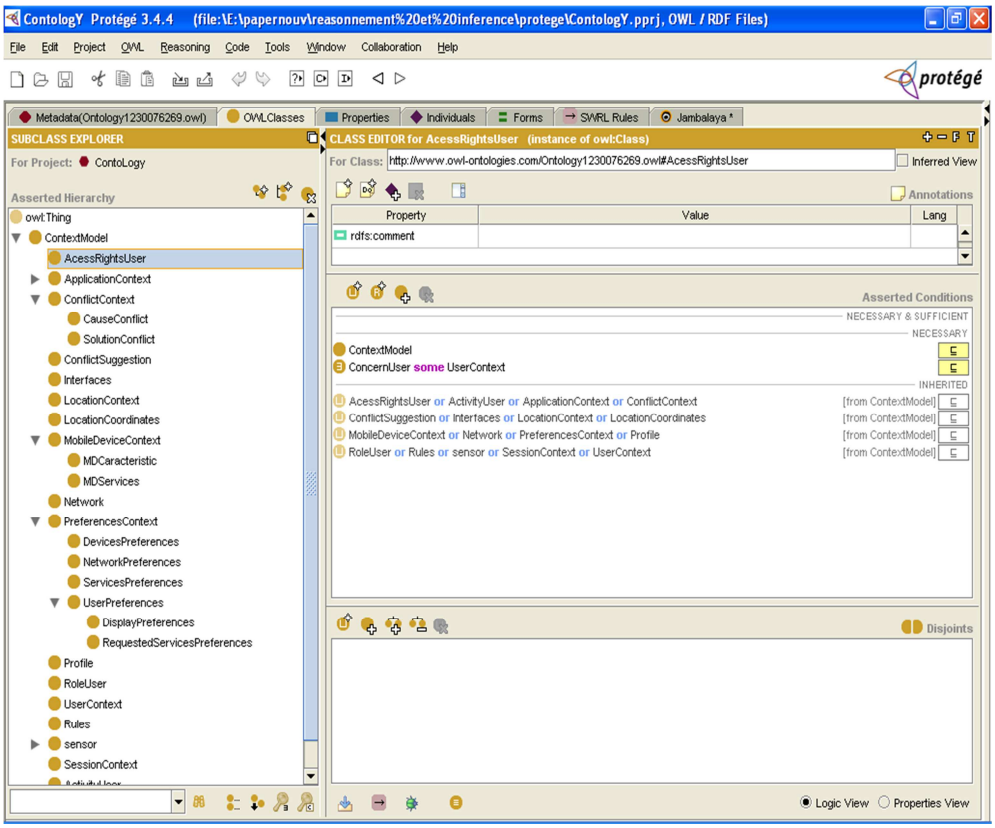


Figure 4.4 – Création des classes

3.4.3 Définition Des Propriétés :

Après avoir construit les classes, nous allons maintenant créer les propriétés pour chacun d'eux, les attributs vont être créés sous PROTEGE 2000 par 'dataTypeProperty' et les relations par 'objectProperty'. La Figure 4.5 montre les propriétés des classes sous PROTEGE 2000.

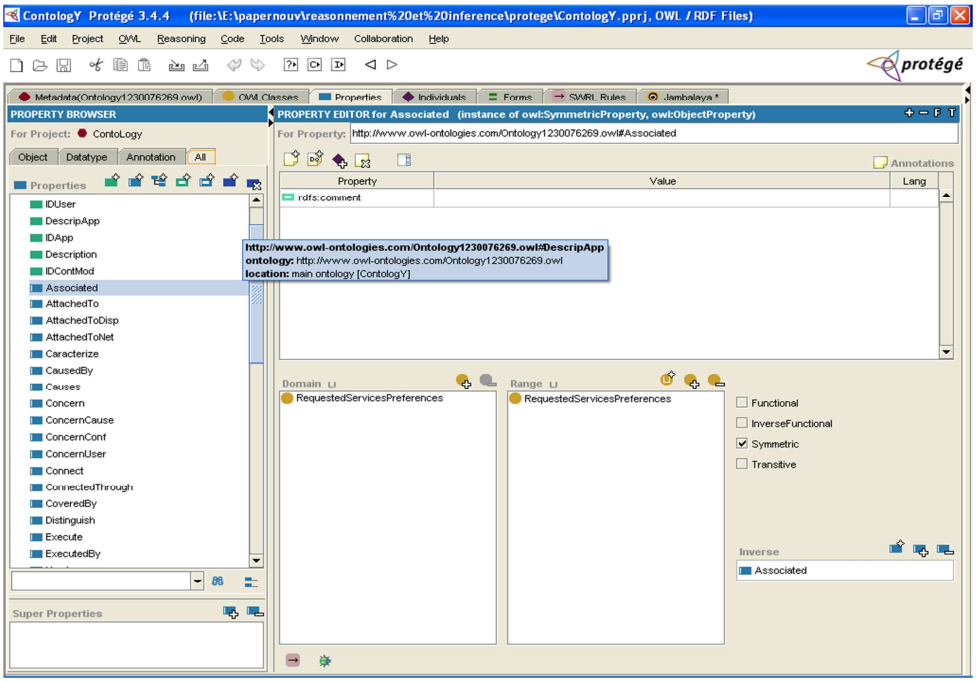


Figure 4.5 - Création des propriétés pour une classe

La Figure 4.6 présente un attribut, son nom, son domaine et son type.

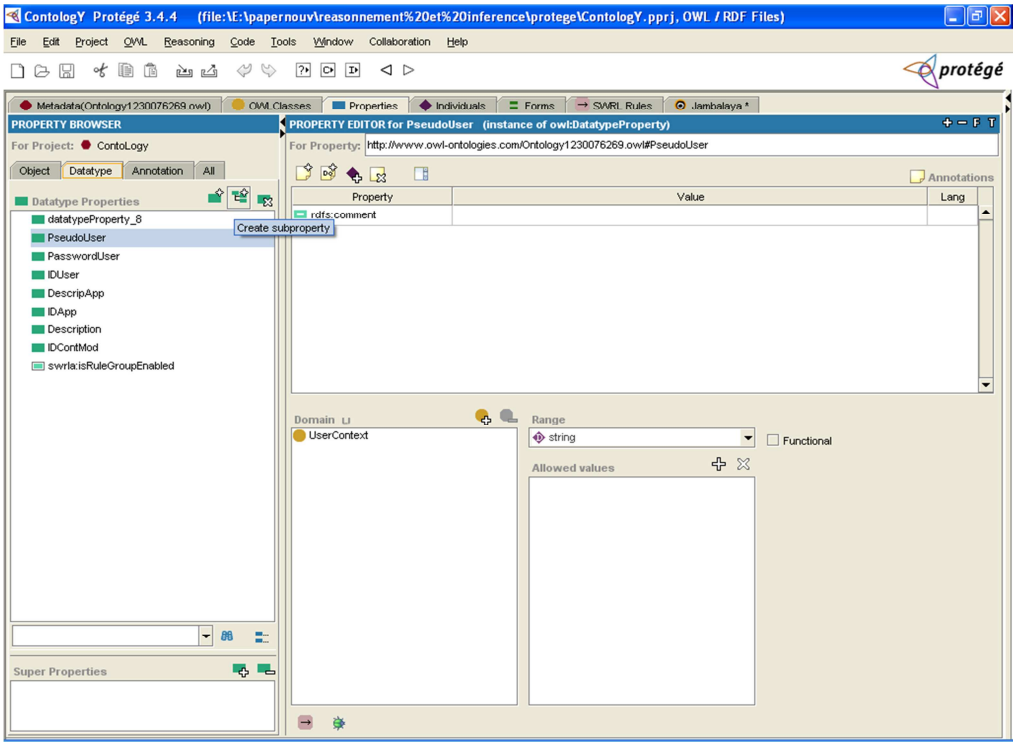


Figure 4.6 – Création d'un attribut

3.4.4 Définitions Des Restrictions :

PROTEGE 2000 nous offre un moyen pour créer des restrictions sur les classes et les propriétés. La Figure 4.7 montre un exemple d'une restriction sur une classe crée sous PROTEGE 2000.

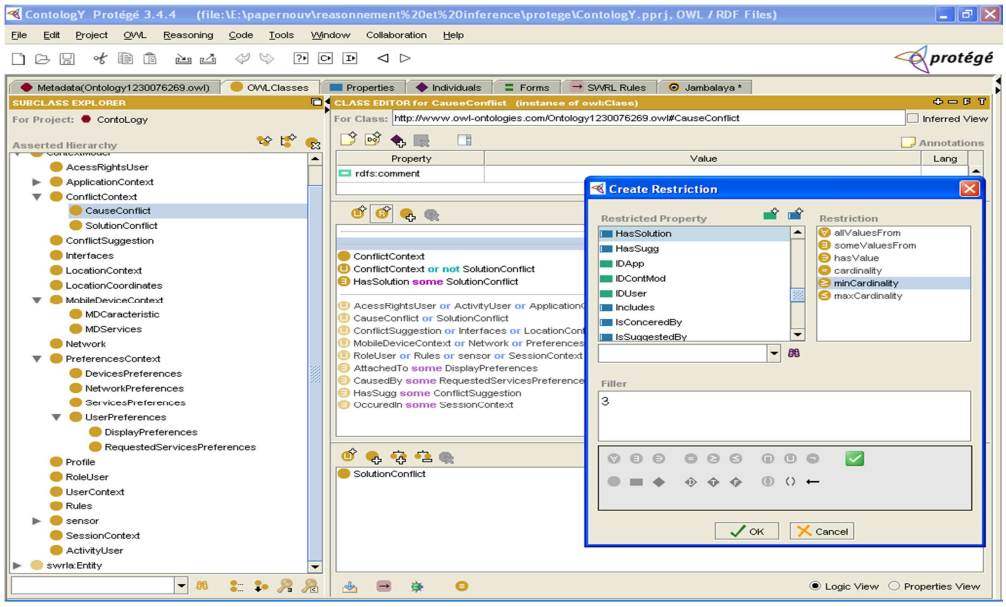
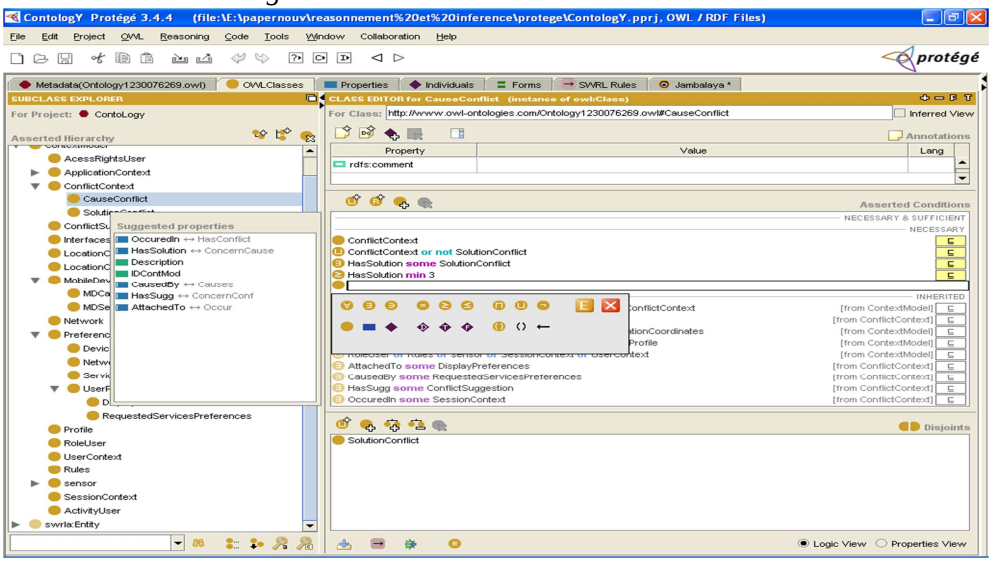
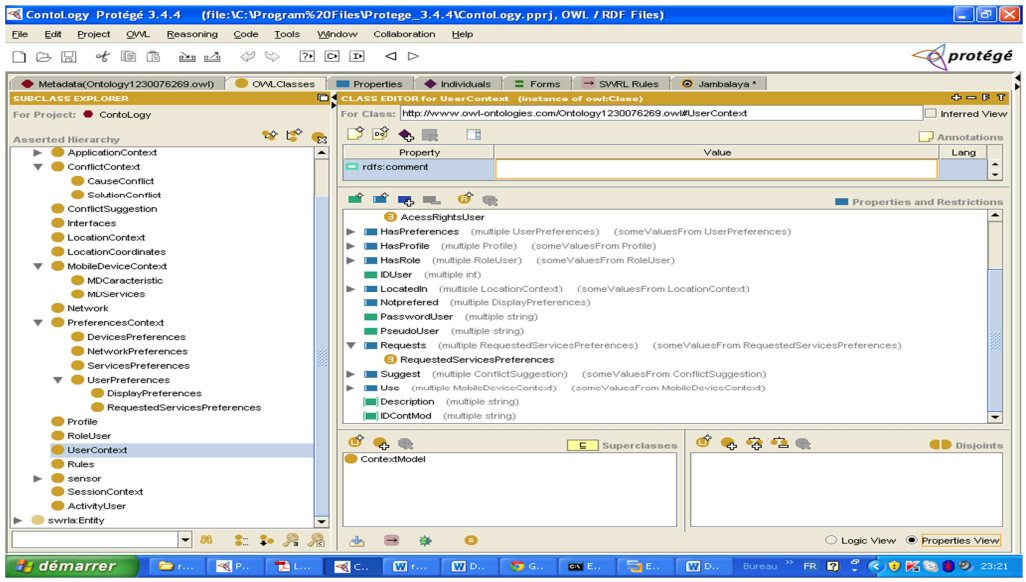
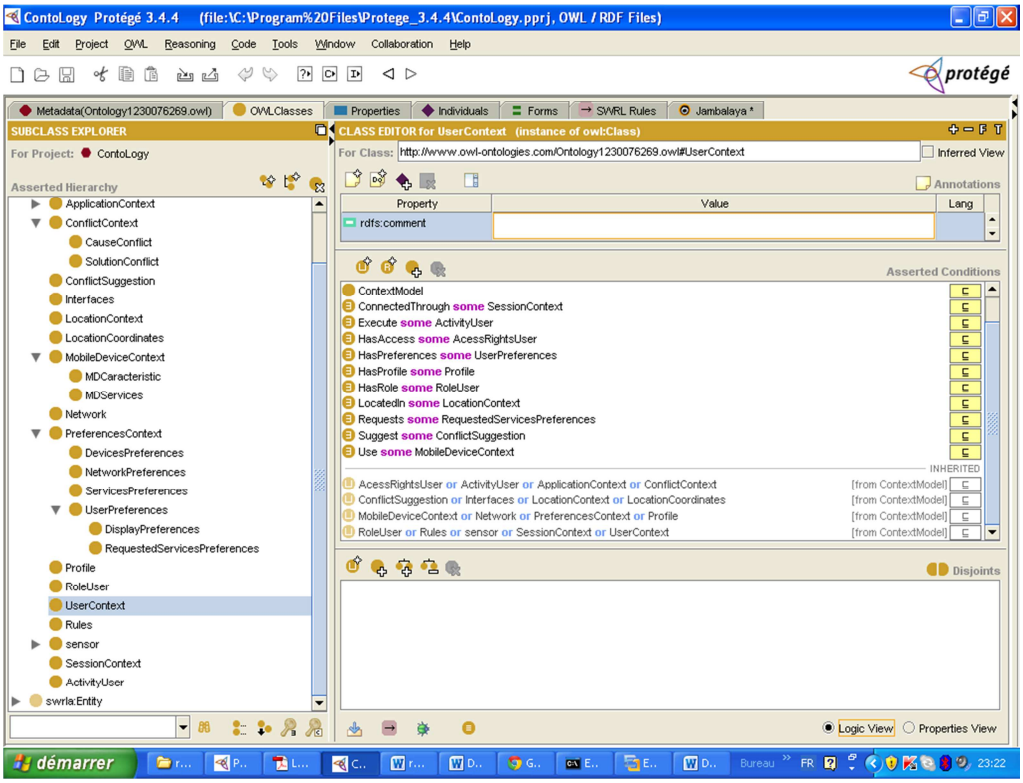


Figure 4.7 – Création d'une restriction sur une classe



➢ Les figures suivantes montrent des exemples sur « ContoLogy »





3.4.5 Création Des SWRL Sous Protégé:

Nous avons utilisé PROTÉGÉ 2000 pour implémenter les règles de la gestion des conflits. Figure 4.8 montre la création des règles SWRL sous protégé

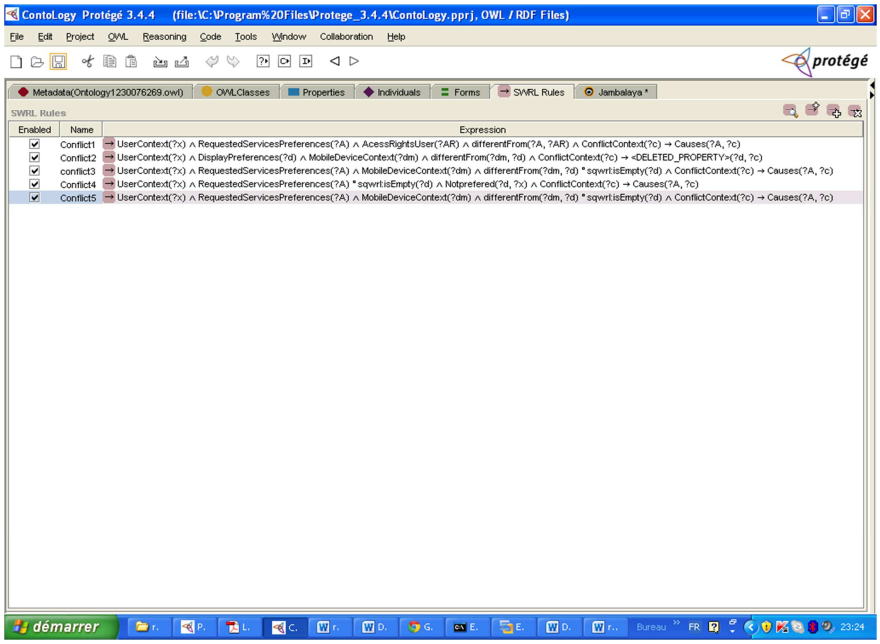


Figure 4.8:SWRL pour gérer les conflits

3.4.6. Génération du code

Après cette étape, nous pouvons transformer notre ontologie “ContoLogy” à un OWL format. Un extrait du modèle de l’ontologie de contexte en OWL est illustré ci-dessous:

```
<?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"/>….
```

3.5 Test De L’ontologie :

Nous avons utilisé le système Racer pour tester l’ontologie « Contology », nous distinguons deux types de tests : test de consistance et test de satisfiabilité; le premier consiste à enlever l’inconsistance entre les concepts et cela en utilisant le test de subsumption incorporé au système Racer, par contre le deuxième permet de vérifier pour chaque concept l’existence des instances; un concept « C » est satisfiable si et seulement s'il existe au moins une interprétation « I » (instance) pour le concept « C ». Racer se présente sous la forme d’un serveur qui peut être accédé par le protocole TCP ou HTTP. Donc, nous devons d’abord configurer la connexion au serveur hébergeant le système Racer. Mais ce que nous souhaitons, est de tester l’ontologie localement. Pour cela nous allons procéder comme suit:

1. Activer le menu OWL de Protégé.
2. Choisir l’option Préférences…
3. Dans la fenêtre qui s’affiche, vérifier que l’URL est « http://localhost:8080 », sinon saisir puis valider en cliquant sur le bouton 'Close'.

