UNIVERSITÉ DU QUÉBEC À MONTRÉAL

UTILISATION D'UNE ONTOLOGIE ET DU RÉSEAU SOCIAL FACEBOOK POUR LA MODÉLISATION DU CONTEXTE POUR LES APPLICATIONS MOBILES DÉPENDANTES DU CONTEXTE

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INFORMATIQUE

PAR

AHMED SAID LOUBIRI

NOVEMBRE 2012

REMERCIEMENTS

Je tiens à remercier sincèrement toutes les personnes qui m'ont apporté leur aide et qui ont contribué à l'élaboration de ce mémoire ainsi qu'à la réussite de ce formidable cursus universitaire.

Je souhaite, en premier lieu, adresser mes remerciements les plus chaleureux à mon directeur de recherche, Monsieur Obaid Abdellatif, pour m'avoir donné la chance de réaliser ce travail et s'être toujours montré à l'écoute et très disponible tout au long de la réalisation de ce mémoire. Il a toute ma reconnaissance pour l'inspiration, l'aide et le temps qu'il a bien voulu me consacrer et sans lui, ce mémoire n'aurait jamais vu le jour.

Mes remerciements s'adressent également à ma codirectrice Madame Fatiha Sadat pour sa générosité et son suivi rigoureux.

J'exprime ma gratitude à tous les professeurs de l'UQAM qui m'ont accompagné pendant ces années et m'ont permis d'avoir une formation de qualité. Je remercie notamment mes deux directeurs de programme, Monsieur Étienne Gagnon et Monsieur Roger Villemaire, pour leur encadrement rigoureux. Je remercie aussi Monsieur Petko Valtchev, directeur de mon laboratoire de recherche, pour sa disponibilité et ses précieux conseils et recommandations.

Je tiens aussi à remercier tout l'ensemble du personnel de l'UQAM, surtout Madame Lise Arsenault et Madame Louise Tremblay pour leur soutien tout au long de la maitrise.

Je remercie également Madame Cynthia Lisée, pour m'avoir appris les règles et les techniques de recherche et surtout d'avoir eu la gentillesse de lire et réviser la bibliographie et la qualité de la langue de ce mémoire.

J'adresse autant mes plus sincères remerciements à mes parents et ma famille pour tous les sacrifices consentis afin de me donner les moyens d'avoir cette formation et leur soutien tout au long de ces dures années d'études.

Enfin, je remercie tous mes proches et amis, qui m'ont toujours soutenu et encouragé au cours de la réalisation de ce travail, ainsi que tout ceux qui ont contribué de près ou de loin à l'élaboration de ce mémoire.

TABLE DES MATIÈRES

LISTE DES FIGURES

Figure Page

LISTE DES TABLEAUX

Tableau Page

RÉSUMÉ

L'informatique dépendante du contexte est un paradigme de l'informatique qui tient en considération les éléments de l'environnement d'une entité (utilisateur, périphérique, etc.).

Ce paradigme, qui prend une place de plus en plus importante grâce au développement rapide des technologies mobiles, a mené à la mise en place de différentes architectures et de divers modèles qui fournissent l'information sur le contexte.

Au niveau des architectures plusieurs solutions ont été proposées mais présentent tout de même certaines limites notamment, au niveau de la modélisation du contexte. De nombreux modèles ont été proposés dépendamment des taches à réaliser, mais se concentrent sur les capteurs physiques comme source principales des données de contexte et ignorent les informations qu'on pourrait acquérir de la fouille des différents profils en ligne d'un utilisateur.

Dans ce travail, nous proposons une architecture middleware pour le traitement et la gestion des informations du contexte pour un usager mobile en exploitant les données de *Facebook* comme source d'informations de contexte. Ainsi, nous analysons le réseau *Facebook* et le représentons sous forme de modèle ontologique. Nous fusionnons, ce modèle avec une ontologie du contexte relative à une simple application du domaine médical. Le modèle obtenu constitue notre solution proposée pour l'exploitation des données disponibles sur les réseaux en ligne dans la modélisation d'un contexte médical simplifié.

Mots-clés : informatique mobile, modélisation du contexte, application dépendantes du contexte, architecture middleware, réseaux sociaux, modélisation de l'ontologie, ontologie du contexte.

CHAPITRE I

INTRODUCTION

L'informatique mobile se repose sur les appareils portatifs et les réseaux sans fils. Cette technologie permet à l'utilisateur d'exécuter des tâches informatiques et d'accéder à l'information à n'importe quel endroit et n'importe quel moment. Durant ces dernières années, l'informatique mobile a remarquablement participé à l'amélioration de la vie quotidienne des personnes en leur permettant de faire différents traitements et d'avoir un accès permanent à l'information indépendamment de leurs déplacements. La mobilité des utilisateurs et des périphériques a donné naissance à un nouveau paradigme en informatique appelé informatique sensible au contexte ou informatique dépendante du contexte.

Le développement rapide des technologies de communication mobile, en l'occurrence des réseaux et appareils, et leur capacité et performance a engendré une augmentation exponentielle de son utilisation. Le marché des téléphones mobiles intelligents (*smart phones*) et des tablettes a connu une explosion des ventes dans les dernières années. Ces machines sont devenues aujourd'hui quasi indispensables à la vie de tous les jours.

Les téléphones intelligents disposent actuellement de meilleures capacités de traitement et une meilleure connectivité, leur permettant d'exécuter des tâches de plus en plus complexes et d'intégrer de nouvelles fonctionnalités.

Les applications mobiles sont des logiciels qui offrent à l'utilisateur différents services offerts par l'application comme la messagerie instantanée, la téléphonie IP, la commande à distance d'appareils et autres traitements dont certains peuvent dépendre du contexte de l'utilisateur de l'appareil.

Une nouvelle génération d'applications mobiles a donc vu le jour comme conséquence de l'évolution de ces technologies. Il s'agit des applications mobiles dépendantes au contexte. Ces applications offrent des services mieux adaptés à la situation de l'utilisateur et qui tiennent compte de certains caractéristiques contextuelles telles que ses préférences personnelles, son environnement physique, son emplacement géographique, ses déplacements, etc.

1.1 Problématique

Les différentes avancées dans le domaine de l'informatique mobile ont conduit au développement de systèmes dépendants du contexte. L'objectif de ces systèmes étant de fournir des informations qui dépendent de l'environnement de l'utilisateur afin d'améliorer son interaction avec l'appareil mobile.

De ce fait, plusieurs architectures ont été proposées et divers modèles de contexte ont été développés. Cependant, nous pouvons identifier les difficultés suivantes par rapport à la modélisation et l'utilisation des données du contexte, dont :

- Absence de formalisme et d'intelligence;
- Non exploitation des réseaux sociaux en ligne : les modèles de contexte existants ne tiennent pas compte de la grande quantité d'informations contextuelles qui existent déjà dans les réseaux sociaux en ligne tels que *Facebook*[1];
- Non-intégration des modèles d'intelligence (en particulier ceux basés sur les ontologies) dans les réseaux sociaux en ligne dans la modélisation du contexte : il n'existe pas de grands travaux de représentation des réseaux sociaux sous forme ontologique pour des fins d'intégration dans les modèles de contexte;
- Absence de lien en ligne entre ces informations et les applications dépendantes du contexte les plus utiles (telles que les applications médicales. En effet, il existe plusieurs applications dans le domaine médical, mais elles ne font pas le lien entre les

[1] https://www.facebook.com

informations sur le patient et son médecin traitant, lesquelles informations se trouvent déjà en grande partie dans les profils des réseaux sociaux comme *Facebook*.

1.2 Objectifs

Afin de remédier aux problèmes soulevés dans la problématique, nous nous sommes fixés les objectifs suivants pour la réalisation de notre mémoire :

- Création d'un modèle de contexte relatif au domaine d'application, en l'occurrence le domaine médical;
- Utilisation des ontologies pour la représentation du contexte;
- Présentation d'une architecture dépendante du contexte capable de gérer les informations captées et de les fournir à des applications mobiles;
- Couplage de l'environnement de l'utilisateur avec des informations tirées des réseaux sociaux en ligne comme *Facebook;*
- Utilisation dans le cadre d'une application simple dans le domaine médical qui utilise des données physiologiques mesurées pas le biais de capteurs corporels;
- Établissement de liens entre les profils *Facebook* d'un utilisateur et son médecin en vue de réagir aux changements des données physiologiques captées;
- Définition d'un ensemble de réactions dépendantes du contexte physiologique et spatio-temporel de l'utilisateur. (Non réalisé)

1.3 Méthodologie

Pour notre travail, nous allons adopter une méthodologie qui est basée sur la formalisation de la notion d'ontologie et son emploi dans le cas d'une application autour de l'environnement *Facebook*. Cette méthodologie s'articule autour des éléments suivants :

- Étude des infrastructures dépendantes du contexte par une revue de la littérature des systèmes déjà existants;

- Conception d'une architecture de gestion des informations du contexte inspirée des modèles existants et des différentes critiques de ces modèles;
- Étudier la structure du réseau social en ligne Facebook : ses objets et les relations entre eux;
- Conception d'une ontologie qui représente les éléments de Facebook et les relations entre ces éléments;
- Choix d'une ontologie de contexte de haut niveau qui représente les concepts généraux indépendants du domaine;
- Extension de l'ontologie de haut niveau avec les concepts du domaine choisi. Nous développerons notre modèle dans le cadre médical;
- Intégration de l'ontologie du domaine et de l'ontologie de Facebook afin d'avoir un modèle de contexte général;
- Implémentation du modèle du contexte avec *Protégé*[2] comme outil d'édition des ontologies;
- Définition des règles de raisonnement pour l'inférence du contexte;
- Simulation du raisonnement ontologique et des règles d'inférence.

1.4 Organisation du mémoire

Notre mémoire est organisé de la manière suivante :

Le premier chapitre présente les notions de base de notre travail. Ainsi, nous définirons la notion de contexte et les systèmes dépendants du contexte. Nous introduirons les ontologies, comme outil de représentation des connaissances contextuelles. Nous les définirons, délimiterons leurs domaines d'utilisation et déterminerons leurs caractéristiques, leurs critères de conception et leur classification;

[2] http://protege.stanford.edu

Le chapitre II est un état de l'art. C'est une revue de littérature des infrastructures dépendantes du contexte existantes. Nous donnerons plusieurs classifications possibles de ces infrastructures. Nous présenterons quelques systèmes existants tout en analysant leurs architectures et leurs modèles de contexte;

Dans le chapitre III nous suggérerons notre propre architecture en nous basant sur les exemples exposés dans la revue la littérature et l'analyse des avantages et limites de ces systèmes;

Le chapitre IV est un chapitre de réalisation. Nous parlerons des réseaux sociaux en ligne et des avantages de leur utilisation pour des systèmes dépendants du contexte. Nous prendrons *Facebook* comme exemple. Nous analyserons son architecture, les objets qui le constituent et les relations entre eux. Nous représenterons cette architecture sous forme d'ontologie que nous implémenterons sur un éditeur d'ontologies (*Protégé*);

A travers le chapitre V, nous expliquerons notre approche pour la conception d'un modèle de contexte pour le domaine médical. Nous intégrerons ce modèle avec l'ontologie développée pour *Facebook* afin de l'enrichir avec les informations disponibles de ce réseau social en ligne. Enfin, nous définirons des règles d'inférence pour ce modèle de contexte, qu'on testera par la suite.

CHAPITRE II

CONCEPTS DE BASE

2.1 La notion de contexte

La notion du contexte a été utilisée en linguistique et psychologie avant d'être adoptée en informatique, mais a aussi une origine lointaine et une longue histoire en philosophie [1]. La mobilité a donné une dimension importante au contexte qui a touché à de nombreux champs d'application en informatique comme l'informatique ubiquitaire, l'intelligence artificielle, le traitement de la langue naturelle, l'interaction personne-machine et la sécurité informatique, etc.

Il existe plusieurs définitions du contexte, qui varient selon le domaine d'utilisation et de l'objectif final de son application. La définition courante la plus utilisée est celle de Dey et al. [2] qui définit le contexte comme étant :

> « *Toute information qui peut être utilisée pour caractériser la situation d'une entité. Une entité est une place, une personne, un ou objet qui est considéré pertinent à l'interaction entre un utilisateur et une application, incluant l'utilisateur et l'application eux-mêmes* » [Notre traduction].

Une définition plus générale a été proposée par Zainol et Nakata [1] qui considère « *qu'une information est dite de contexte quand elle désigne une situation qui est pertinente pour l'interaction entre l'utilisateur et l'environnement de l'application* » [Notre traduction].

2.2 Les systèmes dépendants du contexte

Les systèmes dépendants du contexte ont été largement étudiés en informatique et plusieurs ont tentés de définir cette catégorie d'applications.

Dans [3], les auteurs affirment qu'en informatique dépendante du contexte, un utilisateur « *doit être capable d'accomplir facilement une action qui peut possiblement inclure une coopération et une collaboration avec d'autres utilisateurs en utilisant de multiples périphériques et réseaux alors qu'il se déplace dans l'environnement* » [Notre traduction]. Les périphériques et applications doivent alors s'adapter automatiquement aux besoins courants de l'utilisateur.

Les auteurs de [1] proposent une définition des systèmes sensibles au contexte qui « *permettent aux périphériques d'exploiter leurs informations environnantes et d'adapter leur comportement selon la situation actuelle de l'utilisateur* » [Notre traduction]. La situation actuelle de l'utilisateur caractérise donc son contexte. Afin de pouvoir fournir l'information appropriée, et donc pertinente, à l'utilisateur, les systèmes ont besoin de collecter et d'utiliser des informations de contexte.

Une définition plus généraliste a été proposée dans [2]. Les auteurs affirment qu'un système est dépendant au contexte « *s'il utilise le contexte pour fournir une information pertinente ou/et des services à l'utilisateur, où la pertinence dépend de la tâche de l'utilisateur* » [Notre traduction]. Cette définition est étroitement liée à celle que les mêmes auteurs ont proposée pour définir le contexte. Nous pouvons remarquer que ces auteurs utilisent la « pertinence » qui reste une notion subjective et peu tangible.

Les systèmes dépendants du contexte ont besoin de collecter différentes informations relatives au contexte de l'utilisateur à partir de diverses sources afin de s'adapter à sa situation ou même ses intentions. Ces sources peuvent être distribuées et souvent n'ont pas de relations ou de liens de connexions directes entre elles.

Les applications sensibles au contexte, permettent d'améliorer l'interaction personne-machine et n'ont pas besoin d'une intervention humaine pour s'adapter au contexte de l'utilisateur.

Il est alors très important de définir et modéliser le contexte de l'utilisateur afin d'offrir une information de contexte aux systèmes dépendants du contexte. Cette démarche nous pousse à concevoir des modèles susceptibles de représenter les différentes composantes du contexte et les relations qui existent entre elles. La plus prometteuse de ces méthodes repose sur les ontologies.

2.3 Les réseaux sociaux

Les réseaux sociaux en ligne ont envahi la toile au cours de ces dernières années, déclenchant un phénomène sans précédent. Ces réseaux comme *Twitter*[3]ou *Facebook* grandissent d'une manière exponentielle en termes de nombre d'utilisateurs. Ces services permettent aux personnes de créer des liens virtuels avec d'autres utilisateurs et de maintenir des relations définissant un réseau. Vu la popularité des réseaux sociaux et le nombre important des utilisateurs, il est primordial d'étudier leurs structures et fonctionnement.

L'analyse de ces réseaux peut nous conduire à des résultats intéressants puisqu'ils contiennent un nombre important de données personnelles et d'informations sur les utilisateurs et leurs relations sociales en ligne. L'aspect le plus intéressant des ces réseaux est qu'ils peuvent représenter un vrai reflet des relations sociales. Les personnes interconnectées sur un réseau social sont souvent des personnes qui ont des liens familiaux ou qui fréquentent le même travail, école, association ou qui partagent les mêmes passions ou visions politiques.