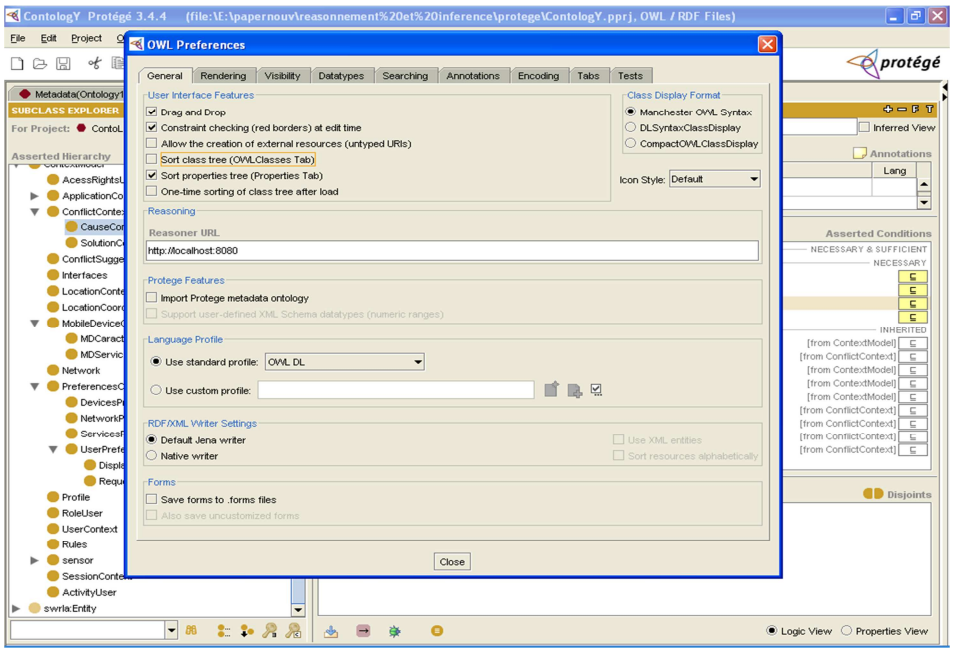


Figure 4.9 – Vérification de l'URL de Reasoner

4. Télécharger et exécuter Racer localement, le service HTTP va être activé sur le port 8080 de la machine local (localhost)

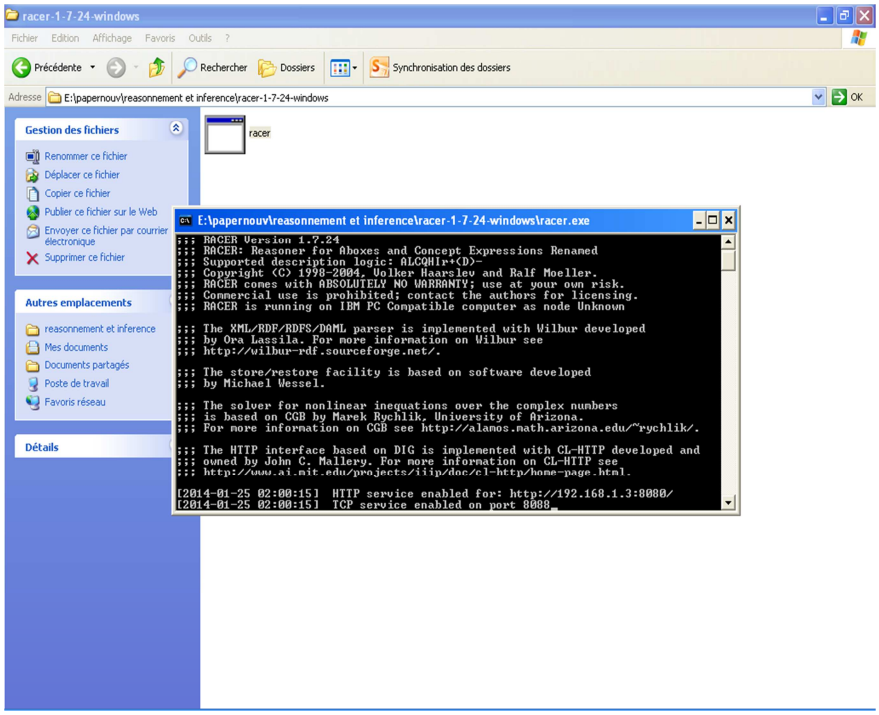


Figure 4.10 – Lancement de Racer

5- Puis, effectuer les 3 tests sur notre ontologie

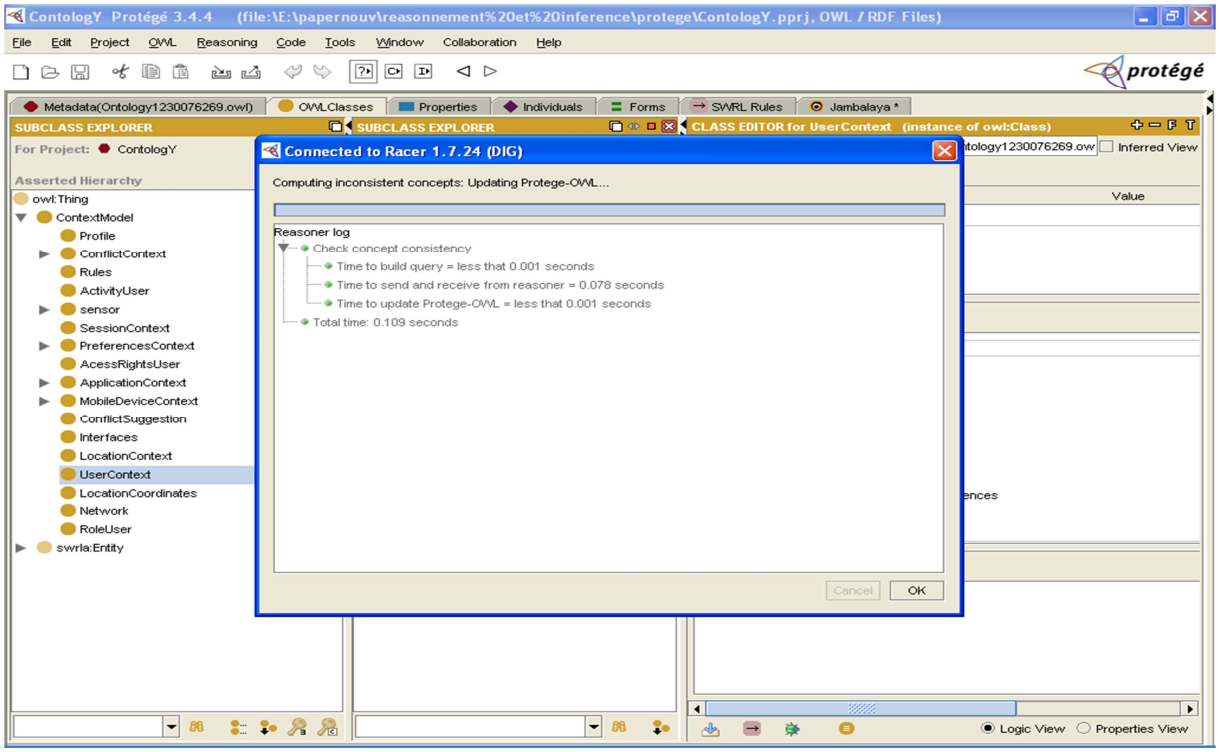


Figure 4.11 – Test de consistance

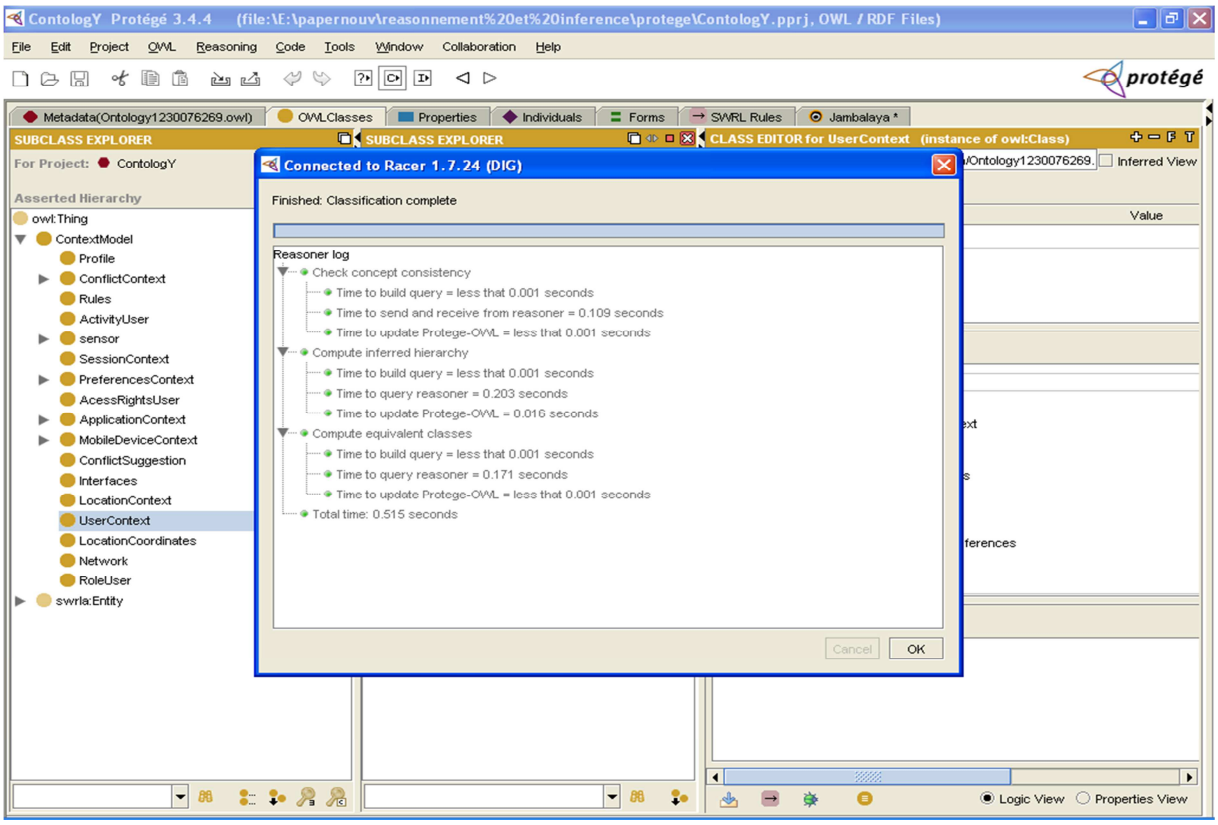


Figure 4.12 – Test de classification

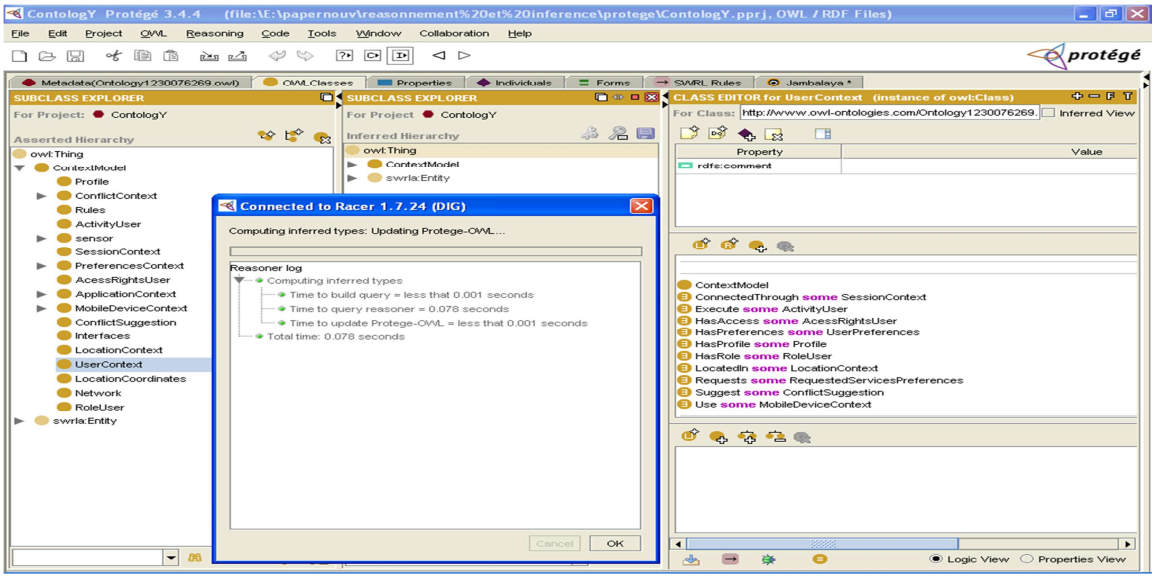


Figure 4.13– Test d'inférence

D'après les tests que nous avons appliqués à l'ontologie « ContoLgy », aucune erreur ne s'est produite lors du test.

4. L'EXPLOITATION DE L'ONTOLOGIE DE CONTEXTE : LE PROCESSUS D'ADAPTATION

Nous exploitons et nous utilisons l'ontologie de contexte "ContoLogy" pour l'adaptation de la requête initiale de l'utilisateur au contexte courant. Ainsi, nous proposons une architecture à base de web service (voir Annexe) pour assurer le processus d'adaptation. En utilisant l'ontologie de contexte "ContoLogy", l'architecture d'adaptation peut résonner sur le contexte de l'utilisateur et adapter la requête initiale à ce contexte courant. Parmi les différents paramètres du contexte, nous focalisons sur : la localisation et le dispositif mobile utilisé(DM).

Après avoir implémenté l'application, il est obligatoire, pour de nombreuses raisons, de la subir au processus d'adaptation. Ces raisons peuvent être classées en quatre catégories [Ketfi02]: (1) l'adaptation correctionnelle, (2) l'adaptation adaptative (3) L'adaptation évolutive et (4) l'adaptation perfective. Dans notre approche, nous sommes intéressés à l'adaptation adaptative afin d'adapter les applications ubiquitaires à leur environnement d'exécution. Nous adoptons ce genre d'adaptation parce que l'application fonctionne correctement, mais son environnement change (par exemple, le contexte de l'utilisateur). Dans ce cas, l'application est adaptée en réponse à des changements dans son environnement d'exécution.

Par conséquent, pour assurer ce processus d'adaptation, nous utilisons l'ontologie de contexte "contology". L'architecture proposée pour l'adaptation est composé de deux parties principales: la partie statique et la partie dynamique.

(1) **la partie statique:** cette partie est décrite par l'ontologie “Contology”. Elle se focalise, d'une part, sur la modélisation de l'information contextuelle des utilisateurs et leurs préférences et, d'autre part, sur la gestion des conflits potentiels qui peuvent survenir entre les préférences des utilisateurs au cours de leur processus de vérification.

(2) **La partie dynamique**: le rôle de cette partie est d'assurer l'adaptation dynamique et fonctionnelle des applications sensibles au contexte. Le processus d'adaptation adoptée par cette partie est basé sur "ContoLogy" afin d'offrir une meilleure réponse à l'utilisateur. En outre, ce processus est assuré par l'adaptation de la requête initiale de l'utilisateur au contexte de l'utilisation et les préférences de l'utilisateur en utilisant l'ontologie "ContoLogy" .La méthodologie dans notre approche consiste en trois étapes principales:

(1) La modélisation du contexte de l'utilisation et la gestion des préférences de l'utilisateur, basée sur une nouvelle définition du contexte qui sépare les données contextuelles des données de l'application en utilisant " ContoLogy "

(2) La résolution des conflits qui peuvent survenir lors de la gestion des préférences.

(3) L'adaptation dynamique et fonctionnelle des applications sensible au contexte à base de web service. L'accomplissement des deux dernières étapes est basé sur “ContoLogy”.