2.4 Les ontologies

Comme nous avons basé notre travail sur l'approche ontologique pour la modélisation du contexte, nous définissons ci-dessous les ontologies et nous présentons leurs domaines

[3] https://twitter.com/

d'utilisation, leurs caractéristiques, leurs classifications ainsi que les critères de leur conception.

L'origine du terme ontologie vient de la philosophie. Il s'agit de l'étude de ce qui existe : l'être, ses modalités et ses propriétés. Cette branche est considérée comme partie de la métaphysique. Cependant, longtemps, la notion d'ontologie a suscité l'intérêt des académiciens dans d'autres domaines tels que la linguistique et surtout l'informatique.

En informatique, une définition fréquente est celle de Gruber [4] qui définit l'ontologie comme étant « *une spécification explicite d'une conceptualisation* » [Notre traduction]. Plus explicitement, l'ontologie est la représentation des objets d'un domaine donné et les différentes relations qui existent entre ces objets. Les ontologies se basent sur un vocabulaire prédéfini qui rend cette représentation formelle.

Les ontologies sont alors très importantes pour la représentation des connaissances d'un domaine donné. Elles offrent, en plus, un langage formel. Les ontologies sont nécessaires pour le partage des connaissances.

Les ontologies sont utilisées dans différents domaines de l'informatique, et ce, particulièrement en intelligence artificielle (IA), où elles permettent de fournir des structures pour faciliter la représentation des connaissances et le processus d'inférence. Les ontologies ont été utilisées dans l'ingénierie de la connaissance, l'apprentissage automatique et aussi pour la désambiguïsation dans le traitement automatique du langage [5]. Les ontologies ont été aussi utiles dans la recherche d'information et la fouille de texte. Dans l'ingénierie logicielle, les ontologies fournissent des supports intéressants pour le développement logiciel.

Avec l'émergence du Web sémantique, les ontologies jouent un rôle crucial dans ce domaine. Elles permettent de structurer les informations du modèle sémantique et de supporter l'interopérabilité. Le Web sémantique dépend essentiellement de l'intégration des données sémantiques qui dépendent de la disponibilité d'une ontologie [6].

Une ontologie est constituée généralement de concepts (les différents objets et classes), leurs attributs (propriétés, caractéristiques) et les relations qui les lient.

Les principales caractéristiques des ontologies sont [7] :

- *Formelles* : elles sont exprimées dans une langue qui a une syntaxe bien définie et compréhensible par la machine de façon à ce qu'elles puissent être traitées automatiquement par les programmes appropriés. Les ontologies permettent le raisonnement logique et supportent des règles d'inférence. Contrairement au modèle relationnel (ER) et UML qui sont des langages semi-formels, les ontologies peuvent être directement traitées par la machine.
- *Lisibles par les humains* : autre que les programmes informatiques, les ontologies peuvent être comprises et développées par les informaticiens.
- *Vastes* : les ontologies couvrent largement les champs de connaissance d'un domaine donné.
- *Partageables* : les ontologies sont relativement faciles à combiner et à fusionner même si elles sont développées séparément. Les ontologies permettent à différents systèmes hétérogènes de partager diverses informations.

Les performances des systèmes basés sur les ontologies dépendent fortement de la qualité de la conception du modèle ontologique. Afin d'améliorer la conception de ces modèles, certains chercheurs [8] ont essayé de mettre en place un processus de développement des ontologies ainsi que certains critères de conception.

Pour le processus [8], le développement des ontologies obéit à une méthodologie définissant les étapes et les règles à suivre. Les activités énoncées sont, dans l'ordre : l'étude de faisabilité, l'acquisition des connaissances, la spécification des exigences de l'ontologie et finalement la définition de cycle de vie de l'ontologie et les ressources humaines nécessaires pour ce projet d'ontologie qu'on appelle la phase de plan.

Concernant les critères de conception des ontologies, Gruber [4] considère qu'ils dépendent de l'objectif final de leur développement et propose un ensemble de critères pour qui sont comme suit :

- *Clarté* : les termes de l'ontologie doivent être définis de manière claire, objective et complète. C'est-à-dire, que la définition des termes doit être efficacement communiquée par l'ontologie et qu'un prédicat doit être défini par des conditions « nécessaires et suffisantes ».
- *Cohérence* : les ontologies doivent inférer des faits qui sont cohérents avec la définition informelle de l'ontologie. Les classes et les relations du modèle ontologique doivent suivre une certaine logique.
- *Extensibilité* : nous pensons que l'extensibilité est l'une des caractéristiques majeures des ontologies. Gruber définit l'extensibilité de l'ontologie comme étant le fait « *d'être en mesure de définir de nouveaux termes pour une utilisation spéciale basée sur le vocabulaire d'une ontologie existante sans pour autant avoir recours à la révision des définitions existantes* » [Notre traduction].
- *Biais d'encodage minimal* : les ontologies sont souvent partagées par des sources hétérogènes et dont les langages de programmation diffèrent d'un système à un autre. De ce fait, et afin qu'elles soient compréhensibles et traitables par tous les systèmes, les ontologies doivent être développées indépendamment des langages d'implémentation.
- *Engagement ontologique minimal* : l'engagement d'une ontologie doit être minimisé en se contentant juste d'énoncer les principes liés à un domaine donné, et ce en définissant juste les termes essentiels du vocabulaire et en évitant ceux qui sont hors domaine.

Les ontologies sont différentes et peuvent être classifiées selon leurs niveaux de généralité représentés par la Figure 2.1 ci-dessous [9] :

- *Les ontologies de haut-niveau* (*top-level* ou *upper-level ontologies*) : représentent des concepts généraux et universels, qui ne dépendent pas d'un domaine particulier. Par exemple, une date.
- *Les ontologies du domaine :* sont des ontologies de moindre portée que les ontologies de haut-niveau. Elles spécialisent les ontologies de haut-niveau en représentant des concepts liés à un domaine donné comme l'ontologie pour les soins de santé (*health care domain ontology*).

- *Les ontologies de tâche :* elles spécialisent aussi les ontologies de haut-niveau en décrivant une tâche ou un événement générique comme la vente.
- *Les ontologies d'application :* sont plus restreintes que les ontologies du domaine ou de tâche et décrivent des concepts qui sont propres d'un domaine mais aussi d'une tâche particulière.

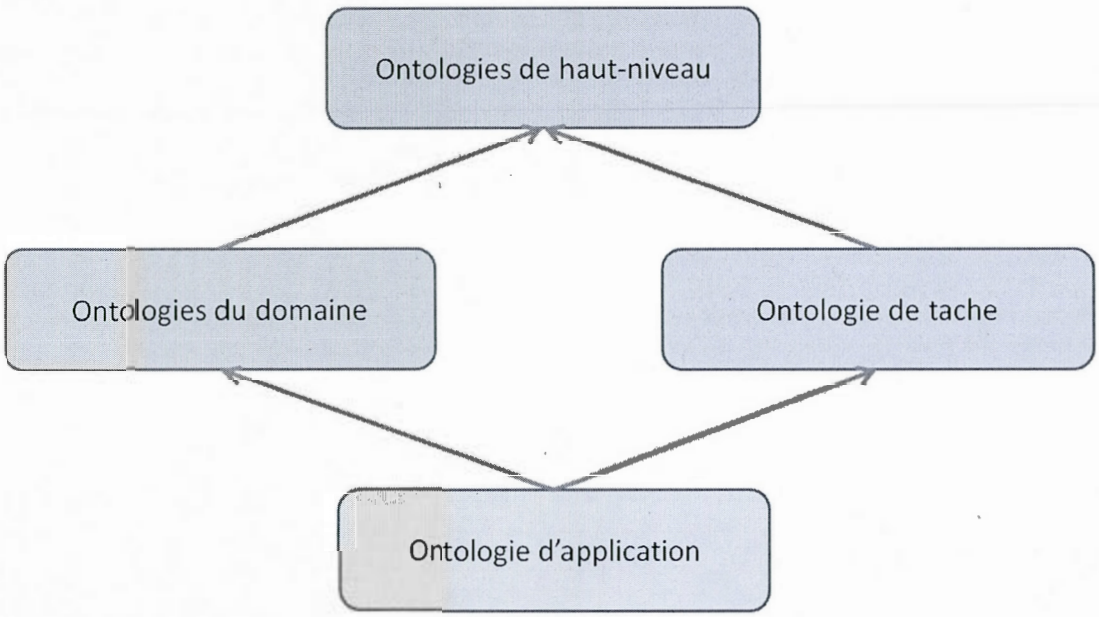

Figure 2.1 : Catégorisation de Guarino pour les ontologies [9].