La figure 4.14 montre l'architecture générale de l'approche proposée.

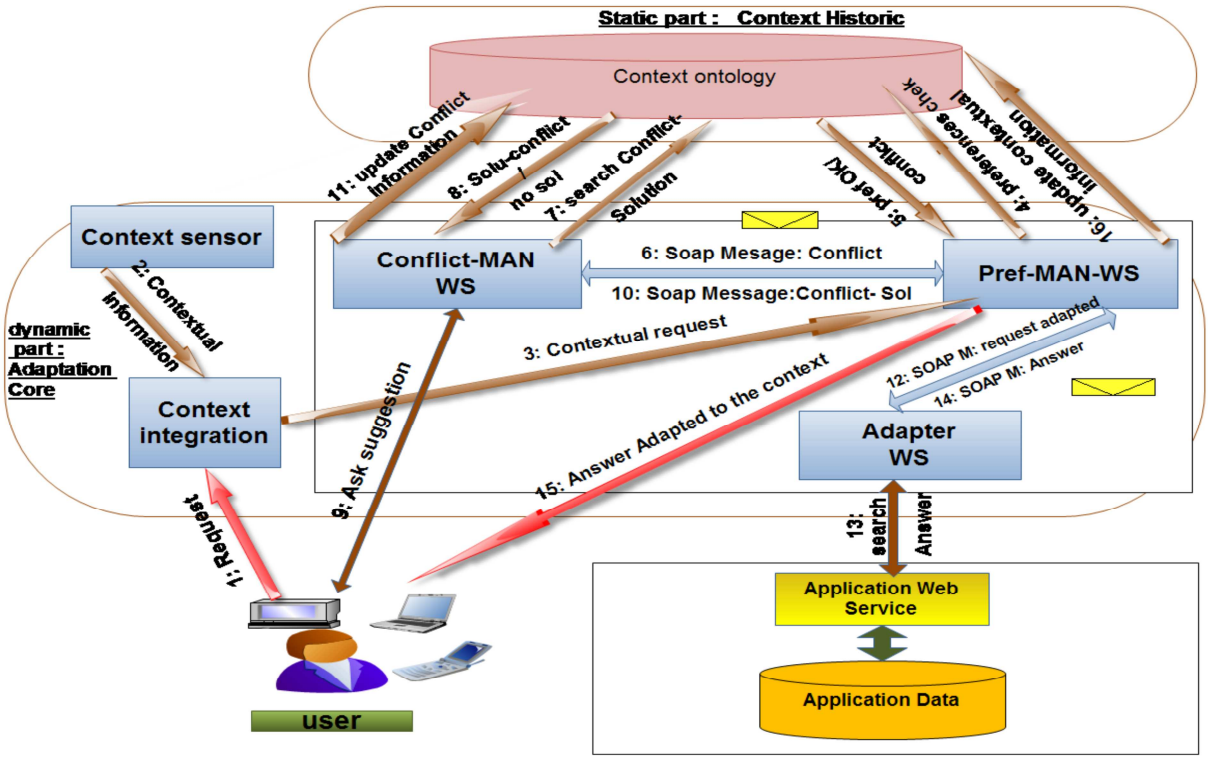


Figure 4.14 L'architecture de notre approche

4.1 Les Etapes de Processus d'Adaptation

Dans cette section, nous allons présenter les 16 étapes de notre approche pour l'adaptation :

(1) ***Request***: l'utilisateur envoie sa requête à la plateforme de notre architecture, en utilisant son DM. le module Context integration (CI) reçoit cette requête.

(2) Contextual information: le module Context sensor envoie les informations contextuelles de l'utilisateur au module Context integration, à savoir : le DM utilisé et la localisation.

(3) ***Contextual request***: Dans cette étape, le module Context Integration augmente la requête de l'utilisateur en lui ajoutant les informations contextuelles, le résultat de cette étape est une requête contextuelle. Le module (CI) envoie cette requête au Preferences Management Web Service (PMWS).

(4) ***Preferences check***: dans cette étape, le PMWS vérifie la requête contextuelle en utilisant l'ontologie de contexte "ContoLogy". Il vérifie la conformité entre les préférences de l'utilisateur et ses droits d'accès et le type du DM utilisé.

(5) ***Prefrences OK/ Conflict***: en consultant l'ontologie, le PMWS peut détecter que les préférences sont vérifiés ou il peut trouver un conflit.

(6) ***Soap Message: Conflit***: dans le cas de conflit, le PMWS envoie un message SOAP contenant le conflit au Conflict Management Web Service (CMWP).

(7) ***Search Conflict Solution:*** en utilisant l'ontologie de contexte, le CMWS cherche une solution pour le conflit détecté.

(8) ***Solution conflict/ no solution:*** dans cette étape, on a deux cas, que soit une solution pour le conflit, sinon pas de solution.

(9) ***Ask suggestion:*** dans le cas ou pas de solution pour le conflit, le CMWS demande une suggestion de solution de l'utilisateur.

(10) ***Soap message: conflict solution:*** dans cette étape, si l'utilisateur envoie une suggestion de solution pour le conflit à le CMWS, ce dernier l'envoie au PWSM.

(11) ***Update conflict information:*** le CMWS mis à jour les informations sur le conflit en ajoutant les informations du conflit de la session courante.

(12) ***Soap message: request updated:*** le PMWS envoie la requête de l'utilisateur après la vérification à l'adapter web service (AWS).

(13) ***Search answer:*** l'AWS cherche une réponse pour la requête de l'utilisateur.

(14) ***Soap message: answer:*** une fois la réponse est trouvée, l'AWS l'envoie au PMWS.

(15) ***Answer adapted to the context:*** ce dernier, envoie cette réponse adaptée au contexte à l'utilisateur.

(16) ***Update contextual information:*** finalement, le PMWS mis à jour les informations contextuelles en ajoutant à l'ontologie de contexte les informations de contexte de la session en cours.

4.2 Présentation Du Processus D'adaptation

Dans cette section, nous présentons la partie dynamique de notre approche, qui adapte les applications sensibles au contexte à ce dernier. Le contexte d'utilisation d'un utilisateur nomade, en plus d'être composé de multiples aspects, est très variable et en constante évolution, ce qui rend l'opération d'adaptation de l'application en question une tâche difficile. Afin d'assurer ce processus d'adaptation et pouvoir changer le comportement de tel application sensible au contexte d'utilisation, nous proposons dans le cadre de ce travail, d'utiliser les Web Services lors du développement de ce type d'applications et nous suggérons aussi d'utiliser un Web Service [Ponge04] qui assure cette adaptation.

La partie dynamique de notre approche est composée de trois services Web : Preferences Manager Web Service (PMWS), Conflicts manager Web Service (CMWS) et Adapter Web Service (AWS) et deux modules: Context integration et context sensor.

4.2.1. Preferences Manager Web Service

Ce service Web est chargé de la gestion des préférences. Par conséquent, il assure la vérification des préférences de l'utilisateur en utilisant la requête initiale de l'utilisateur et "ContoLogy", le PMWS peut raisonner sur le contexte de l'utilisateur. Le PMWS peut analyser le contexte de l'utilisateur qui apparaît dans la requête contextuelle de l'utilisateur. Par conséquent, il vérifie la conformité entre les préférences demandées et le contexte d'utilisation, principalement: le DM utilisé, la localisation et ses droits d'accès. Cette étape peut générer des conflits qui peuvent être détectés par PMWS. En outre, il reformule la requête initiale de l'utilisateur, dans le cas de conflits, en ajoutant les nouvelles préférences. Il envoie à l'utilisateur la réponse adaptée au contexte, et stocke le nouveau contexte pour l'utiliser dans les prochaines sessions, lorsque nous recevons le même contexte et requête (voir Figure 4.15).

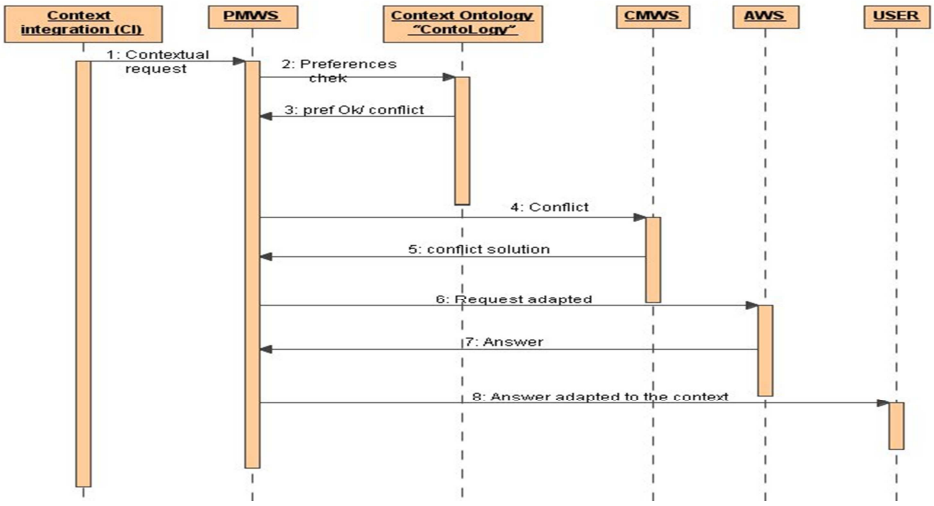


Figure 4.15. Le diagramme de séquence de PMWS.

4.2.2 Conflicts Manager Web Service

Le rôle de ce service Web est de gérer les conflits qui peuvent apparaître entre les préférences de l'utilisateur. Les conflits sont gérés, par notre approche, selon le diagramme de séquence suivant (figure 4.16). Plus précisément, notre approche gère cinq conflits (voir chapitre 4, tableau 4.2). Ce service web exécute la solution proposée pour chaque conflit survenu (voir chapitre 4, tableau 4.3). Après avoir reçu un message contenant le conflit qui a eu lieu, le CMWS raisonne et déduit une solution à ce conflit survenu en utilisant "ContoLogy", sinon; il implique l'utilisateur pour donner ses suggestions pour ce conflit. S'il n'y a pas de suggestions, il prend une solution par défaut, pour chaque conflit (à savoir notre approche propose une solution déterminée (voir tableau 4.3)). A la fin, il envoie un message qui contient la solution du conflit au PMWS. Par conséquent, il met à jour l'historique des

informations sur les conflits. Ce service web assure: la résolution des conflits à l'aide de "Contology", et le stockage de l'information du conflit survenu.

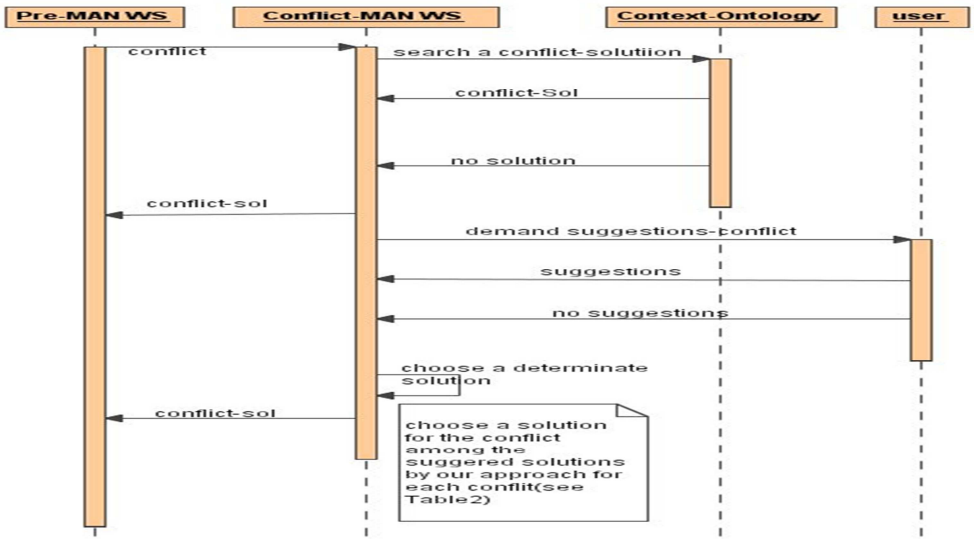


Figure 4.16 Le diagramme de séquence de CMWS.

4.2.3 Adapter Web Service

Son rôle est de retourner une réponse adapté à l'utilisateur. Il exécute les étapes suivantes: premièrement, il accède au Services Web de l'application et cherche dans le WSDL de ces derniers, afin d'en extraire des services Web avec leurs interfaces, leurs activités et le nombre d'interfaces spécifiques à chaque service Web. Deuxièmement, le choix du service Web qui répond mieux à la requête de l'utilisateur. Ensuite, il reformule et envoie à PMWS la réponse adaptée au contexte d'utilisation.

4.2.4 Context Sensor

Ce module est responsable de la capture du contexte de l'utilisateur au moment de la connexion, à savoir: la localisation, DM, session. Ensuite, il envoie cette information contextuelle au module " Context integration ". Il est composé de deux sous-modules suivants:

1- *Logical context sensor*: un ensemble d'interfaces utilisé par l'utilisateur pour entrer son contexte.
2- *The physical context sensor*: un ensemble de dispositifs physiques utilisés pour capturer le contexte de l'utilisation.

4.2.5 Context Integration

Ce module reçoit la requête initiale de l'utilisateur et la reformule en ajoutant les informations contextuelles. Ensuite, il envoie cette demande contextuelle au PMWS.

CONCLUSION

Dans ce chapitre, nous avons détaillé notre proposition de modélisation de contexte. Nous avons commencé par énumérer les motivations pour les ontologies dans la section1. La section2 a fait l'objet de la représentation du modèle de contexte. Ensuite, nous avons présenté plus précisément le processus de construction de notre ontologie, qui modélise le contexte d'un utilisateur nomade et présente des solutions pour résoudre les conflits qui peuvent survenir entre les préférences de ce dernier, dans la section3.

Nous avons aussi détaillé notre contribution pour l'exploitation de « ContoLogy » et l'adaptation des applications sensibles au contexte, dans la section4

Le chapitre suivant est consacré pour l'implémentation, il va détailler une ETUDE DE CAS : THE TRAVEL BOOKING APPLICATION.

CHAPITRE 5

ETUDE DE CAS : THE TRAVEL BOOKING APPLICATION

INTRODUCTION

Dans le chapitre 4, nous avons présenté en détail notre proposition pour la modélisation du contexte et la gestion des conflits, qui est une ontologie « ContoLogy ». Et nous avons détaillé notre architecture pour l'adaptation qui exploite l'ontologie de contexte dans le processus de l'adaptation.

Dans ce chapitre, nous allons montrer en utilisant une étude de cas, comment notre approche adapte la requête de l'utilisateur au contexte, en utilisant l'ontologie de contexte « ContoLogy ».

Tout d'abord, nous allons présenter l'environnent de l'implémentation, ainsi que les outils utilisés. Ensuite, nous allons détailler notre étude de cas qui utilise une application de : TRAVEL BOOKING.

1. ENVIRONMENT DE L' IMPLEMENTATION :

Dans cette section nous allons présenter l'environnement et les outils que nous avons utilisé pour implémenter notre architecture.

1.1 Microsoft Visual Studio:

Visual Studio (voir figure 5.1) est un ensemble complet d'outils, pour le développement des applications ASP.NET Web, les services Web à base d'XML, les applications de bureau et les applications mobiles. Visual Basic, Visual C ++, Visual C # et Visual J # utilisent tous le même environnement de développement intégré (IDE), qui leur permet de partager des outils et leur facilite la création de solutions de langage mixte. En outre, ces langages sont influés par les fonctionnalités de .NET Framework, qui à son tour offre un accès aux technologies clés, qui simplifient le développement d'applications Web ASP et des services Web à base d'XML.

Figure 5.1: le logo de Microsoft visual studio

- *Visual Web Developer*

Visual Studio dispose d'un nouveau concepteur de pages Web nommé Visual Web Developper qui comprend de nombreuses améliorations pour la création et l'édition des pages Web ASP.NET et de pages HTML. Il offre un moyen plus rapide et plus simple pour la création des pages Web Forms.

- *Smart Device Applications*

L'environnement intégré Visual Studio a inclue des outils pour les périphériques de cible comme : les PDAs et les Smartphones. Les améliorations comprennent essentiellement: les outils visuels de C++, un nouvel émulateur et des outils de traitements de données.

- *Web Forms*

Web Forms sont une technologie ASP.NET utilisée pour la création des pages Web programmables. Web Forms se rendent comme navigateur compatible HTML et de script, qui permet à tout navigateur sur toute plate-forme d'afficher les pages.

- *Windows Forms*

Windows Forms est un outil utilisé pour la création des applications Microsoft Windows sur le .NET Framework. Ce Framework fournit une orientée objet et un ensemble extensible claire de classes qui permet de développer des applications Windows riches.

- *XML Web Services*

Ils sont des applications qui peuvent recevoir les demandes et les données en utilisant XML sur HTTP. XML Web Services ne sont pas liés à une technologie de composant particulier ou d'une convention d'objet, par conséquent, ils peuvent donc être consultés par tout langage, modèle de composant ou système d'exploitation. Dans Visual Studio, vous pouvez rapidement créer et inclure des XML services Web en utilisant Visual Basic, Visual C #, JScript, ou ATL Server.

- *C#*

C # est un langage orienté objet élégant et de type sécurisé qui permet aux développeurs de créer une variété d'applications sûres et robustes qui fonctionnent sur le .NET Framework. Il est utilisé pour créer des applications Windows traditionnelles de la clientèle, les XML services Web, composants distribués, applications client-serveur, les applications de base de données. Visual C # 2010 fournit un éditeur avancé de code, des concepteurs d'interface utilisateur pratique, débogueur intégré, et de nombreux autres outils pour faciliter le développement d'applications basé sur la version 4.0 du langage C # et la version 4.0 du .NET Framework.