2.5 Langages et outils de conception des ontologies

Il existe plusieurs langages pour la définition des ontologies dont : RDF (*Resource Description Framework*)[4], RDFS (*RDF Schema*)[5], et OWL (*Web Ontology Language*)[6].

[4] http://www.w3.org/RDF/
[5] http://www.w3.org/TR/rdf-schema/
[6] http://www.w3.org/TR/owl-features/

Dans notre projet, nous avons choisi de travailler avec le langage OWL pour les différents avantages qu'il nous fourni [10]:

- OWL est le langage le plus expressif parmi les autres énoncés précédemment;
- OWL nous permet de rajouter des contraintes de cardinalités sur les classes, ce qui n'est pas possible avec RDF, RDFs et RDFa.;
- OWL supporte la réutilisation, le mappage (fusion, alignement, intégration) et l'interopérabilité de l'ontologie. Ce qui est important dans notre cas, puisque nous envisageons d'enrichir une ontologie existante avec celle que nous avons développés;
- Il existe divers outils disponibles pour l'intégration des ontologies OWL dans le développement des applications logicielles tel que Protégé.

Pour l'outil d'édition des ontologies, nous avons choisi de travailler avec le logiciel *Protégé* qui est un éditeur libre développé en Java par l'Université de Stanford et qui est un des plus populaires dans le domaine.

Nous avons choisi cet outil pour son extensibilité, sa flexibilité et la convivialité des interfaces d'utilisation. De plus, il offre la possibilité de le combiner avec un moteur d'inférence.

CHAPITRE III

ÉTAT DE L'ART

3.1 Introduction

La mobilité a pris une place très importante dans la recherche en informatique, et les systèmes *Dépendants du contexte* suscitent désormais un intérêt majeur au sein de la communauté des chercheurs.

Durant la dernière décennie, de nombreuses infrastructures dépendantes du contexte ont été développées pour gérer ces systèmes. Cependant, ces infrastructures diffèrent beaucoup dans leurs architectures et leurs implémentations. Elles dépendent des exigences des systèmes et du processus d'acquisition, de transformation et de traitement de l'information de contexte. Ces systèmes sont différents non seulement dans l'architecture, qui est généralement organisée en couches, mais aussi du modèle du contexte adopté.

En se basant sur la méthode d'acquisition de l'information, on peut distinguer trois approches [11]: *l'acquisition directe*, l'utilisation du *middleware* et l'utilisation des *serveurs de contexte :*

a) L'*acquisition directe* consiste à récupérer directement les informations des capteurs de la part du client mobile. Les pilotes des capteurs sont installés dans le client et les capteurs sont souvent intégrés dans le périphérique.
b) L'approche du *middleware* est la plus souvent utilisée. Elle se base sur une architecture organisée en couches et permet l'utilisation des méthodes d'encapsulation. Les avantages de cette approche sont certainement la gestion de l'hétérogénéité de l'information et l'extensibilité facile du système.

c) L'utilisation du *serveur de contexte* étend l'approche du *middleware* et permet au client d'accéder à des sources de données distantes et distribuées. Cette structure permet d'alléger la charge du traitement de l'appareil mobile et d'avoir de meilleures performances pour la gestion du contexte.

D'un autre côté, en se basant sur la coordination des composantes et des processus dans ces applications, on peut distinguer deux approches différentes : le modèle dit de *widget*, et le modèle de *blackboard* [12].

Le modèle *widget* permet aux applications d'accéder aux informations du contexte par une requête directe à un widget particulier qui constitue un intermédiaire entre une source de données et une application. Un widget est « *une petite application multimédia interactive qui peut être trouvée dans les appareils mobiles* » [13] [Notre traduction]. Ces applications doivent s'inscrire auprès du widget pour pouvoir accéder à ces informations. Cependant, l'inscription et la désinscription des applications auprès des widgets est moins robuste et moins flexible en cas de changement rapide du contexte.

Le modèle *blackboard* permet aux applications d'accéder aux informations du contexte pas le biais d'une inscription à un élément centralisé qui est le *blackboard*. Cet élément gère le traitement interne des informations et notifie l'application quand un événement prédéfini se produit. Le *blackboard* est un modèle dont l'architecture est centrée sur les données. L'avantage de ce modèle est la facilité d'ajouter ou échanger des capteurs ainsi qu'une configuration flexible pour les changements dynamiques.

Les infrastructures différèrent aussi par leur modèle de contexte. Le modèle du contexte est une composante majeure des systèmes dépendants du contexte. Elle permet de modéliser, définir et interpréter le contexte. On peut les classer en trois catégories [14]: l'*approche axée sur l'application*, l'*approche axée sur le modèle* et l'*approche basée sur les ontologies* :

a) Le *modèle axé sur l'application* modélise le contexte pour des applications spécifiques. C'est-à-dire, que le contexte est représenté par un modèle uniforme qui n'est pas extensible ou applicable par d'autres ressources. De ce fait, ce modèle souffre d'une faible expressivité et offre un formalisme limité.

b) L'*approche basée sur le modèle* modélise le contexte en le représentant par des modèles conceptuels graphiques ou orientés-objet tels que le modèle entité-association ou UML. Le modèle du contexte est représenté en utilisant des diagrammes et des bases de données relationnelles.

c) L'*approche basée sur les ontologies* utilise les ontologies pour la représentation du contexte et pour le partage des connaissances entre les sources distribuées de données. Les ontologies définissent les concepts et les relations entre eux. Elles offrent une alternative très intéressante pour la modélisation du contexte grâce à leur forte expressivité, leur extensibilité facile et les techniques de raisonnement qu'on peut y appliquer. Les ontologies permettent aussi de surpasser le problème d'hétérogénéité des sources d'information. Ces modèles utilisent le langage OWL (*Web Ontpolgy Language*)[7] pour la définition des ontologies.

3.2 Quelques modèles existants

Nous énumérons ci-dessous certaines de ces architectures.

3.2.1 Le Service-Oriented Context-Aware Middleware (SOCAM) [15]

SOCAM est une architecture introduite qui fournit les informations de contexte aux services mobiles dépendants du contexte. SOCAM est une architecture en middleware qui se base sur l'approche orientée services. Cette approche permet aux applications d'intégrer plusieurs services en ligne pour leurs domaines spécifiques de contexte.

Le middleware offre la possibilité d'intégrer et de réutiliser aisément des composantes dans le système telles que des nouveaux capteurs. Il permet aussi de faire abstraction de

[7] www.w3.org/TR/owl-features

l'hétérogénéité des sources de données. Ce système consiste en des composantes organisées en modules qui permettent d'effectuer des tâches spécifiques. Ces modules communiquent et interagissent entre eux et permettent d'acquérir les informations du contexte à partir de différentes sources distribuées, de traiter et d'inférer le contexte en se basant sur un module de raisonnement et de livrer l'information du contexte aux services du contexte localisés en haut de l'architecture.