1.2 Protégé

Protégé2000 [Noy01] est une interface modulaire, développée au StanfordMedicalInformatics de l'Université de Stanford7, permettant l'édition, la visualisation, le contrôle (vérification des contraintes) d'ontologies, l'extraction d'ontologies à partir de sources textuelles, et la fusion semi-automatique d'ontologies. Le modèle de connaissances de Protégé2000 est issu du modèle des frames et contient des classes (concepts), des slots (propriétés) et des facettes (valeurs des propriétés et contraintes), ainsi que des instances des classes et des propriétés. De nombreux plug-ins sont disponibles ou peuvent être ajoutés par l'utilisateur. (Voir chapitre 3, section 7.4)

Figure 5.2 interface de Protégé

2. ÉTUDE DE CAS

Dans cette section, nous présentons en utilisant une étude de cas, la façon dont nous avons exploité l'ontologie "ContoLogy" pour l'adaptation de la requête de l'utilisateur. Tout d'abord, nous allons présenter l'application : « travel booking », que nous avons développé et nous allons expliquer les étapes de l'implémentation, en utilisant un exemple détaillé, de la réception de la requête de l'utilisateur en passant par la résolution des conflits, jusqu'à la réception de la réponse adaptée par l'utilisateur. Néanmoins avant d'expliquer le cas d'étude choisi, il convient de détailler les motivations pour l'utilité des services Web.

2.1. Motivations pour l'utilisation des Service Web

Dans le cadre de notre approche, nous optons pour les Services web pour les avantages qu'ils procurent. Les principales motivations d'utilisation de ce paradigme peuvent être résumées aux points suivants:

1. Le but principal de l'approche Service Web est de rendre le Web un dispositif distribué de calcul dont l'objectif est de permettre aux programmes (services) d'interagir d'une façon intelligente avec la possibilité de : se découvrir automatiquement, de négocier entre eux et de se composer en des services plus complexes [Kouadri03].

2. L'utilisation des services web facilite la communication dans les environnements hétérogènes. De plus, les services web ont la possibilité de s'implémenter sur différentes plates-formes en utilisant différents langages. Aussi, ils facilitent l'interopérabilité entre systèmes et plates-formes hétérogènes [Kadima03].

3. Pour le bon fonctionnement, les Services web utilisent des protocoles *Web standard* (HTTP et TCP/IP) *et XML*. Actuellement, plusieurs entreprises ont déjà une infrastructure Web, alors, le personnel a donc les connaissances et l'expérience de sa maintenance. C'est pour cette raison que le coût d'accès aux Services web est très baissé par rapport aux technologies précédentes.

2.2. L'Application Travel Booking

Travel booking est une application basée sur les services Web pour gérer une agence de voyage et de réservation en ligne (voir figure 5.3 *[Menad08]*). Elle permet à l'utilisateur d'effectuer une réservation de vol et réservation d'hôtels. Cette application est adaptée par notre architecture au contexte et au profil de l'utilisateur. En utilisant cette application, l'utilisateur peut rechercher un vol, un hôtel et une voiture, et il peut recevoir une réponse adaptée à son contexte, par exemple: adapté à: son emplacement, sa ville et l'emplacement de l'aéroport. Par exemple: l'utilisateur peut recevoir une liste des hôtels situé près de l'aéroport.

Pour la conception des services d'agence, nous avons distingué trois services web: (1) *Airline Service*: il offre des services chargés de la gestion en ligne des réservations des vols des clients. (2) *Hotel Service:* Il offre des services qui ont comme fonction, le contrôle en ligne des hôtels et des réservations des clients. (3) *Location Car service:* Il classe tous les services chargés de la gestion en ligne de voitures et emplacement.

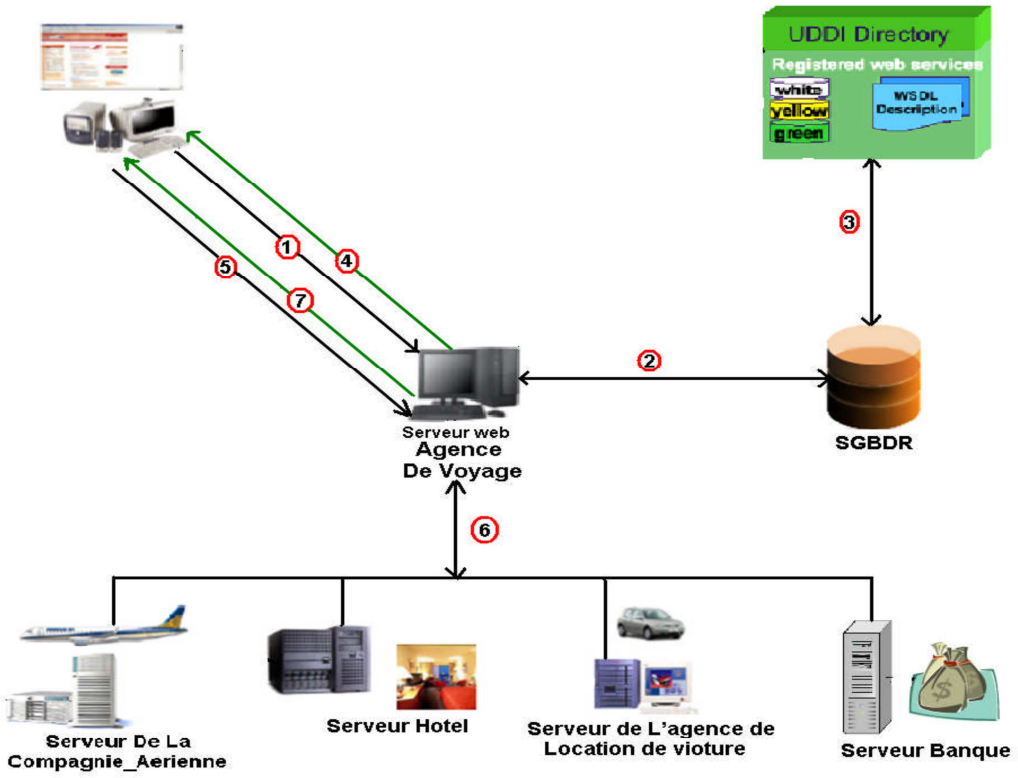


Figure 5.3. L'architecture globale de l'application

Nous avons créé un portail de services qui sert comme une passerelle vers divers services web. Ce portail ne stocke aucune donnée sur sa base physique, mais agit comme un

prestataire de services. L'application que nous avons développé permet à un client d'éviter de faire plusieurs recherches sur le web (compagnies aériennes, hôtels, voitures ...), de planifier son voyage. Le portail que nous avons implémenté fournit les interfaces nécessaires à la planification de voyage à travers l'utilisation des technologies de services Web. Cette application sera utilisée par notre architecture pour l'adaptation au contexte d'utilisation et le profil de l'utilisateur.

La partie dynamique de notre approche garantit le processus d'adaptation. Tous les services Web liés à la partie dynamique qui sont nécessaires pour valider notre approche sont créés en utilisant Microsoft Visual Studio. Plus précisément, trois services web ont été créés pour gérer l'interaction et les messages entre l'utilisateur et l'application.

Après la création des services Web, une page de C # apparaîtra sur lequel nommé Service1.asmx.cs. La page contient la bibliothèque que nous avons besoin et le code de service Web. Dans ce qui suit, nous présentons un exemple du WSDL du PMWS et un message SOAP.

```
<wsdl:definitionstargetNamespace="http://tempuri.org/"><wsdl:types>
s:schemaelementFormDefault="qualified" targetNamespace="http://tempuri.org/">
<s:element name="context_infos"><s:complexType><s:sequence>
<s:elementminOccurs="0" maxOccurs="1" name="cont_num" type="s:string"/>
sequence></s:complexType></s:element><s:element name="context_infosResponse">
"tns:ArrayOfString"/></s:sequence></s:complexType></s:element>
```

Soap message:

```
wsdl:message name="context_infosSoapIn"><wsdl:part name="p
element="tns:context_infos"/></wsdl:message><wsdl:message name="context_infosS
<wsdl:part name="parameters" element="tns:context_infosResponse"/></wsdl
…….
```

2.3 Déroulement du Processus d'Adaptation : Exploitation de l'Ontologie.

Dans cette section, nous allons expliquer notre approche pour gérer les préférences et les conflits, et les détails du processus de l'adaptation, en utilisant un exemple qui explique les interactions entre les services Web de notre architecture, l'application ubiquitaire ; l'ontologie de contexte "ContoLogy" et l'utilisateur. Pour cela, nous présentons un exemple qui comporte essentiellement les points suivants:

(1) L'interaction entre l'utilisateur et la partie dynamique et l'ontologie de contexte "ContoLogy".

(2) La réception et la vérification du requête de l'utilisateur.

(3) la résolution des conflits. Pour cela, nous prenons un conflit qui peut se produire et nous expliquons comment le système va gérer ce conflit et nous allons voir comment le système résout ce conflit, étape par étape.

(4) L'adaptation de la réponse de l'application à l'information contextuelle. Dans ce cas, nous allons prendre comme exemple:

Le ConflictContext(ConflictContext_2)="Contradiction entre display préférences et les caractéristiques du dispositif mobile DM"

1. Cause par*CauseConflict(CauseConflict_2)=* "l'utilisateur demande un affichage qui ne convient pas avec le DM utilisé"
2. avec:
    1. La solution *SolutionConflict_3*= "ContoLogy" qui stipule: "résonner et inférer une meilleure solution à partir de "ContoLogy"".
    2. Sinon, alors *SolutionConflict_1="suggestion"* qui stipule, "demande une suggestion de l'utilisateur".
    3. Sinon, alors *SolutionConflict_4=" un display_preference par défaut"*

2.3.1 Interaction Entre l'utilisateur, La Partie Dynamique Et l'Ontologie De Contexte

À la première connexion de l'utilisateur avec notre système, le système lui demande d'être enregistré, en donnant ses informations personnelles telles que : le nom, l'adresse, l'e-mail et choisir ses services et ses préférences. Le système va automatiquement obtenir les caractéristiques du DM à partir du fichier d'information du DM. Les caractéristiques de DM dans l'ontologie seront enregistrées comme suit:

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

Après, l'utilisateur choisi: le service***« recherche d'un vol et un hôtel »***, avec la préférence d'affichage de format ***« image »***. Aussi, il choisit le service: ***« vol et la chambre dans l'hôtel ».*** L'information de l'utilisateur et l'information de RequestedServicesPreferences et son display preferences, vont être ajouté à l'ontology "ContoLogy":

1. ***Utilisateur:***

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".

2. ***La préférence de Service "Montrer vol":***

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" .

On peut remarquer que le service choisi a un service associé "preser_i043502", qui est le service "montrer hôtel".

3. ***Service Preference* "Montrer 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".

4. Display Preference pour : "montrervol" et "montrer hotel":

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

2.3.2 Vérification de la requête de l'utilisateur

Après que l'utilisateur s'identifier, la figure suivante présente le résultat de la recherche d'un Vol.

Figure 5.4. Résultat de la recherche d'un Vol

Après le clic sur "show details", le PMWS reçoit la requête et les informations contextuelles de l'utilisateur, et les vérifie avec les préférences de l'utilisateur et l'ontologie "ContoLogy", selon les étapes suivantes:

(1) PMWS reçoit le ID du service et les informations contextuelles (localisation et DM utilisé) par la méthode "Service_check". Cette méthode retourne le Service_Associé et DisplayPreference (figure 5.5).

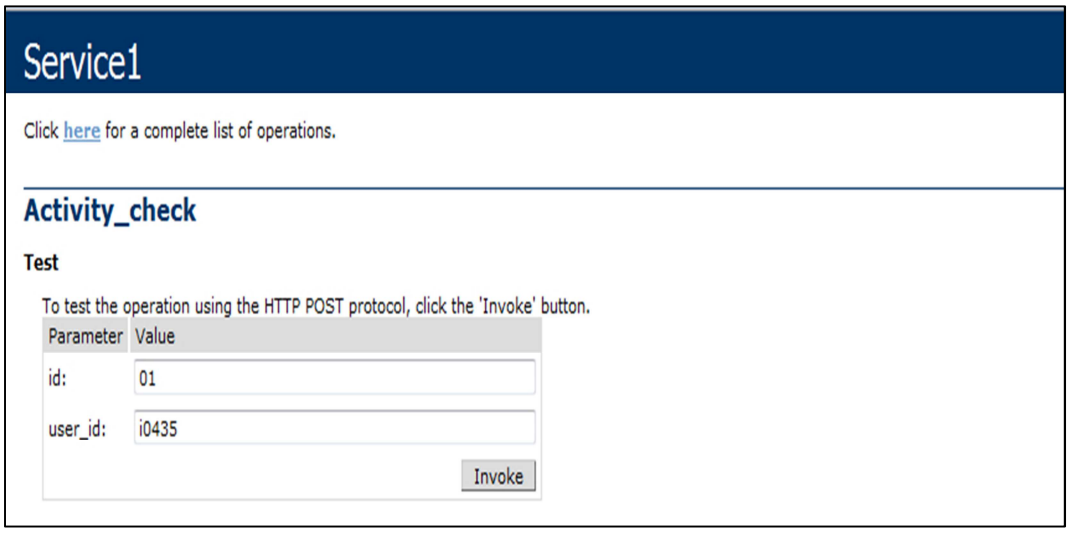


Figure 5.5 .l'appel de la méthode "Service_check"l

(2) La figure suivante présente le message soap reçu par le PMWS

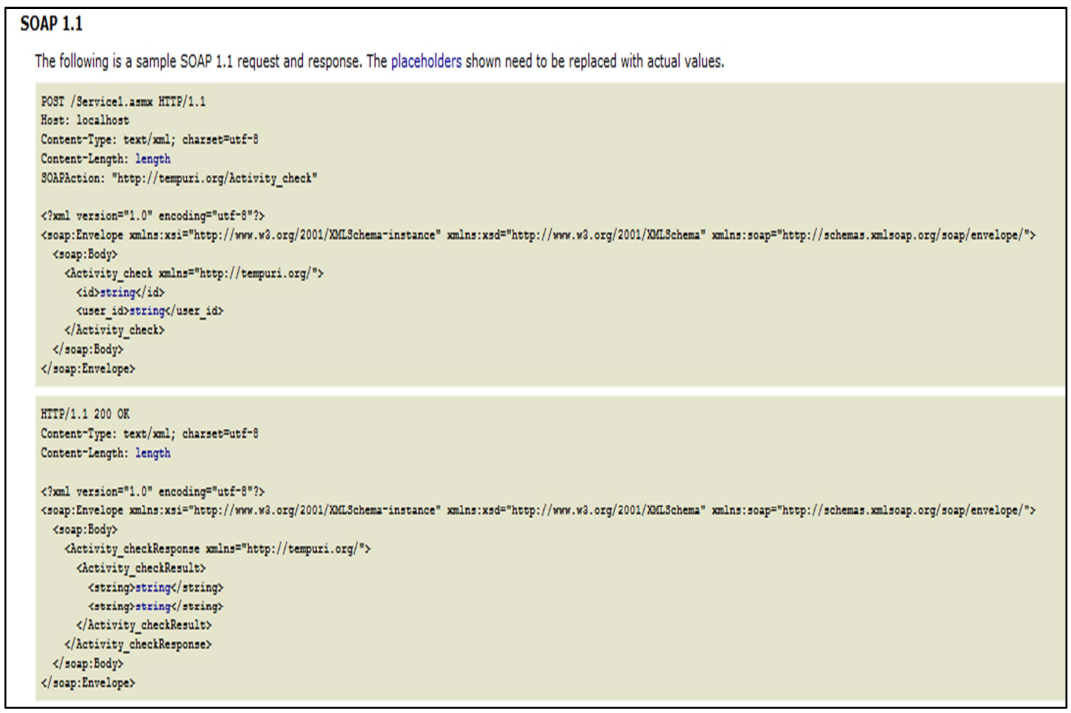


Figure 5.6. La méthode Service_check SOAP 1.1

(3) dans la figure 5.7, on trouve le résultat reçu par le PMWS après la vérification de la requête de l'utilisateur en utilisant "ContoLogy".

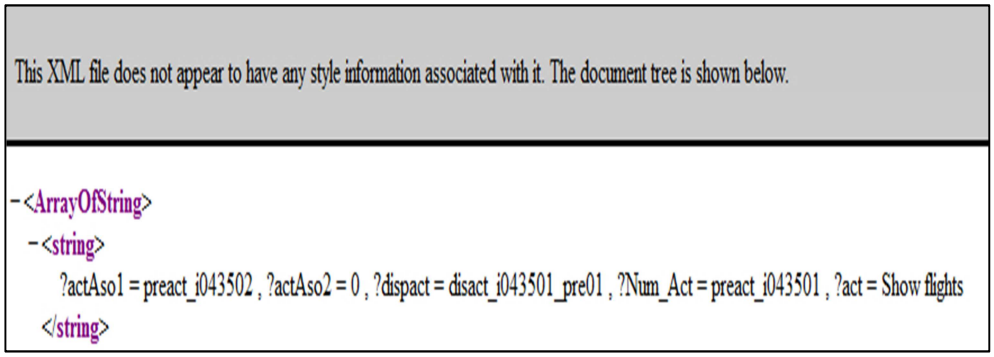


Figure 5.7. Le résultat de" Service_check »

(4) Dans l'étape suivante, le PMWS compare la valeur retournée de la méthode "MD_check", et la méthode "display_check". Dans notre exemple, les valeurs ne doivent pas être les même, car le DM de l'utilisateur ne supporte pas l'affichage « image », que sa valeur est 0. (figure 5.8).