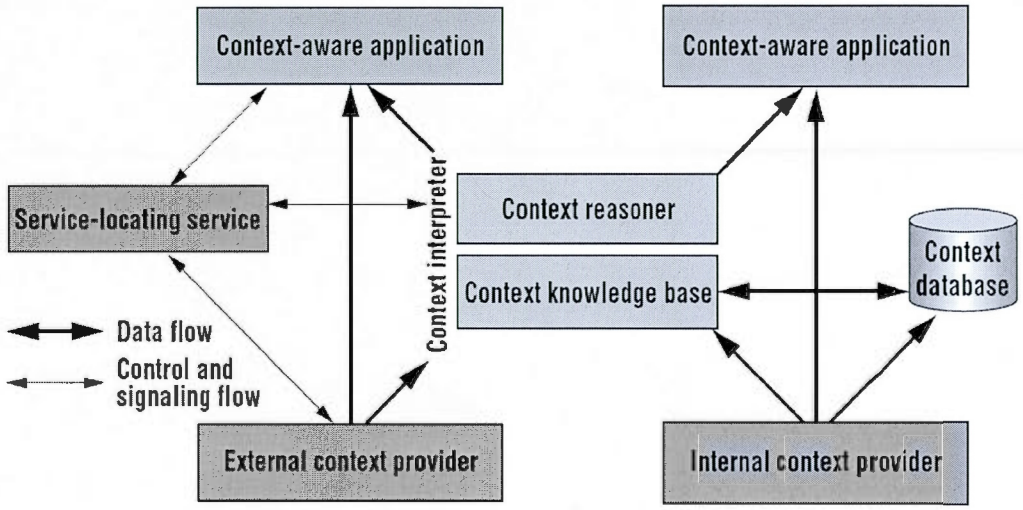

Figure 3.1 : Architecture SOCAM [15, fig 1]

L'architecture est organisée en trois couches (Figure 3.1) : la *couche de captage du contexte*, le *middleware* et la couche *des applications dépendantes du contexte* :

La couche de captage est constituée des différents capteurs utilisés pour saisir les données du contexte. Ces capteurs peuvent être physiques comme un GPS ou un détecteur de mouvement, ou virtuels comme un service Web.

a) La *couche Middleware* est constituée des différents modules. Les *fournisseurs du contexte* récupèrent les données de différentes sources de contexte constituées des capteurs physiques et des capteurs virtuels. Le *fournisseur du contexte* s'enregistre alors dans un *registre de services* afin d'être intégré à la plateforme. Les *fournisseurs du contexte* peuvent être externes ou internes. Ils permettent notamment de représenter les données capturées sous forme de description OWL. L'*interprétateur*

du contexte est un autre module du système qui permet de traiter le contexte à l'aide d'un raisonnement logique. La fonction de ce module est d'inférer le contexte de haut niveau à partir des informations de contexte de bas niveau comme la géo-localisation. Ce module est composé d'un *raisonneur* et une *base de connaissance du contexte*. Le *raisonneur* est constitué des règles d'inférence qui sont prédéfinies par les experts du domaine en forme de logique de premier ordre. La *base de connaissances du contexte* contient les ontologies du contexte et leurs instances. Le *middleware* contient aussi une base de données du contexte où sont stockés les ontologies du contexte et un historique des contextes inférés précédemment. Le *service de localisation des services* est le module qui permet aux clients de localiser les fournisseurs du contexte et les interprétateurs du contexte pour un domaine donné.

b) La couche supérieure est la couche d'*applications dépendantes du contexte.* Elle contient des serveurs d'applications dépendantes du contexte qui demandent les informations du contexte en s'inscrivant au service de localisation des services.

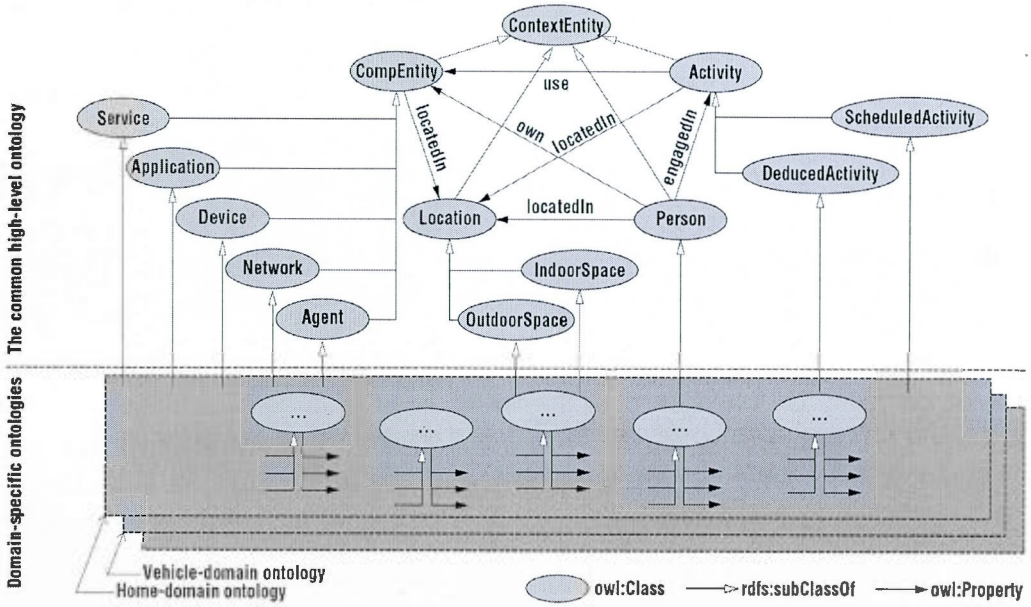

Figure 3.2 : Ontologie du modèle de contexte SOCAM [15, fig 2]

Pour le modèle du contexte (Figure 3.2), les auteurs adoptent l'approche basée sur les ontologies qui permet une description sémantique du contexte. Le langage utilisé pour la description des ontologies est OWL. La conception des ontologies est faite d'une manière hiérarchique sur deux niveaux qui réduit la taille et la complexité de l'ontologie. L'ontologie de haut niveau définit les concepts généraux tels que : *personne*, *localisation*, *activité* et *entité computationnelle*. L'ontologie spécifique au domaine est un ensemble d'ontologies de bas niveau qui représentent les concepts généraux dans chaque sous-domaine et leurs propriétés.

Deux ontologies de domaine spécifique ont été développées : *maison* avec 89 classes et 156 propriétés et *véhicule* avec 32 classes et 57 propriétés. Ce système fournit des résultats acceptables en termes de vitesse du traitement et du nombre de requêtes traitées. Cependant, les performances de l'interprétateur avec le raisonnement ontologique sont moins intéressantes qu'avec des règles prédéfinies par le développeur. Le raisonnement ontologique devrait être allégé afin de permettre à la plateforme de fournir de meilleurs résultats.

3.2.2 Le modèle Context Broker Architecture (CoBrA) [10]

CoBrA est une architecture centralisée basée sur l'agent. Un agent est une entité physique ou virtuelle qui accompli des taches spécifiques en agissant dans un environnement : détecter de états, communiquer avec d'autres agents dans un environnement distribué, etc. [16].

Cette approche permet à divers agents distribués d'être regroupés sous un même modèle de contexte qui leur permet de partager leurs informations via une infrastructure (Figure 3.3).

L'architecture contient les éléments suivants :

- L'*agent de contexte* « *Context broker* » est l'élément central de l'architecture, il permet de gérer le model de contexte pour les différents agents et périphériques qui y sont connectés. Chaque agent permet d'obtenir des informations du contexte à partir des capteurs ainsi que d'autres agents, et les transformer en un modèle de contexte formel afin qu'ils puissent être partagés par tous les agents. Les agents peuvent être des applications mobiles. L'agent de contexte est composé de quatre modules

principaux : la *base de connaissance du contexte*, le *module de raisonnement*, le *module d'acquisition* et le *module de gestion de la confidentialité*.

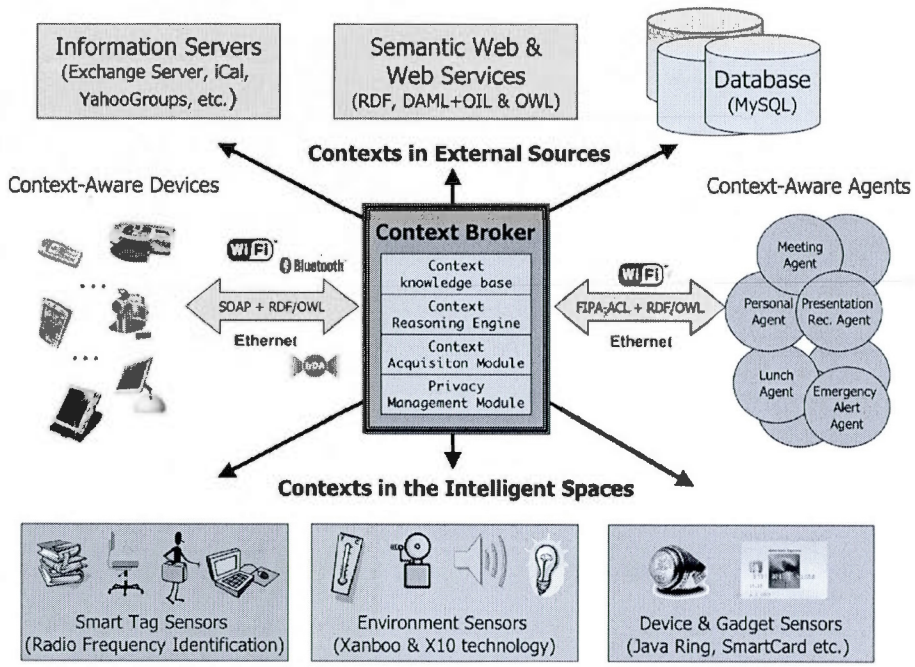

Figure 3.3 : Architecture CoBrA [10, fig1].

- La *base de connaissance du contexte* contient le modèle de contexte de chaque domaine et comporte l'*ontologie du domaine*, les *heuristiques du domaine* (les entités clés du domaine et les relations entre elles) et la *politique de confidentialité des agents.*

- L'*ontologie du domaine* est un ensemble de concepts et de relations entre eux dans un domaine spécifié. Les heuristiques *du domaine* servent à résoudre les problèmes de cohérence. La *politique de confidentialité des agents* définit les informations qui peuvent être partagées par chaque agent.

- La *base des connaissances du contexte* est reliée au module de raisonnement, responsable de l'inférence de l'information du contexte et composé de trois sous

modules qui sont le *module de raisonnement du contexte*, le *module de raisonnement de l'ontologie* et le *module du maintien de la connaissance*.

- Le module de *raisonnement du contexte* est responsable du raisonnement sur un domaine donné à travers des règles logiques d'inférence. Le *module de raisonnement de l'ontologie* agit par rapport aux contraintes sur les relations entre les concepts dans l'ontologie du domaine. Le *module de maintien de la connaissance* vérifie l'uniformité des informations de la base de connaissance du contexte.

- Le *module d'acquisition de contexte* est le middleware qui permet de récupérer les informations de contexte de bas niveau à partir des divers capteurs. Ce module a une architecture en deux couches : les *capteurs du contexte* et l'*interprétateur du contexte*. Les capteurs, physiques ou virtuels, fournissent les informations du contexte. L'interprétateur du contexte récupère ces informations et les interprète.

- Le *module de gestion de la confidentialité* est responsable de la communication du *Context Broker* avec les autres agents en définissant des protocoles de transmission appropriés.

- Pour le modèle de contexte, CoBrA se base sur l'approche axée sur les ontologies. CoBrA-Ont [17] est un ensemble d'ontologies exprimées dans le langage OWL et qui définissent les concepts et les relations qui les lient dans chaque domaine. Cet ensemble contient 41 classes et 36 propriétés qui sont catégorisées selon des domaines distincts : places, agents, contexte de localisation des agents et contexte d'activité des agents.

CoBrA est une infrastructure intéressante mais qui a certaines limites. L'architecture est composée de plusieurs modules comportant eux-mêmes d'autres sous-modules, ce qui ralentit la vitesse de traitement de l'information [17]. De plus, certains modules comme les heuristiques du domaine et le module de maintien de la connaissance jouent le même rôle de maintien de la cohérence de la base de connaissances du contexte. Pour le modèle du contexte l'ontologie développée, CoBrA-ONT n'est pas assez élaborée pour couvrir de vastes

contextes. CoBrA offre cependant un haut niveau de confidentialité et de sécurité pour l'échange des informations grâce au module de politique de confidentialité des agents.

3.2.3 Context-Awarness Sub-Structure (CASS) [18]

CASS est un middleware (Figure 3.4) orienté vers le serveur. Il est structuré en différents modules : l'*interprétateur*, le *récupérateur du contexte*, le *module des règles*, l'*écouteur des capteurs* et *la base de données*.

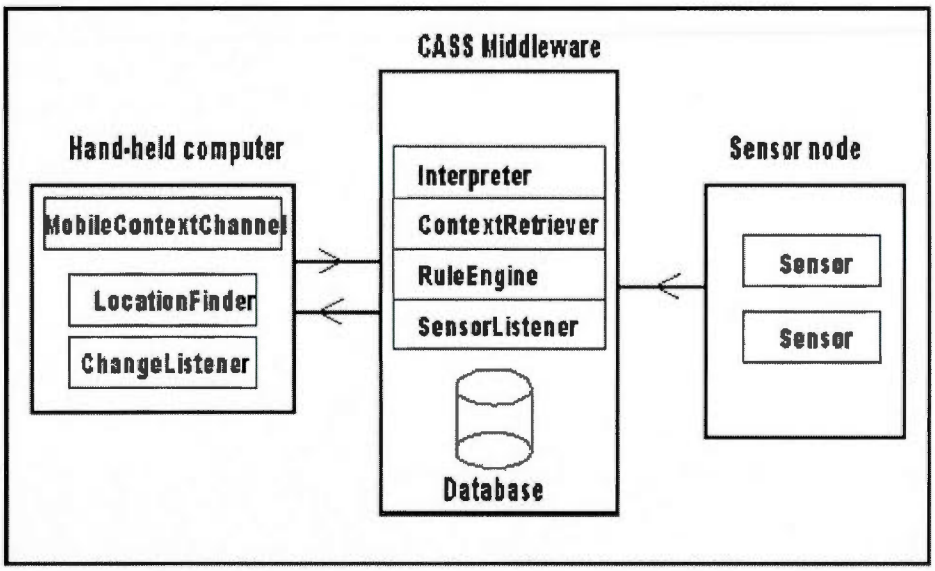

Figure 3.4 : Architecture CASS [18].

La conception du système est faite selon un modèle UML (Figure 3.5) où les modules sont représentés sous forme de classes. L'*écouteur des capteurs* reçoit les mises à jour de la part des nœuds de capteurs et enregistre les informations du contexte reçues dans la base de données. L'*écouteur des capteurs* est relié à l'écouteur des changements qui est responsable de communiquer aux périphériques le changement de contexte. Le *récupérateur du contexte* récupère les données stockées dans la base de données.

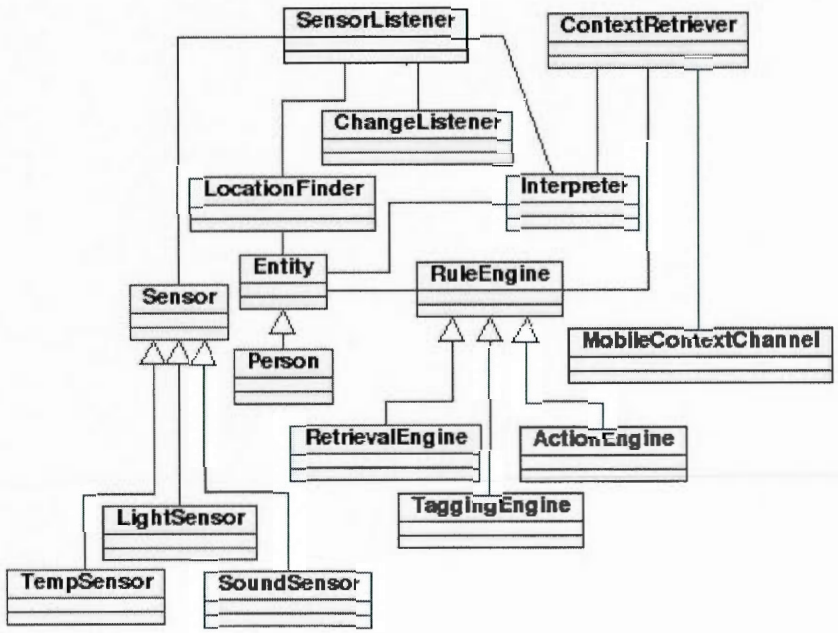

Figure 3.5 : Modèle de contexte CASS [18].