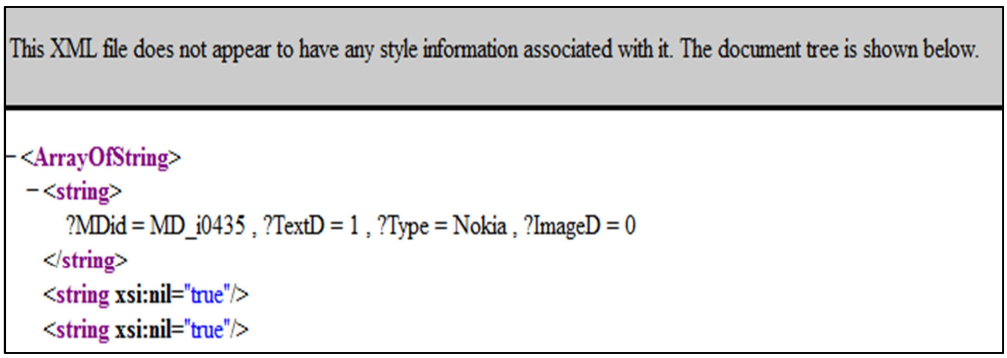


Figure 5.8. le résultat de "MD_check".

(5) les formats "text" et " image" dans " display preference"ont les deux la valeur 1 (figure 5.9).

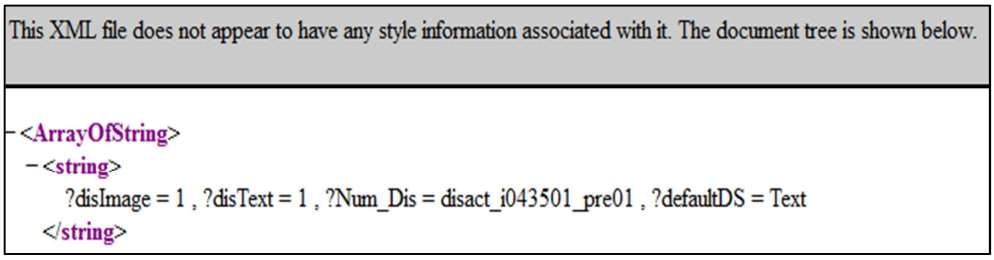


Figure 5.9. Le résultat de "Display_check"

2.3.3 Le Conflit entre les Caractéristique du DM et Display Préférences

Dans cette étape, le PMWS doit détecter le conflit entre le display_preference et les caractéristiques du DM (figure 5.10) et (figure 5.11). PMWS envoie le conflit à CMWS, qui doit vérifier le conflit et choisir sa solution dans l'ontologie "ContoLogy". Le système doit vérifier l'historique de l'utilisateur en utilisant la méthode History_check" pour les services similaires et ses préférences. Si non solution de l'historique de l'utilisateur, le système doit demander la suggestion de l'utilisateur. (Figure 5.10).

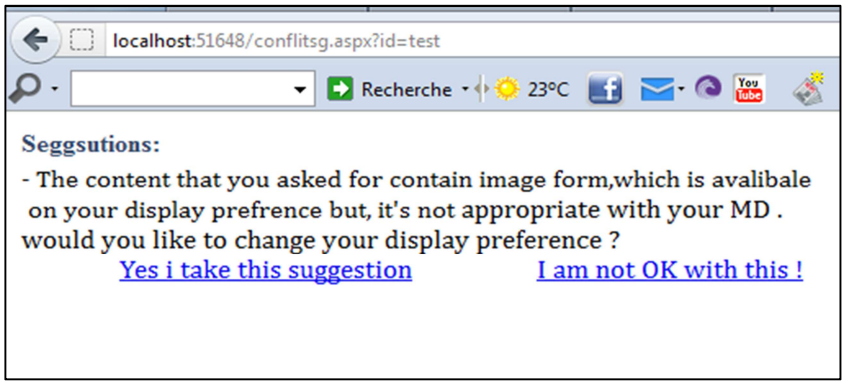


Figure 5.10 Suggestions pour le Conflit

Si l'utilisateur suggère une suggestion, le CMWS envoie à le PMWS la suggestion par la méthode "change_cont_info" pour modifier le display_preference et changer l'image de l'affichage à la valeur 0 (figure 5.11).

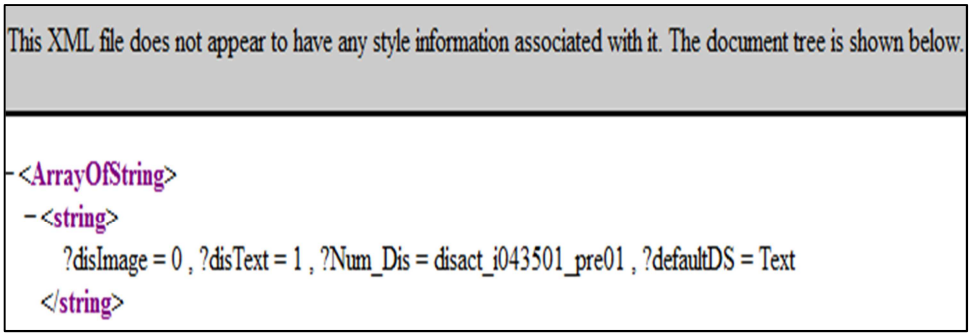


Figure 5.11. vérification du résultat d'affichage après la MAJ.

2.3.4 L'adaptation De La Requête De L'utilisateur

Après la MAJ du display_preference, le PMWS reformule la requête en ajoutant l'information contextuelle et l'envoie à l'AWS pour avoir le résultat adapté de travel-booking application (voir figure 5.12).

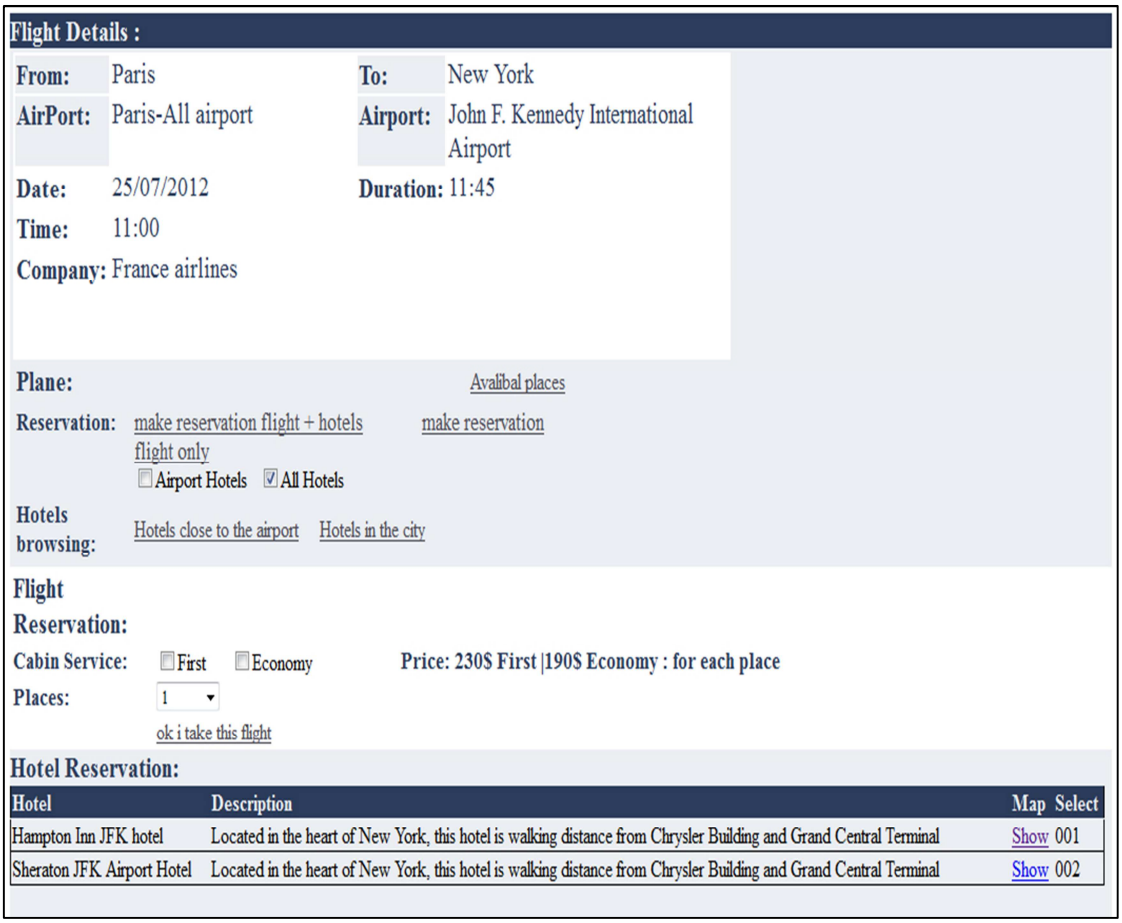


Figure 5.12. Le résultat après l'adaptation

Figure 5.12 montre le résultat de la requête de l'utilisateur adapté au contexte et préférences. Notre processus d'adaptation est assuré par l'adaptation de la requête de l'utilisateur à ses préférences.

Selon les étapes de cette section, nous avons pu montrer l'exploitation de l'ontologie "ContoLogy" pour la gestion des préférences et la résolution des conflits, afin d'adapter la requête initiale de l'utilisateur à son contexte d'utilisation et son profil, qui comprend ses préférences.

CONCLUSION

Dans ce chapitre, nous avons présenté l'implémentation d'un prototype et les performances de notre système.

Dans ce chapitre, nous avons montré en utilisant une étude de cas, comment notre approche adapte la requête de l'utilisateur au contexte, en utilisant l'ontologie de contexte « ContoLogy ». Nous avons essayé d'expliquer comment nous mettons en œuvre les services Web, l'ontologie et montré le déroulement du processus d'adaptation pour résoudre les conflits par un exemple détaillé.

CONCLUSION GÉNÉRALE ET PERSPECTIVES

L'informatique ubiquitaire se concentre sur l'utilisation de deux notions essentielles: profil de l'utilisateur et le contexte d'utilisation afin de mieux satisfaire les demandes des utilisateurs nomades.

De ce fait, la modélisation du contexte est une étape essentielle dans le processus de sensibilité au contexte. Cette modélisation permet à l'application de faciliter l'interaction avec le contexte en fournissant une description abstraite des informations qui constituent ce dernier. La dite modélisation permet d'offrir les fondations pour une représentation expressive du contexte et une simplification de son utilisation.

Par conséquent, une modélisation fiable de ce contexte et une adaptation du comportement des applications au contexte d'utilisation est requise.

En effet, les travaux dans cette thèse portent sur les points suivants :

1. Premièrement, nous avons présenté une nouvelle approche permettant,
    a. D'une part, la modélisation du contexte d'utilisation et les profils de l'utilisateur en utilisant une ontologie, pour soutenir la représentation du contexte et le raisonnement,
    b. D'autre part, la résolution des conflits en utilisant des solutions proposées.
    c. Une architecture illustrant l'adaptation dynamique et fonctionnelle des applications ubiquitaires basées sur les services web est également proposée.
2. Deuxièmement, nous avons détaillé le fonctionnement de l'outil développé et un système de performance. Grâce à cette partie dans cette thèse, nous avons essayé d'expliquer comment nous mettons en œuvre les services Web, l'ontologie et expliquer le processus d'adaptation pour résoudre les conflits par un exemple détaillé.

Comme orientations futures de ce travail, nous projetons de :

1. Compléter l'implémentation du module *context acquisition* composé de deux sous-modules: *context sensor* et *context integration*.
2. Utiliser une approche probabiliste pour représenter les préférences des utilisateurs. C'est un défi très complexe pour représenter les préférences des utilisateurs ainsi que le contexte, en utilisant ces approches, à cause de l'ambiguïté posée par ces applications ubiquitaires. Par exemple: la temporalité, l'incertitude, l'hétérogénéité, le traitement en ligne, et les informations contradictoires. Dans la littérature, plusieurs approches probabilistes (SVM, CPnet, HMM, HHMM, etc) sont étudiées et nous sommes principalement portées sur le modèle de Markov hiérarchique caché (HHMM). Le HHMM est lisible, facile pour la représentation des préférences et ne nécessite pas d'expertise préalable.

3. Le Cloud Computing offre la prochaine génération de base Internet, les systèmes informatiques ubiquitaires hautement évolutifs dont les ressources informatiques sont fournies comme un service. Un nouveau modèle de calcul qui permet un accès facile et un réseau à la demande à un pool partagé de ressources informatiques configurables (réseaux, serveurs, stockage, applications et services) qui peuvent être provisionnés rapidement. Cependant, l'informatique ubiquitaire se réfère à un scénario dans lequel l'informatique est omniprésente, en particulier là où les appareils qui ne ressemblent pas à des ordinateurs ont les capacités de calcul. L'idée est de savoir comment utiliser les ressources de Cloud Computing de manière efficace et de gagner un maximum de profits avec les systèmes ubiquitaires?

ANNEXE

ANNEXE 1 : la notion d'adaptation

1. NOTION D'ADAPTATION

La notion d'adaptation n'est encore définie en termes de consensus. Plusieurs définitions sont données à cette notion dans la littérature, nous citons entre autres, celle donnée par Chaari et al. [Chaari04]. Ces auteurs considèrent l'adaptation comme étant la capacité de définir différents versions d'un service ou distinctes présentations d'un document afin de satisfaire les besoins de l'utilisateur, les contraintes de l'environnement et du dispositif, etc.

Selon le dictionnaire Larousse « Adapter*: rendre (un dispositif, des mesures, etc.) apte à assurer ses fonctions dans des conditions particulières ou nouvelles* » En reprenant la définition sémantique du terme, une adaptation est le processus de modification du système, nécessaire pour permettre un fonctionnement adéquat dans un contexte donné. Le terme adéquat signifie que le système correspond correctement à ce que l'on attend de lui dans un contexte précis.

1.1 Raisons D'adaptation

Après la mise en place d'une application, différents raisons peuvent amener à l'adapter aux différents changements. Ces raisons peuvent être classées en quatre catégories [Ketfi02] :

1. *Adaptation correctionnelle:* cette adaptation peut être appliquée lorsqu'on remarque que l'application en cours d'exécution ne fonctionne pas exactement ou comme envisagé. Alors, dans ce cas de figure, il faut identifier la partie de l'application qui présente le problème et de la remplacer par une nouvelle version qu'on suggère correcte. La nouvelle version de l'application offre les mêmes fonctionnalités que l'ancienne, elle se contente simplement de corriger ses défauts.

2. *Adaptation adaptative:* dans certains cas, même si l'application s'exécute correctement, parfois des changements parviennent au niveau : de son environnement d'exécution, des composants matériels ou d'autres applications ou données dont elle dépend. Alors, dans ce cas, l'application est adaptée en réponse aux changements influant son environnement d'exécution.

3. *Adaptation évolutive :* pendant le processus et au moment de la réalisation on peut négliger quelques fonctionnalités et pour répondre aux besoins évolués de l'utilisateur, l'application doit être étendue avec de nouvelles fonctionnalités.
4. *Adaptation perfective* **:** ce genre d'adaptation a pour objectif, l'amélioration des performances de l'application. A titre d'exemple, on peut avancer que la réalisation d'une partie de l'application que ce soit un composant où un service Web n'est pas améliorée.

1.2 Les Catégories d'Adaptation

On peut distinguer deux types d'adaptation :

1. *l'adaptation fonctionnelle* : elle consiste à adapter le fonctionnement de l'application dans l'objectif de répondre à titre d'exemple à des changements de spécification ou à une évolution des fonctions.
2. *L'adaptation structurelle* : elle consiste à modifier la structure de l'application sans toucher aux fonctions fournies par cette dernière.

En conclusion, dans notre travail, nous nous intéressons à l'adaptation adaptative dynamique et fonctionnelle des applications à base de Services web, en utilisant une architecture de Services web [Ponge04], ce qui revient à adapter le comportement de ces applications lors de l'exécution de la nouvelle situation contextuelle représentée par l'ontologie proposée. La section suivante détaille le concept de service Web.

ANNEXE 2 : notion de Service Web

2. NOTION DE SERVICE WEB

Cette section détaille la notion de service web.

2.1 Définition Des Services Web

Les services Web sont la nouvelle vague des applications Web. Ce sont des applications modulaires, auto-contenues et auto-descriptives qui peuvent être publiées, localisées et invoquées depuis le Web [Ponge04]. Le consortium *W3C*[1] définit un service Web comme étant une application ou un composant logiciel qui vérifie les propriétés suivantes [W3C03]:

1. Il est identifié par un *URI*[2] ;
2. Ses interfaces et ses liens (binding) sont décrits en *XML*[3] ;
3. Sa définition peut être découverte par d'autres services Web;
4. Il peut interagir directement avec d'autres services Web à travers le langage XML et en utilisant des protocoles Internet.

En résumé Un Service Web est un composant implémenté dans n'importe quel langage, déployé sur n'importe quelle plate-forme et enveloppé dans une couche de standards dérivés du XML. Il doit pouvoir être découvert et invoquer dynamiquement par d'autres services.

2.2 Architecture Des Services Web

Selon [Booth03], l'architecture des services Web offre un modèle et un contexte fondamental pour comprendre les services Web et les relations entre les distinctes spécifications et technologies utilisées. L'objectif de l'architecture n'est pas de spécifier la façon dont les services Web sont implémentés ou d'exiger des limitations sur la façon de combiner des services, elle définit plutôt un ensemble de caractéristiques qui doivent être communes aux services Web.

Nous distinguons deux types d'architecture. La première est l'***architecture*** dite de ***référence*** et la deuxième est l'***architecture étendue*** ou encore ***en pile***.