Le système utilise une base de données relationnelle où sont stockées le contexte, les données utilisateurs, les connaissances du domaine et les règles d'inférence.

L'inférence du contexte dans l'infrastructure CASS se fait à l'aide du *module d'inférence* et la *base de connaissances*. Le *module d'inférence* utilise une technique de chaînage avant. Cette technique d'intelligence artificielle consiste à utiliser des faits pour en inférer d'autres, ceux-ci sont à leur tour utilisés pour déduire de nouvelles conclusions. La base de connaissances contient les *règles de production*. Elle est stockée sous formes de tables dans la base de données relationnelle.

Les limites de ce système consistent dans le modèle orienté-objet pour la gestion du contexte. Ce modèle n'est pas facilement extensible et offre une expressivité limitée. De plus, l'accès aux bases de données relationnelles s'avère couteux en termes de temps; ce qui influence négativement les performances de l'infrastructure.

3.2.4 Le projet Hydrogène [19]

Ce projet est une plateforme de contexte disposée en couches. Afin de séparer les traitements et les tâches effectués, une architecture (Figure 2.6) sur 3 niveaux a été proposée : *couche d'adaptation*, *couche de gestion* et *couche application*.

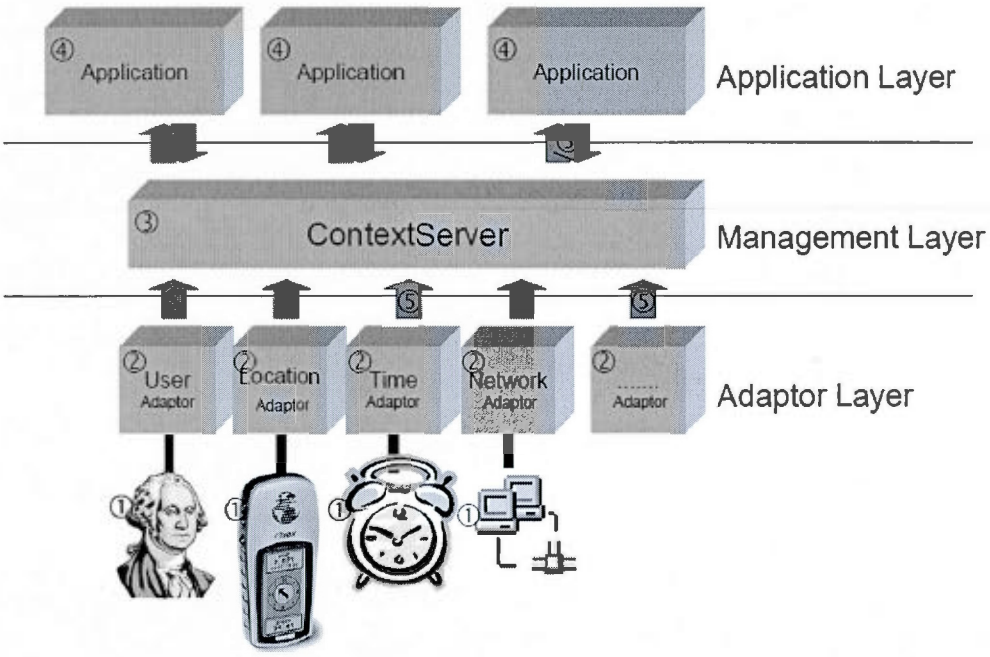

Figure 3.6 : Architecture Hydrogène [19].

La couche d'adaptation récupère les données du contexte à partir des différents capteurs organisés par domaine : localisation (position), utilisateur, réseau, etc.

La couche de gestion contient la composante principale de l'architecture appelée serveur de contexte. Le serveur de contexte permet aux applications de s'inscrire aux différentes sources de contexte. Il enregistre toutes les données du contexte récupérées de la couche d'adaptation. Cette couche utilise une communication pair à pair pour s'approvisionner et partager les informations du contexte.

La couche application contient les applications qui sont inscrites dans l'infrastructure pour acquérir les informations du contexte. L'application peut s'inscrire au contexte de deux façons possibles : synchrone ou asynchrone.

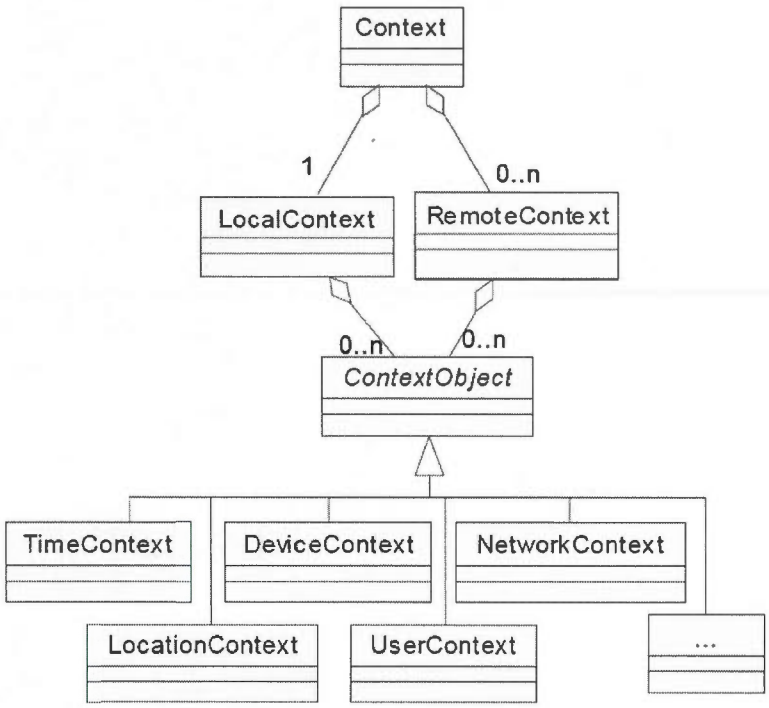

Figure 3.7 : Modèle de contexte Hydrogène [19].

Pour le modèle de contexte, cette infrastructure se repose sur un modèle entité-association (Figure 3.7) qui distingue entre le *contexte local* et le *contexte à distance*. Ce contexte (classe) est constitué du contexte local et du contexte à distance, qui contiennent des *objets de contexte*. Ces *objets de contexte* sont une spécialisation des cinq types d'informations de contexte fournies par les capteurs : la *date*, l'*emplacement*, le *périphérique* (type, caractéristiques, etc.), l'*utilisateur* et le *réseau*. Cette composante permet la conversion des informations de contexte en données XML.

Les avantages de cette plateforme sont la structure en couches qui permet de séparer les tâches comme l'accès aux données et la récupération des données par l'application. La couche de gestion permet d'éviter l'accès multiple à une seule source de données par les

applications. La localisation de la structure dans l'appareil mobile lui-même permet d'éviter les problèmes de déconnection. La communication pair à pair permet d'éviter le recours à un serveur central. Les limites de cette architecture résident dans son incapacité à enregistrer l'historique des contextes; ce qui ne permet pas le recours à des algorithmes d'apprentissage afin d'améliorer le traitement du contexte. Le modèle entité-association adopté est moins expressif que le modèle ontologique.