2.2.1 Architecture de référence

D'après [Kreger01, Kadima03], cette architecture se repose sur l'interaction entre trois rôles: le Service registery (*annuaire de service*), le Service provider (*fournisseur de service*)

[1] W3C : *World Wide Web Consortium*, voirhttp://www.w3.org/
[2] URI : *Uniform Resource Identifier*, est une courte chaine de caractères qui identifie les ressources sur le Web: documents, images, fichiers, service, etc.
[3] XML : *eXtensibleMarkupLanguage*.

et le Service requester (*demandeur de service*) (figure A1). Cette interaction se réalise à travers les trois opérations suivantes: la publication de description de services (*publish*), la recherche et la découverte de la bonne description du service (*find*) et l'association ou l'invocation du service basée sur sa description (*bind*).

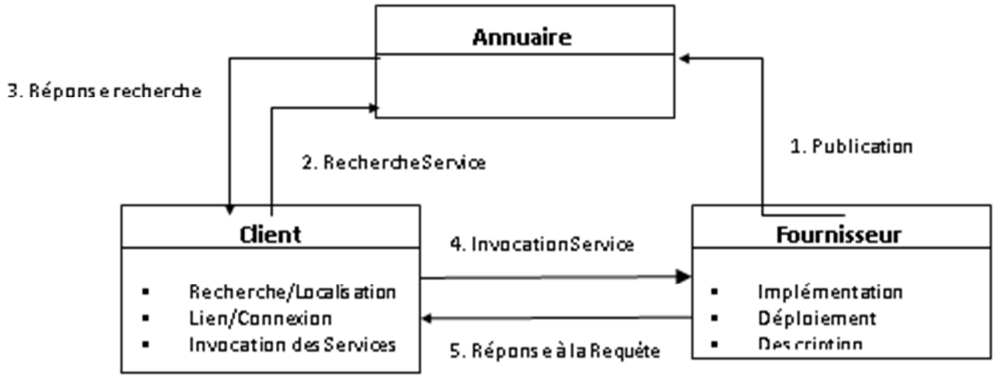


Figure A1 : Architecture de référence des services Web [Kouadri03].

2.2.2 Architecture étendue

Cette architecture est aussi appelée *pile des services Web*, parce qu'elle est formée de plusieurs couches se superposant les unes aux autres [Lopez04].

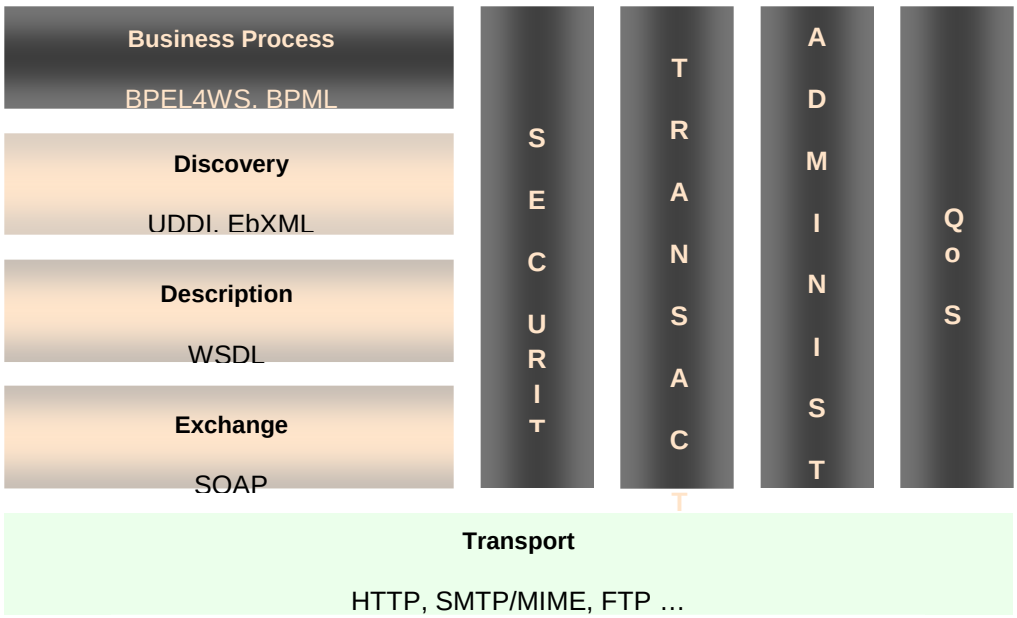


Figure A2 : Architecture en pile des services Web [Lopez04].

Dans cette architecture, et selon [Kellert04], nous pouvons différencier trois types de couches :

1. **L'infrastructure de base** (*discovery*, *description*, *exchange*): elle décrit les

fondements techniques présents dans l'architecture de référence. Pour éviter l'interopérabilité, les différentes couches de la pile des Web services s'interfacent avec des standards [Kadima03]*:* ***SOAP*** pour l'échange de messages, ***WSDL*** pour la description de services et ***UDDI*** ou ***EbXML***[4] pour la publication.

2. **La couche Business Processus** (*Business Process*): elle permet l'intégration de Web services. Cette couche définie un *business process* comme un ensemble de Web services.

3. **Les couches transversales** (*Security*, *Transactions*, *Administration*, *QoS*[5]): ces couches ont pour objectif la durabilité de l'utilisation effective des services dans le monde de l'industrie.

2.3 Technologies Standards

Actuellement, les consortiums et les éditeurs de logiciels sont nombreux, il existe diverse technologies. Le concept des Service Web s'articule actuellement autour des trois acronymes suivants : SOAP, WSDL et UDDI, ils sont basés sur XML.

2.3.1 SOAP (Simple Object Access Protocol)

L'architecture des services Web est distribuée pour permettre des échanges de données entre des applications aux sources multiples, pour cela, les outils d'échange et partagent doivent nécessairement être standardisés. SOAP est un standard recommandé par le W3C qui décrit un protocole assurant les appels de procédures à distance (RPC[6]), en s'appuyant spécialement sur le protocole HTTP et sur XML [Kadima03].Selon [Pasinelli03], SOAP est un protocole léger, qui a pour but l'échange de données structurées dans un environnement distribué et décentralisé. Il est indépendant de tout système d'exploitation ou langage de programmation. SOAP permet de définir un ensemble de règles afin de structurer des messages. Selon [Kadima03, Lopez04], Un message SOAP est un document XML appelé « enveloppe ». L'enveloppe est constituée de deux parties : l'entête (*header)* et le corps du message (*body)*.

2.3.2 WSDL (Web Service Description Language)

WSDL définit l'interface d'utilisation d'un service Web (méthodes et propriétés des

[4]*Electronic Business using eXtensibleMarkup Language*
[5]*Quality of Service*, Ensemble de paramètres échangés pendant une communication avec connexion pour que les informations passent correctement.
[6]*RemoteProcedure Call*, appel de procédures localisées sur des machines distantes.

composants de l'application), ses liaisons de protocoles et ses détails de déploiement [Christophe02]. Il utilise le standard XML pour définir les formats de messages, ce qui désigne que WSDL est indépendant de tout langage de programmation et la plateforme [Christensen01]. Selon [Lopez04, Kadima03], Un document WSDL (élément définition) est composé de sept sous-éléments distincts, divisés en deux parties. Une première partie réutilisable, appelée *abstraite*, détaille le service. Une seconde partie non réutilisable, appelée *Concrète*, expose la localisation du service.

2.3.3 UDDI (Universal description, Discovery and Integration)

UDDI est un protocole d'annuaire (basé sur XML) permettant de trouver le service Web que l'on cherche et de savoir la disponibilité [Christophe02]. Il est à noter que l'UDDI est lui-même un service Web car on y accède en SOAP et il est décrit par l'intermédiaire de WSDL [Daniel03]. Son architecture (voir la figure A3) englobe cinq structures de données suivantes, définies en XML : [Lopez04] :

1. ***BusinessEntity (entité commerciale)***: elle décrit des informations sur l'entreprise qui publie le service, à savoir : son activité, le type de services proposés et l'adresse de son site web.
2. ***BusinessService (offres de services)***: cette couche définit les informations sur le service métier et permet de différencier les services en reconnaissant leurs noms, description, code, etc.
3. ***BindingTemplate (liaisons UDDI)***: cette couche englobe des informations techniques permettant aux clients de se connecter et d'invoquer le service Web.
4. ***tModel (type de services)***: cet élément permet d'associer un service à sa description en WSDL. L'utilisateur peut ainsi avoir connaissance des conventions d'utilisation du service et en avoir une description plus précise.
5. ***PublisherAssertion (affirmation d'éditeur)***: cette partie permet de définir l'organisation dans son intégrité si elle est composée de plusieurs divisions.

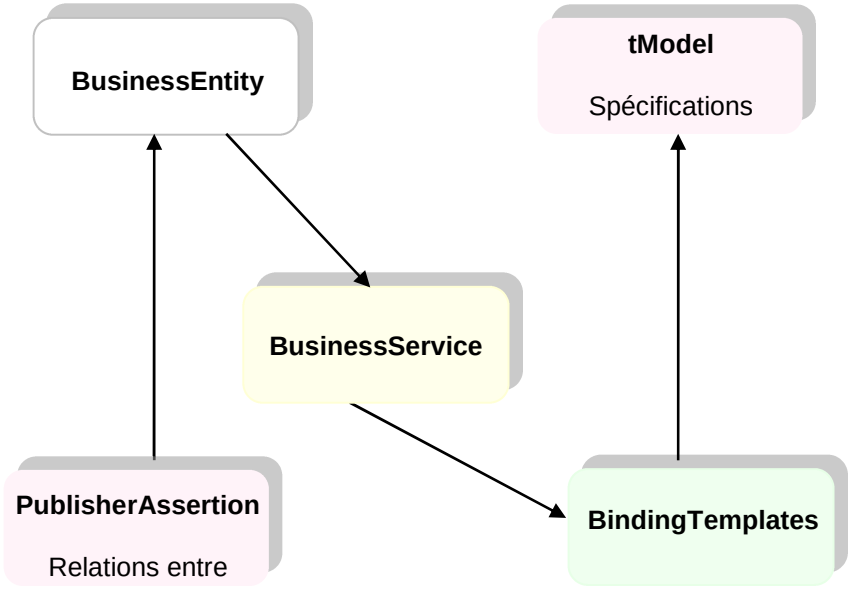


Figure A3 : Entités composant l'annuaire UDDI [Gardarin02].

En résumé, l'annuaire UDDI est un moyen pour qu'une entreprise puisse publier ses informations, son activité et ses Web services. En outre, il lui permet de découvrir les services Web proposés par d'autres.

Références Bibliographiques

[Akman97] Akman.V. , Surav. M., “The use of situation theory in context modelling”, Computational Intelligence 13, 3 (1997), 427–438.

[Baader03a] Baader. F., Nutt. W. , “Basic description logics”. Dans Baader, F., Calvanese, D., McGuinness, D., Nardi, D. et Patel-Schneider, P. (éditeurs), “The Description Logic Handbook : Theory, Implementation and Applications”. Cambridge University Press, pp. 47100. 2003.

[Baader03b] Baader. F., Horrocks. I., Sattler. U., “Description logics as ontology languages for the semantic web”. Dans Hutter, D. et Stephan, W. (éditeurs), “Festschrift in honor of Jörg Siekmann”. Lecture Notes in Artificial Intelligence. Springer-Verlag. 2003.

[Bacon97] Bacon. J., Bates. J. , Halls. D., “Location-oriented multimedia”. IEEE Personal Communications 4, 5 (1997).

[Barwise83] Barwise. J. , Perry. J. , “Situations and Attitudes”. MIT Press, 1983.

[Bauer03] Bauer. J. , ”Identification and Modeling of context for different information in Air Traffic”, Master’s thesis, Université d’électronique et d’informatique de Berlin, Mars 2003.

[Bechhofer01] Bechhofer. S., Horrocks. I., Goble. C., Stevens. R., “OILed : a reason-able Ontology Editor for the semantic Web”, in Proceedings of KI2001, Joint German/Austria Conference on Artificial Intelligence, Springer Verlag LNAI 2174, pages 396-408, 2001.

[Belotti04] 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.

[Belhanafi06] Belhanafi. N., ”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 2006.

[Berthouzoz99] Berthouzoz. C. “A Model of Context Adapted to Domain-Independent Machine Translation”. In CONTEXT ’99 : Proceedings of the Second International and Interdisciplinary Conference on Modeling and Using Context, pages 54–66, London, UK, 1999. Springer-Verlag.

[Booth03] Booth. D. , Haas.h. , McCabe. F., Eric Newcomer, Champion. M., Ferris. C., Orchard. D., “Web Services Architecture”, W3C Working Group Note, 11 February 2003. http://www.w3.org/TR/2004/NOTE-ws-arch-20040211

[Bouzy97] Bouzy. B., Cazenavze. T., " Using the Object Oriented Paradigm to Model Context in Computer Go". In Proceedings of Context'97 (Rio, Brazil, 1997).

[Brézillon99] Brezillon. P. and Pomerol. J., "Contextual knowledge sharing and cooperation in intelligent assistant systems" . In Le Travail Humain, volume 3, pages 223–246, Paris, 1999.

[Brickley03] Brickley. D. and Miller. L. "FOAF vocabulary specification" .In RDFWeb Namespace Document. RDFWeb, xmlns.com, 2003.

[Brown97] Brown. P.J., Bovey. J.D., Chen. X., "Context-aware applications: from the laboratory to the marketpl"ace" , IEEE Personal Communications, vol. 4, n° 5, octobre 1997, pp. 58-64.

[Brown99] Brown. P. J., Davies. N, Smith. M., Steggles. P., "Towards a better understanding of context and context-awareness", In H.-W. Gellerson, editor, Handheld and ubiqitous computing, number 1707 in Lecture Notes in Computer Science, pages 304–307. Springer, September 1999.

[Capra01] Capra. L., Emmerich. W., Mascolo. C., "Reflective Middleware Solutions for Context-Aware Applications", In Proc. of REFLECTION 2001. The Third International Conference on Metalevel Architectures and Separation of Crosscutting Concerns, volume 2192 of lncs, pages 126–133, Kyoto, Japan, September 2001.

[Capra03] Capra. L., Emmerich. W., and Mascolo. C., "CARISMA : Context-Aware Reflective mIddleware System for Mobile Applications", IEEE Transactions on Software Engineering, 29(10):929– 945, oct 2003.

[Carrillo07] Carrillo. R.A., "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 2007.

[Chaari04] Chaari. T., Laforest. F., Celentano. A., " Design of context-aware applications based on web services", Technical Report RR-2004-033, LIRIS, Lyon, octobre 2004. Disponible sur http://liris.cnrs.fr/publis/rr_html (décembre 2004).

[Chaari05] Chaari, T., Laforest, F., Flory, A., " Adaptation des applications au contexte en utilisant les services Web ", In : Coutaz, J., Lecomte, S. (Eds.), Actes des deuxièmes journées francophones : Mobilité et Ubiquité 2005 (UbiMob'05), 31 mai – 3 juin 2005, Grenoble, France, pp. 111-118.

[Chaari06] Chaari. T., Laforest. F., " 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 2006.

[Chahuara13] Chahuara. P., " Contrôle intelligent de la domotique à partir d'informations temporelles multi-sources imprécises ", . Thèse doctorale. s.l., France : Université de Grenoble, 27 mars 2013.

[Chalmers04] Chalmers. D., Dulay. N., and Sloman. M., " Towards Reasoning About Context in the Presence of Uncertainty ", In Proceedings of Workshop on Advanced Context Modelling, Reasoning And Management at UbiComp 2004.

[Chen03] Chen, H., Finin, T., AND Joshi.A., " Using OWL in a Pervasive Computing Broker ", In Proceedings of Workshop on Ontologies in Open Agent Systems (AAMAS 2003) (2003).

[Chen04a] Chen, H., Finin, T., AND Joshi. A., " An Ontology for Context-Aware Pervasive Computing Environments", Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review, 18(3) :197–207, May 2004.

[Chen04b] Chen. H., Perich. F., Chakraborty.D. , Finin. T., and Joshi. A., " Intelligent Agents Meet Semantic Web in a Smart Meeting Room", In Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS 2004), New York City, NY, July 2004.

[Chen04c] Chen. H., Finin. T., and Joshi. A. "A Context Broker for Building Smart Meeting Rooms", In Proceedings of the Knowledge Representation and Ontology for Autonomous Systems Symposium, 2004 AAAI Spring Symposium, pages 53–60, Stanford, California, 2004.

[Chen04d] Chen. H., Perich. F., Finin. T., Joshi. A., " SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications", International Conference on Mobile and Ubiquitous Systems: Networking and Services, Boston, 22-25 August 2004, pp. 258-267.

[Cheverst99] Cheverst. K., Mitchell. K., and Davies. N. "Design of an object model for a context sensitive tourist GUIDE". Computers and Graphics 23, 6 (1999), 883–891.

[Christensen01] Christensen. E., Curbera. R., Meredith.G. , Weerawarana. S., W3C, Mars 2001. http://www.w3.org/TR/2001/NOTE-wsdl-20010315

[Christophe02] Christophe.N., " Définition des services-web " , Université de bourgogne, Octobre 2002. http://www.iai-france.org/projets/batinterop/RT_Nicolle_0210_c.pdf

[Chtcherbina03] Chtcherbina. E., and Franz. M. "Peer-to-peer coordination framework (p2pc): Enabler of mobile ad-hoc networking for medicine, business, and entertainment", In Proceedings of International Conference on Advances in Infrastructure for Electronic Business, Education, Science, Medicine, and Mobile Technologies on the Internet (SSGRR2003w) (L'Aquila/Italy, January 2003).

[Collier03] Collier. R.W., O'Hare G.M.P., Lowen, T., Rooney, C.F.B., " Beyond Prototyping in the Factory of the Agents". In: Marík, V. Müller, J.P., Pechoucek, M. (eds.): Proceedings of the 3rd International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS 2003) (Prague, Czech Republic, June 16-18, 2003), Lecture Notes in Computer Science, vol. 2691, Springer-Verlag, Berlin Heidelberg (2003), pp. 383-393.

[Daniel03] Daniel.J , "Services web Concepts, techniques et outils ", édition Vuibert, avril 2003.

[Dey00] Dey. A.K., "Providing Architectural Support for Building Context-Aware Applications ", PhD Thesis, Georgia Institute of Technology, 2000.

[Dey01a] Dey. A.K., Abowd. G. D., and Salber. D., "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications". Human-Computer Interaction,16 :97–166, 2001

[Dey01b] Dey. A.K. "Understanding and using context ". Personal and Ubiquitous Computing, vol. 5, n° 1, 2001, pp. 4-7.

[Dey01c] Dey. A. K. "Supporting the construction of context-aware applications". In Dagstuhl Seminar on Ubiquitous Computing, September 2001.

[DictionnaireF] Dictionnaire de la langue française du XIXeme et XXeme siècle. Centre national de la recherche scientifique.

[Dourish04] Dourish. P., "What we talk about when we talk about context" , Personal and Ubiquitous Computing, vol. 8, n° 1, 2004, pp. 19-30.

[Doulkeridis06] Doulkeridis.C, Loutas.N, Vazirgiannis.M. "Asystem architecture for context aware service discovery". J Electron NotesThe oreticComputSci, pp 101–116. 2006

[Driouche07] Driouche.R.. "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. 2007. université Mentouri de Constantine. faculté des sciences de l'ingénieur. département d'informatique.

[Espinasse09] Espinasse. Bernard. " Introduction au langage Ontology Web Language (OWL) ", Professeur à l'Université d'Aix-Marseille. 31 mars 2009.

[Evren-Sirin06] Evren. Sirin. , Bijan P., Bernardo. C. G., Aditya K., and Yarden. K., Pellet, "A practical owl-dl reasoned". Submitted for publication to Journal of Web Semantics, 2006.

[Fernandez97] Fernandez. M., Gomez-Perez. A., Juristo. N., "METHONTOLOGY : from ontological art towards ontological engineering", in Proceedings of the Spring Symposium Series on Ontological Engineering (AAAI'97), AAAI Press , 1997.

[Farrar10] Farrar. S. , Langendoen. D.T., “An owl-dl implementation of gold- an ontology for the semantic web”, Journal of Linguistic modeling of Information and Markup Languages 40:45–66. 2010.

[FÜRST02] frédéric. fürst, “l’ingénierie ontologique”. rapport de recherche no 02-07 octobre 2002.

[Gandon04] Gandon. F., Sadeh. N., “Semantic Web Technologies to Reconcile Privacy and Context Awareness”. Journal of Web Semantics [En ligne]. vol. 1, no. 3. (October 31, 2004). Disponible sur : http://www.websemanticsjournal.org/ps/pub/2004-17

[Gardarin02] Gardarin.G.. XML: Des bases de données aux services Web, Dunod, 2002, pp. 236-296.

[Giunchiglia93] Giunchiglia. F., “Contextual reasoning. Epistemologica” - Special Issue on I Linguaggi e leMacchine 16 (1993), 345–364. Also IRST-Technical Report 9211-20, IRST, Trento, Italy.

[Giunchiglia01] Ghidini. C., And Giunchiglia. F., “Local models semantics, or contextual reasoning= _locality+compatibility.”, Artificial Intelligence 127, 2 (2001), 221–259.

[Gómez-Pérez04] Gomez-Perez. A., Fernandez. M., De Vicente A. J., “Towards a Method to Conceptualize Domain Ontologies”, in Proceedings of the European Conference on Artificial Intelligence, ECAI’96, pages 41-52, 1996.

[Goslar04] Goslar. K, Schill. A., “modelling contextual information using active data structures”. In: Proceedings of the EDBT workshops.Lecture notes in computer science, vol 3268. Springer,2004.

[Gray01] Gray. P. And Salber. D. “Modelling and Using Sensed Context Information in the design of Interactive Applications”. In LNCS 2254: Proceedings of 8th IFIP International Conference on Engineering for Human-Computer Interaction (EHCI 2001) (Toronto/Canada, May 2001), M. R. Little and L. Nigay, Eds., Lecture Notes in Computer Science (LNCS), Springer, p. 317.

[Gruber93] Gruber. T. G. “A translation approach to portable ontologies”, Knowledge Acquisition 5, 2 (1993), 199–220.

[Gruber94] Gruber. T., Olsen. G., “An ontology for enginering mathematics”, in J. DOYLE F. S.&TORANO P., eds., Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasonning, Morgan-Kauffmann, 1994.

[Gruber95] Gruber T.R., “Toward principles for the design of ontologies used for knowledge sharing”. International Journal of Human Computer Studies. 1995.

[Grudin01] Grudin. J., « Desituating action: digital representation of contexte », Human-Computing Interaction, vol. 16, n° 2-4, 2001, pp. 269-286.

[Gruninger95] Gruninger. M., Fox. M. S., “Methodology for the design and evaluation of ontologies”, in Proceedings of the Workshop on Basic Ontological Issues on Knowledge Sharing, IJCAI’95, 1995.

[Gu04] Gu, T. et al. “An ontology-based context model in intelligent environments”. Proceedings of Communication Networks and Distributed Systems Modelling and Simulation Conference, San Diego (CA), USA. 2004.

[Guarino98] Guarino. N., “Formal Ontology in Information Systems”. Proceedings of FOIS’98, Trento, Italy, 6-8 June 1998. Amsterdam, IOS Press, pp. 3-15.

[Haarslev01] Haarslev. V. and M¨oller.R. “Racer user’s guide and reference manual version 1.6”. Technical report, University of Hamburg, Computer Science Department, 2001.

[Halpin01] Halpin. T. A. “Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design”. Morgan Kaufman Publishers, San Francisco, 2001.

[Hayes79] Hayes. P. J, “The logic of frames”. In D. Metzing (Ed). Frame Conceptions and Text Understanding. Walter de Gruyter & Co., 1979.

[Held02] Held. A, Buchholz. S., and Schill.A., “A Modeling of Context Information”, for Pervasive Computing Applications. 2002.

[Henricksen04] Henricksen. K. and Indulska. J., “Modelling and Using Imperfect Context Information”, In PerCom Workshops, pages 33–37, 2004.

[Hristova04] Hristova. N., O'Hare, G., “Ad-me: wireless advertising adapted to the user location, device and emotions” . In: Proceedings of 37th Annual Hawaii International Conference on System Sciences (HICSS37), Minitrack on Mobile Distributed Information Systems (MDIS) (Hawaii, Janvier 5–8, 2004), part of the Software Technology Track, IEEE Computer Society Press (2004), pp. 1-10.

[Ian Horrocks01] Ian Horrocks, Peter Patel-Schneider, Tim Berners-Lee ,Dan Brickley Dan Connolly Mike Dean Stefan Decker Dieter Fensel Richard Fikes Pat Hayes Jeff Heflin Jim Hendler Ora Lassila Deb McGuinness Lynn Andrea Stein, Frank van Harmelen. Specification DAML+OIL, 2001. http://www.daml.org/ 2001/03/daml+oil-index.html.

[Horrocks03] Horrocks. I., P. F. Patel-Schneider, et F. van Harmelen, “From shiq and rdf to owl: The making of a web ontology language”. J. of Web Semantics 1 (1), 726. 2003.

[Horrocks04] Horrocks. Ian, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. SWRL: “A Semantic Web Rule Language Combining OWL and RuleML”. W3C Member Submission, 21 May 2004. Available athttp://www.w3.org/Submission/SWRL/.

[Indulska03] Indulska. J., Robinsona.R., Rakotonirainy. A., And Henricksen. K. “Experiences in using cc/pp in context-aware systems”. In LNCS 2574:Proceedings of the 4th International Conference on Mobile Data Management (MDM2003) (Melbourne/Australia, January 2003), M.-S. Chen, P. K. Chrysanthis, M. Sloman, and A. Zaslavsky, Eds., Lecture Notes in Computer Science (LNCS), Springer, pp. 247–261.

[Jiang04] Jiang. X., Chen. N. Y., Hong. J. I, Wang. K., L. Takayama, and J. A. Landay. Siren : contextaware Computing for Firefighting. In Pervasive, pages 87–105, 2004.

[Kadima03] Kadima. H. & Montfort.V., “Les services Web: Techniques, démarches et outils XML, WSDL, SOAP, UDDI, Rosetta, UML“, Dunod, 2003.

[Kapitsaki09] Kapitsaki. G., Kateros. D., Prezerakos. G., Venierris. I., “Model driven development of composite context-aware web applications”. J InformSoftwTechnol 51:1244–1260.2009.

[Kanellopoulos08a] Kanellopoulos. D., "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. 194-205.2008.

[Kanellopoulos08b] Kanellopoulos. D., Panagopoulos A. "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. 2008.

[Kanellopoulos09] Kanellopoulos. D., "Adaptive multimedia systems based on intelligent context management", International Journal of Adaptive and Innovative Systems, Vol. 1, No.1, pp.30-43.2009.

[Kassel00] Kassel.G., Abel. M., Barry. C., Boulitreau.P., Irastorza. C., Perpette. S., “Construction et exploitation d’une ontologie pour la gestion des connaissances d’une équipe de recherche“, in Actes des journées francophones d’Ingénierie des Connaissances (IC’2000), 2000.

[Kellert04] Kellert. P. & Toumani. F., Les web services sémantiques, Rapport final d’activité spécifique CNRS/STIC, 2004.

[Ketfi02] Ketfi. A., Belkhatir. N., “P-Y Cunin Adaptation Dynamique Concepts et Expérimentations“ ,,ICSSEA 2002.

[Keita07] Keita. A., “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, 2007.

[Kifer89] Kifer. M. and Lausen. G., “ F-logic : a higher-order language for reasoning about objects, inheritance”, and scheme. In SIGMOD ’89 : Proceedings of the 1989 ACM SIGMOD international conference on Management of data, pages 134–146, New York, NY, USA, 1989. ACM Press.

[Kirsch06] Kirsch-Pinheiro. M. "Adaptation Contextuelle et Personnalisée de l’information de Conscience de Groupe au sein des Systèmes d’Information Coopératifs“ . Thèse de doctorat, université Joseph Fourier, Grenoble, 29 Septembre 2006.

[Kirsh01] Kirsh D., « The context of work », Human Computer Interaction, vol. 16, n° 2-4, 2001, pp. 305-322.

[Klyne04] Klyne. G., Reynolds. F., C. Woodrow, H. Ohto, J. Hjelm, M. H. Butler, and L. Tran. “Composite Capability/Preference Profile (CC/PP) : Structure and vocabularies 1.0”. Technical report, W3C recommadation, 15 January 2004.

[Koch04] Koch, F., Rahwan, I. “Classification of Agent-based Mobile Assistant”. In: Proceedings of the Workshop on Agents for Ubiquitous Computing (UbiAgents04) (Columbia University, New York City, USA July 20, 2004) in conjunction with AAMAS2004 [En ligne]. Disponible sur: http://www.ift.ulaval.ca/~mellouli/ubiagents04/ (Consulté en Septembre 2006).

[Kosala13] Kosala.Y, MingXue.W, Claus. P., “ 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. 2013.

[Kouadri03] Kouadri. Mostefaoui. S. , Hirsbrunner. B., “ vers une approche orientée contexte pour la découverte et la composition des services dans des environnements mobiles “, Octobre 2003.

[Kreger01] Kreger. H.. Web services conceptual architecture (WSCA 1.0), mai 2001, IBM Software Group. http://www 06.ibm.com/software/solutions/webservices/WSCA.pdf.

[Larousse92] « Larousse de Poche », Larousse, 1992.

[Lekhchine09] Lekhchine R., “ Construction d’une ontologie pour le domaine de la sécurité : Application aux agents mobiles“. Mémoire Pour l’obtention du diplôme de Magister en Informatique. Université Mentouri – Constantine Faculté des Sciences de l’Ingénieur – Département d’Informatique . 2009.

[Lemlouma04] Lemlouma T., « Architecture de négociation et d'adaptation de Services Multimédia dans des Environnements Hétérogènes », Thèse de Doctorat, Institut National Polytechnique de Grenoble, Grenoble, France, avril 2004.

[Lopez04] Lopez. C. “ Services Web et adaptabilité à l’utilisateur “, Master Mathématiques Informatique, 2eme année Recherche, Spécialité Systèmes d’Informations, Soutenu le 28 juin 2004.

[Maass97] Maass. H., “Location-aware mobile applications based on directory services”. In 3rd annual ACM/IEEE international conference on Mobile computing and networking, pages 23–33, New York, NY, USA, 1997. ACM Press.

[Manola04] Manola. F. and Miller. E.. “RDF Primer. Technical report”, W3C recommandation, 10 February 2004.

[McCarthy93] McCarthy. J. “Notes on Formalizing Contexts”. In T. Kehler and S. Rosenschein, editors, Proceedings of the Fifth National Conference on Artificial Intelligence, pages 555–560, Los Altos, California, 1993. Morgan Kaufmann.

[McCarthy94] McCarthy.J. and Buvac. S. “Formalizing Context (Expanded Notes)”. Technical report, Stanford, CA, USA, 1994.

[Medjahed07] Medjahed. B., Atif Y., “Context-based matching for web service composition”, J Distrib Parallel Databases 21:5–37.2007.

[Menad08] Menad. S., Hadj kaddour. K., “Conception et Réalisation D’une Application De Gestion D’une Agence De Réservation De Voyages Par Les Services Web “.Mémoire De Fin D’étude pour l’obtention du Diplôme d’Ingénieur d’Etat en Informatique de l’université Djilali Liabes – Sidi Bel Abbes. 2008.

[Miao13] Miao.L.V., Chun.JIN, Yoshiyuki.H., Jim. C., “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. 2013.

[Minsky75]. Minsky. M., “A framework for representing knowledge”. The Psychology of Computer Vision. McGraw-Hill. 1975.

[Minsky81] Minsky .M., “A framework for representing knowledge”. Dans Haugeland, J. (éditeur), Mind Design. The MIT Press, pp. 95128. 1981.

[Mostéfaoui04] Mostéfaoui. K., Pasquier-Rocha. J., Brézillon, P., “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.

[Nardi03]. Nardi. D., Brachman. R. J., “An introduction to description logics”. Dans Baader, F., Calvanese, D., McGuinness, D., Nardi, D. et Patel-Schneider, P. (éditeurs), “The Description Logic Handbook : Theory, Implementation and Applications”. Cambridge University Press, pp. 544. 2003.

[Noy00] Noy. N., Fergerson. R. W. and Musen. M. A., “The knowledge model of Protégé2000 : combining interoperability and flexibility”, in Proceedings of the International Conference on Knowledge Engineering and Knowledge Management (EKAW’00), 2000.

[Noy01] Natalya. F. Noy and Deborah L. McGuinness, “Ontology Development 101: A Guide to Creating Your First Ontology”, Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, March 2001.

[O’Hare02] O’Hare. G., O’Grady. M. “Addressing Mobile HCI Need through Agents”, In: Paterno, F. (ed.): Proceedings of the 4th International Symposium on Human Computer Interaction with Mobile Devices and Services (MobileHCI’02) (Pisa, Italy, September 18-20, 2002), Lecture Notes in Computer Science, Springer Verlag, Berlin Heidelberg, vol. 2411 (2002), pp. 311-314.

[Otzturk97] Otzturk. P., and Aamodt. A., ”Towards a model of context for case-based diagnostic problem solving”, In Context-97; Proceedings of the interdisciplinary conference on modeling and using context (Rio de Janeiro, February 1997), pp. 198–208.

[Pagels06] Pagels. M., “The DARPA Agent Markup Language”, available at :http ://www.daml.org/, feb 2006.

[Pan04] Pan. F., and Hobbs. J. R.; “Time in OWL-S. In Proceedings of AAAI-04 Spring Symposium on SemanticWeb Services”, Stanford University, California, 2004.

[Pascoe97] Pascoe. J. “The stick-e note architecture : extending the interface beyond the user”, In IUI ’97 : Proceedings of the 2nd international conference on Intelligent user interfaces, pages 261–264, New York, NY, USA, 1997. ACM Press.

[Pasinelli03] Pasinelli. P., “Seminar-Service Interoperability on Context Levl”, in Ubiquitous Computing Environments, University of Fribourg, Switzerland ,2003.

[Pittarello05] Pittarello. F. , ” 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.

[Pitoura94] Pitoura.E. and Bhargava. B., “Building information systems for mobile environments”, In CIKM ’94 : Proceedings of the third international conference on Information and knowledge management, pages 371–378, New York, NY, USA, 1994. ACM Press.

[Ponge04] Ponge .J., “comptabilité et substitution dynamique des web services”, mémoire de fin d’études, université Blaise Pascal Clermont II, juillet 2004.

[Powers03] Powers. S. “Practical RDF”, O’Reilly & Associates, 2003.

[Quillian68]. Quillian,M., “Semantic memory”. Semantic Information Processing. MIT Press, Cam. 1968

[Rahwan04] Rahwan, T., Rahwan, I., Ashri, R. “Agent-Based Support for Mobile Users Using AgentSpeak (L)”, In: Giorgini P., Henderson-Sellers B., Winikoff, M. (eds.): Proceedings of the Workshop on Agent-Oriented Information Systems (AOIS 2003) (Melbourne, Australia, July 14, 2003 - Chicago, USA, October 13, 2003), Lecture Notes in Artificial Intelligence, vol. 3030 Springer-Verlag, Berlin Heidelberg (2004), pp. 45-60.

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

[Rey04] Rey. G., Coutaz. J., “Le contexteur : capture et distribution dynamique d’information contextuelle“, Mobilité & Ubiquité’04 (UbiMob’04), Nice, France, 2004. pp. 131-138.

[Ryan06] Ryan. N., “ConteXtML : Exchanging contextual information between a mobile client and the fieldnote server”, http ://www.cs.kent.ac.uk/projects/mobicomp/fnc/ConteXtML.html, Accessed in 2006.

[Samulowitz01] Samulowitz. M., Michahelles. F., and Linnhoff-Popien. C., “ CAPEUS : An Architecture for Context-Aware Selection and Execution of Services”, In Proceedings of the IFIP TC6 / WG6.1 Third International Working Conference on New Developments in Distributed Applications and Interoperable Systems, pages 23–40, Deventer, The Netherlands, The Netherlands, 2001. Kluwer, B.V.

[Sattler03] Sattler. U., Calvanese .D., and Molitor. R., “Relationships with other formalisms”. Dans Baader, F., Calvanese, D., McGuinness, D., Nardi, D. et Patel-Schneider, P. (éditeurs), “The Description Logic Handbook: Theory, Implementation and Applications”. Cambridge University Press, pp. 142183. 2003

[Schilit93] Schilit. B., Theimer .M., and Welch. B., “Customizing Mobile Application”, In USENIX Symposium on Mobile and Location-independent Computing, pages 129–138, Cambridge, MA, US, 1993.

[Schilit94a] Schilit. B. , Adams. N, and Want. R. “Context-aware computing applications”, In IEEE Workshop on Mobile Computing Systems and Applications (Santa Cruz, CA, US, 1994

[Schilit94b] Schilit. B.N., Theimer. M.M., “ Disseminating active map information to mobile hosts”, IEEE Network, vol. 8, n°5, septembre/octobre 1994, pp. 22-32.

[Schilit94c] Schilit. B.N., Adams. N., Want. R., “Context-Aware Computing Applications “, Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, USA, December 1994, pp. 85-90.

[Schmidt99a] Schmidt. A., Beigl. M., and Gellersen. H.W. “There is more to context than location”, Computers and Graphics 23, 6 (1999), 893–901.

[Schmidt99b] Schmidt. A., Aidoo. K. A., Takaluoma. A., U. Tuomela, K. V. Laerhoven, and W. V. de Velde, “Advanced Interaction in Context”, In HUC ’99 : Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing, pages 89–101, London, UK, 1999. Springer-Verlag

[Schmidt01] Schmidt. A., and Laerhoven. K. V., “How to Build Smart Appliances”, IEEE Personal Communications (August 2001).

[Sintek02] Sintek.M. and Decker. S., “TRIPLE - A Query, Inference, and Transformation Language for the Semantic Web”, In ISWC ’02 : Proceedings of the First International Semantic Web Conference on The Semantic Web, pages 364–378, London, UK, 2002. Springer-Verlag.

[Soukkarieh10] Soukkarieh. B., “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 2010.

[Sowa84] Sowa.j., “Conceptual Structures: Information Processing in Mind and Machine”, Addison-Wesley.A , Reading, MA, 1984.

[Swartout97] Swartout. B., Ramesh. P., Knight. K. and Russ. T., “Toward Distributed Use of Large-Scale Ontologies”, In Symposium on Ontological Engineering of AAAI, Stanford, California, March, 1997.

[Staab01] Staab. S., Maedche. A., “Axioms are objects too: Ontology engineering beyond the modeling of concepts and relations”, Research report 399, Institute AIFB, Karlsruhe, 2000.

[Strang03a] Strang.T., “Service Interoperability in Ubiquitous Computing Environments”, PhD thesis, Ludwig-Maximilians-University Munich, Oct. 2003.

[Strang03b] Strang. T., Linnhoff-Popien. C., and Frank. K., “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 2893 of Lecture Notes in Computer Science (LNCS), pages 236–247, Paris/France, November 2003. Springer Verlag.

[Strang03c] Strang. T., Lihoff-popien. C., and Roeckl. M., “Highlevel Service Handover through a Contextual Framework”, In Proceedings of 8th International Workshop on Mobile Multimedia Communications (MoMuC2003) (Munich/Germany, October 2003), J. Kaefer and M. Zuendt, Eds., vol. 1, Center for Digital Technology and Management (CDTM), pp. 405–410.

[Strang03d] Strang, T., Lihoff-popien. C., and Frank. K. “Applications of a Context Ontology Language”, In Proceedings of International Conference on Software, Telecommunications and Computer Networks (SoftCom2003) (Split/Croatia, Venice/Italy, Ancona/Italy, Dubrovnik/Croatia, October 2003), D. Begusic and N. Rozic, Eds., Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture, University of Split, Croatia, pp. 14–18.

[Strang04] Strang, T., Lihoff-popien. C., “A Context Modeling Survey”, In Workshop on Advanced Context Modelling, Reasoning and Management as part of UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/England, September 2004.

[Sure02] Sure.Y., Erdmann. M., Angele. J., Staab. S., Studer. R. and Wenke. D., “OntoEdit : collaborative ontology development for the semantic web”, in Proceedings of the International Semantic Web Conference, Springer-Verlag LNCS 2342, pages 221-235, 2002.

[studer98] Studer. R., Benjamins. R. et Fensel D., “Knowledge Engineering: Principles and Methods”. Data Knowledge Engineering.1998.

[TEA98] Esprit project 26900: Technology for enabled awareness (TEA, 1998).

[Troncy04] Troncy. B., Charlet. J. R. , “Ontologies pour le Web Sémantique“. Action spécifique 32 CNRS / STIC Web sémantique Rapport final. 2004.

[Uschold95] Uschold. M. and King. M., “Towards a methodology for building ontologies”, in Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI’95, 1995.

[Uschold96] Uschold. M. and Grüninger. M., “Ontologies : principales, methods, and applications”. Knowledge Engineering Review, 11(2) :93–155, 1996.

[Volk98] Volk. M. “The Automatic Translation of Idioms. Machine Translation vs. Translation Memory Systems”. In: Nico Weber(Ed): Machine Translation: Theory, Application, and Evaluation. An assessment of the state of the art. St.Augustin : gardez-Verlag. 1998.

[Wang04a] Wang. X. H., Zhang. D. Q., GU, T., and Pung.H. K., “Ontology Based Context Modeling and Reasoning using OWL”, In Workshop Proceedings of the 2nd IEEE Conference on Pervasive Computing and Communications (PerCom2004) (Orlando, FL, USA, March 2004), pp. 18–22.

[Wang04b] Wang. X. H., Zhang. D. Q., GU, T., and Pung.H. K., “Ontology Based Context Modeling and Reasoning using OWL“, In PerCom Workshops, pages 18–22, 2004

[Wang04c] Wang. X. H., Zhang. D. Q., GU, T., and Pung.H. K., “An Ontology Based Context Model in intelligent Environment”, In Communication networks and Distributed System modeling ans simulation Conference (CNDS 2004), pages 270–275, San Diego, Clalifornia, USA, January 2004.
WAPFORUM. User Agent Profile (UAProf).

[WAPF03] http://www.wapforum.org.

[Weiser91] Weiser. M. “The computer for the 21st century“. In Scientific American, tome 265, n° 3, pages 66-75, septembre 1991.

[Weiser93] Weiser, M., “Some computer science issues in ubiquitous computing”, Communication of the ACM, vol. 36, n° 7, juillet 1993, ACM Press, pp. 75-84.

[Welty01] Welty. C. and Guarino. N., “Supporting ontological analysis of taxonomic relationships”, Data et Knowledge Engineering (39), pages 51-74, 2001.

[W3C03] World Wide Web Consortium. Web Services Glossary, W3C Working Draft, 14 May 2003.http://www.w3.org/TR/2003/WD-ws-gloss-20030514/#webservice.

[W3C04a] Recommendation W3C OWL, 2004. http://www.w3.org/TR/owl-ref/.

[W3C04b] W3C. OWL Web Ontology Language Use Cases and Requirements. http ://www.w3.org/TR/webont-req/, W3C Recommendation 10 February 2004.

[W3C04c] Recommandation W3C RDFS, 2004. http://www.w3.org/TR/2004/REC-rdf-schema-20040210/. 10 February 2004

[W3C04d] Recommandation W3C XML, 2004. http://www.w3.org/TR/2004/REC-xml-20040204. Le 04 February 2004.

[W3C04e] Recommendation W3C XML-S .2004. http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/. 28 October 2004.

[W3C04f] Recommendation W3C RDQL .2004. http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/. 9 January 2004

[W3C05] OWL: http://www.w3.org/TR/webont-req

[W3C06] Recommendation W3C SPARQL .2006. http://www.w3.org/TR/2006/CR-rdf-sparql-query-20060406/. 6 April 2006

[Yau01] Yau. S. S. and Karim. F., "Context-Sensitive Middleware for Real-Time Software in Ubiquitous Computing Environments", In 4th International Symposium on Object-Oriented Real Time Distributed Computing, pages 163–170, 2001.

[Yau04] Yau. S. S. and Karim. F. "An Adaptive Middleware for Context-Sensitive Communications for Real-Time Applications in Ubiquitous Computing Environments", Real-Time Systems, pages 29–61, 2004.

[Zacarias05] Zacarias, M., Caetano, A., Pinto, S., and Tribolet, J., "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.

[Zaidenberg10] Zaidenberg. S., " Reinforcement learning of context model for ubiquitous computing". In Pulsar research meeting. January 17, 2010.

[Zhou Mingtian03] Zhou Mingtian and Zuo Zhihong. "Web ontology language owl and its description logic foundation", pages 157 – 160. Parallel and Distributed Computing, Applications and Technologies, PDCAT'2003., 2003.

[Zou04] Zou. Y., Finin. T., et Chen. H., "F-owl : an inference engine for semantic web". Dans Third NASAGoddard/IEEE Workshop on Formal Approaches to Agent-Based Systems. Greenbelt, Maryland. 2004.

[Zweigenbaum99] Zweigenbaum. P., "Encoder l'information médicale : des terminologies aux systèmes de représentation des connaissances"., in Innovation stratégique en information de santé (ISIS) (2-3), pages 27-47, 1999.

ملخص

في العقد الماضي، أصبحت الحوسبة في كل مكان (UC) طموح لمجتمع تكنولوجيا المعلومات. اليوم هي مهمة جدا، حيث أنه من المستحيل حاليا عدم وجودها على جدول الأعمال العالمي لأبحاث في الكمبيوتر. في الحوسبة في كل مكان ، الهدف الرئيسي هو مساعدة المستخدمين على الوصول إلى الخدمات والموارد في أي وقت، في أي مكان، وخاصة باستخدام الأجهزة النقالة (MD). التطبيقات في هذا المجال هي حساسة لسياق المستخدم. و يجب أن يكونوا قادرين على إدراك هذا السياق، والتكيف مع سياق الاستخدام وتفضيلات المستخدم. في الواقع، الوصول الآمن لنظم المعلومات، لمستخدمي الهواتف المتحركة من خلال مختلف الأجهزة واستجابات التكيف لسياق مستخدم المحمول وسياق الاستخدام، هما إشكاليتين مرتبطتين. نحن نحاول في هذه الأطروحة معالجة هذه القضايا بإقتراح نهج جديد تماما، يسمح ب : (1) تمثيل السياق وتفضيلات مستخدمي الهواتف المتحركة من خلال الأنطولوجيا (2) حل النزاعات التي قد تنشأ بين تفضيلات المستخدم و، (3) تطويع هذه التطبيقات إلى سياق الإستخدام وتعريف المستخدم. ويدعم هذا النهج برمجة التي قمنا بتطويرها. وتقدم دراسة حالة لإعطاء مزيد من التوضيحات.

الكلمات المفاتيحية: سياق الاستخدام؛ نبذة عن المستخدم.نظام المعلومات على شبكة الانترنت. البيئة البدوية. الأنطولوجيا. تفضيلات المستخدم.

الصراعات.الحوسبة في كل مكان. خدمة ويب.

ABSTRACT

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 thesis, 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.

KEYWORDS : Context of use; User profile; Web-based Information System; Nomadic Environment; Ontology; User Preferences; conflicts; ubiquitous computing; web service.

Résumé

Dans la dernière décennie, l'informatique ubiquitaire (IU) est devenue une aspiration de la communauté informatique. Aujourd'hui, sa vulgarisation est importante qu'il est de plus en plus impossible de la distinguer de l'ordre du jour global de la recherche informatique. Dans (IU), l'objectif principal est de fournir aux utilisateurs la possibilité d'accéder à des services et des ressources à tout moment, n'importe où, en particulier en utilisant des appareils mobiles. Dans ce domaine, les applications sont sensibles au contexte. Elles doivent être capables de percevoir ce contexte et à adapter leurs comportements en tenant compte des données qui traitent le contexte d'utilisation et les préférences des utilisateurs. En fait, assurer l'accès des utilisateurs nomades aux systèmes d'information grâce à divers dispositifs, et l'adaptation des réponses aux profils et aux contextes d'utilisation sont deux problèmes inséparables. Dans cette thèse, nous tentons de répondre à ces problèmes et nous proposons une nouvelle approche permettant essentiellement: (1) la représentation du contexte et des préférences des utilisateurs nomades grâce à une ontologie, pour soutenir la représentation du contexte et le raisonnement (2) la résolution des conflits qui peuvent surgir entre les préférences de l'utilisateur et, (3) l'adaptation de ces applications au contexte d'utilisation et le profil de l'utilisateur. L'approche est soutenue par un outil visuel, que nous avons développé. Une étude de cas est présentée pour donner plus d'illustrations.

Mots-clés: Contexte d'utilisation; Profil de l'utilisateur; Système d'information sur le Web; Environnement nomade; ontologie; Préférences de l'utilisateur; conflits; informatique ubiquitaire; service Web.

Publications

Publications internationales:

1. Salima Bourougaa-Tria, Hassina Seridi-Bouchelaghem, Farid Mokhati. "**An Ontology-Based Context Model to Manage Users Preferences And Conflicts"**. Informatica. An International Journal Of Computing And Informatics Volume 40, N°1 (2016):

Conférences internationales

1. Salima Bourougaa-Tria, Hassina Seridi-Bouchelaghem, Farid Mokhati**. "Management of user preferences and conflicts to ubiquitous applications adaptation** ". ICEIS 2012 – Fourteen International Conference on Enterprise Information Systems . Wroclaw, Poland; 28 Juin - 1 Juillet 2012.

2. Salima Bourougaa-Tria. "**Semantics profiles of users for adapting ubiquitous information systems".** 2nd International Conference on Information Systems and Technologies 2012 (ICIST'2012) March 24-26 Sousse, Tunisia

3. Salima Bourougaa-Tria, Hassina Seridi-Bouchelaghem, Farid Mokhati. "**An approach based on semantic management of user profile to ubiquitous applications adaptation.** The Third International Conference on Digital Information Processing and Communications (ICDIPC2013), Dubai, U.A.E; Jan. 30 - Feb. 01, 2013.

4. Salima Bourougaa-Tria, Boudabous Soufiane. "**Toward using HHMM for representation user's preferences in Context-Aware Systems"**. ICSENT 2014 The 3rd International Conference on Software Engineering and New Technologies, Décembre, 20 – 22 2014 hammamet, Tunisia.

5. Salima Bourougaa-Tria, Djeddi soumaya. "**Toward using the cloud computing in ubiquitous systems**". ICIST2015, 5nd International Conference on Information Systems and Technologies, March 21-23 2015 Istanbul, turkey

Conférences nationales

1. **JOMI 2014 - Journées Ouvertes e mathématiques et Informatique, 11,12 mai 2014, Tébessa, algerie.**

BIOGRAPHIE DE L'AUTEUR

Dr. Salima Bourougaa-Tria. Est née en 1972 à Skikda, Algérie. Elle est actuellement Maitre de conférence « B» en Informatique à l'université de Tébessa (Algérie). Elle a obtenu son Baccalauréat, option (Maths) en 1989 du Lycée Fatma Zohra, Tébessa, Algérie, avec la mention très bien. En 1994, elle a obtenu un diplôme d'Ingénieur d'état en informatique, option : systèmes d'informations de la faculté des sciences et des technologies, université de Annaba sujet : Conception et réalisation d'un système d'information pour la gestion d'hôtellerie au niveau d'EGTA d' Annaba. Du 1994 au 2009, elle a occupé plusieurs postes, notamment : Professeur spécialisé d'enseignement professionnel (PSEP2), Inspectrice de la formation et de l'enseignement Professionnel (IFEP).

En 2009, elle a soutenu son mémoire de Magister en informatique, Option : Systèmes d'informations et de connaissances, Intitulé « Représentation des informations contextuelles pour l'adaptation des systèmes d'information ubiquitaires », avec la mention très bien (17.5/20) de Université de Cheikh Larbi Tbessi, Tébessa, Algérie, Faculté des Sciences et des Technologies, et elle s'est intégré à l'université comme Maitre-Assistant « B ». Elle a poursuivi ses études de post-graduation pour préparer une thèse de doctorat d'état sur les Profils Sémantique des utilisateurs pour l'adaptation des services, applications aux SI ubiquitaires. La thèse est soutenue le 18/07/2016. Ses activités de recherche s'intéressent au : domaine des systèmes d'information : conception, modélisation, domaines des bases de données, représentation et modélisation sémantiques : utilisation des ontologies, domaine des SI ubiquitaire et pevasifs. Elle a participé à plusieurs conférences spécialisées nationales et internationales, comme ICEIS. Elle a publié plus de 10 papiers dans une revue et des conférences spécialisées. Elle est membre du laboratoire LAMIS, université de Tébessa, Algérie. Et aussi membre d'un projet de recherche CNEPRU. Elle a participé comme membre du comité d'organisation des 3 conférences : ICIST2011 à Tébessa, Algérie, ICIST2012 à Sousse, Tunis et IT4OD 2014 à Tébessa, Algérie.