3.2.5 Architecture Smart-M3 [20]

C'est une infrastructure pour les espaces intelligents de code source libre de Nokia. Cette architecture repose sur le modèle blackboard et un model de contexte basé sur l'approche ontologique. L'élément central de cette architecture est la composante de *courtage sémantique* (*Sematic Information Broker* ou *SIB*). Les *processeurs de connaissance* (*Knowledge Processor* ou *KP*), quant à eux, accèdent et traitent l'information.

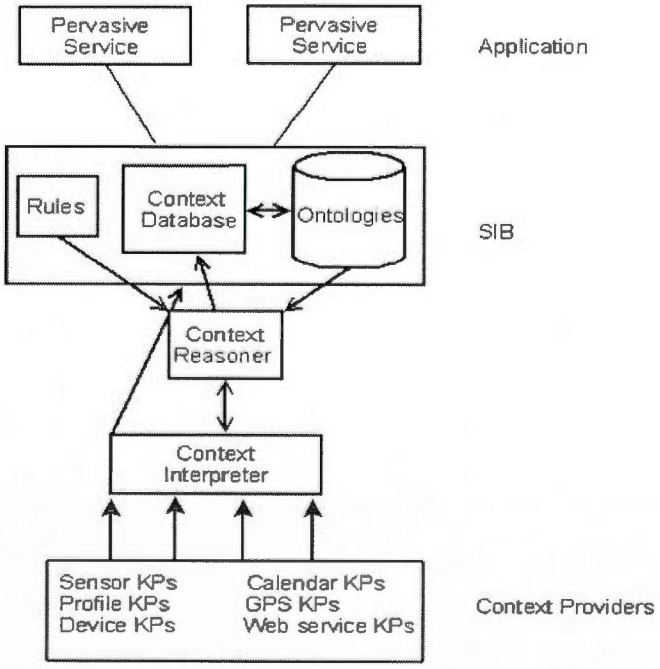

Figure 3.8 : Architecture Smart-3 [20].

L'architecture (Figure 3.8) est composée des éléments suivants :

a) *Fournisseurs de contexte* (context provider) *:* donnent les informations atomiques concernant le contexte de l'utilisateur qui peuvent être des données de bas niveau ou des données du service Web.
b) *Interpréteur du type des donnés de contexte :* les données brutes issues de diverses sources hétérogènes doivent être converties en données sémantiques qui peuvent être utilisées par les différents services.
c) *Raisonneur et interpréteur des règles du contexte :* cette partie est chargée d'inférer de nouvelles données de haut niveau à partir des informations atomiques du contexte. Ces règles sont définies par les développeurs de processeurs d'information.
d) *Ontologies* : représentent les données issues des fournisseurs de contexte et celle inférées par le raisonneur.
e) *Règles d'inférence* : sont nécessaires au processus d'inférence du raisonneur. Ces règles se basent sur la logique et définissent la manière d'inférer l'activité de l'utilisateur à partir des informations du contexte données par le fournisseur du contexte (KP).

Le modèle de contexte (Figure 3.9) de cette infrastructure est basé sur les ontologies. Le contexte de l'utilisateur peut être divisé en deux parties : *contexte atomique* et *contexte inféré*.

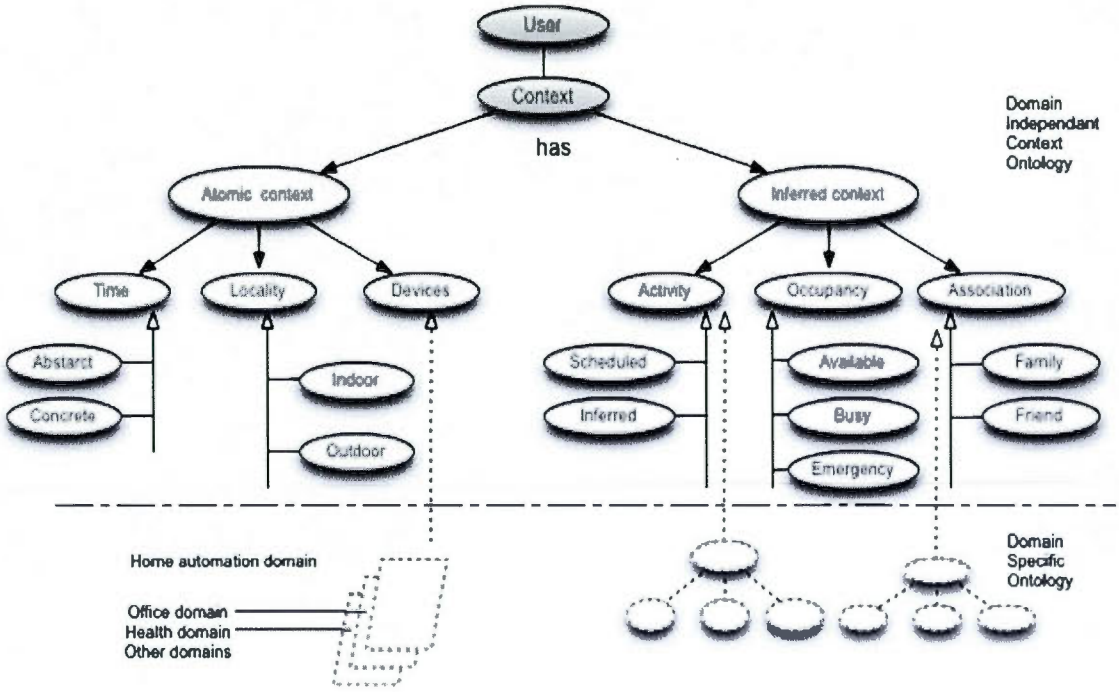

Figure 3.9 : Modèle d'ontologie Smart-3 [20]

Le *contexte atomique* est représenté par les données du contexte obtenues directement des fournisseurs d'informations du contexte (date, localisation, appareil). Le *contexte inféré est composé* des informations déduites des données de contexte (activité, occupation, association).

Le modèle ontologique est divisé en deux parties : les ontologies de haut niveau et les ontologies du domaine. Le contexte de l'utilisateur est représenté par six concepts : temps, emplacement, périphérique, activité, occupation et association.

L'architecture Smart-M3 en s'appuyant sur le modèle *blackboard,* est flexible aux changements dynamiques de l'environnement de l'utilisateur. Le modèle de contexte permet de supporter les sources d'information hétérogènes et offre l'extensibilité et flexibilité du système ainsi qu'une forte expressivité. Les limites de ce modèle consistent au maintien de cohérence de la SIB.

CHAPITRE IV

ARCHITECTURE

4.1 Introduction

En vue d'offrir une information pertinente et adaptée au changement dynamique de l'environnement de l'utilisateur, les applications mobiles sollicitent des services sensibles au contexte qui suscitent un grand intérêt de recherche.

Un service dépendant du contexte est capable d'aller chercher et acquérir des informations de différentes sources, de les interpréter et en inférer le contexte courant d'un utilisateur. Pour arriver à cet objectif, nous devons développer une infrastructure complète qui permet de faire les différentes tâches d'une manière efficace et rapide.

La complexité de ces traitements et les différentes étapes nécessaires à l'obtention d'une information du contexte, nous mènent à modéliser une architecture contenant de différents services et composantes qui communiquent et interagissent entre eux.

Par conséquent, nous proposons un middleware qui inclut des modules indépendants capables d'acquérir des informations de différentes sources et de les traiter afin d'en déduire le contexte et le fournir aux serveurs appropriés.

Dans ce chapitre, nous présentons le middleware proposé. Nous exposons ces différents modules en précisant leurs rôles respectifs et en détaillant leur fonctionnent.

4.2 Architecture proposée

Nous présentons une infrastructure inspirée des concepts présentés dans l'état de l'art et composée de trois principaux niveaux (Figure 4.1). La couche capteur qui contient les capteurs susceptibles de fournir des informations du contexte. La couche middleware est responsable de la récupération et du traitement des informations. Et enfin, la couche service de contexte qui sollicitent le middleware afin d'obtenir des informations du contexte.

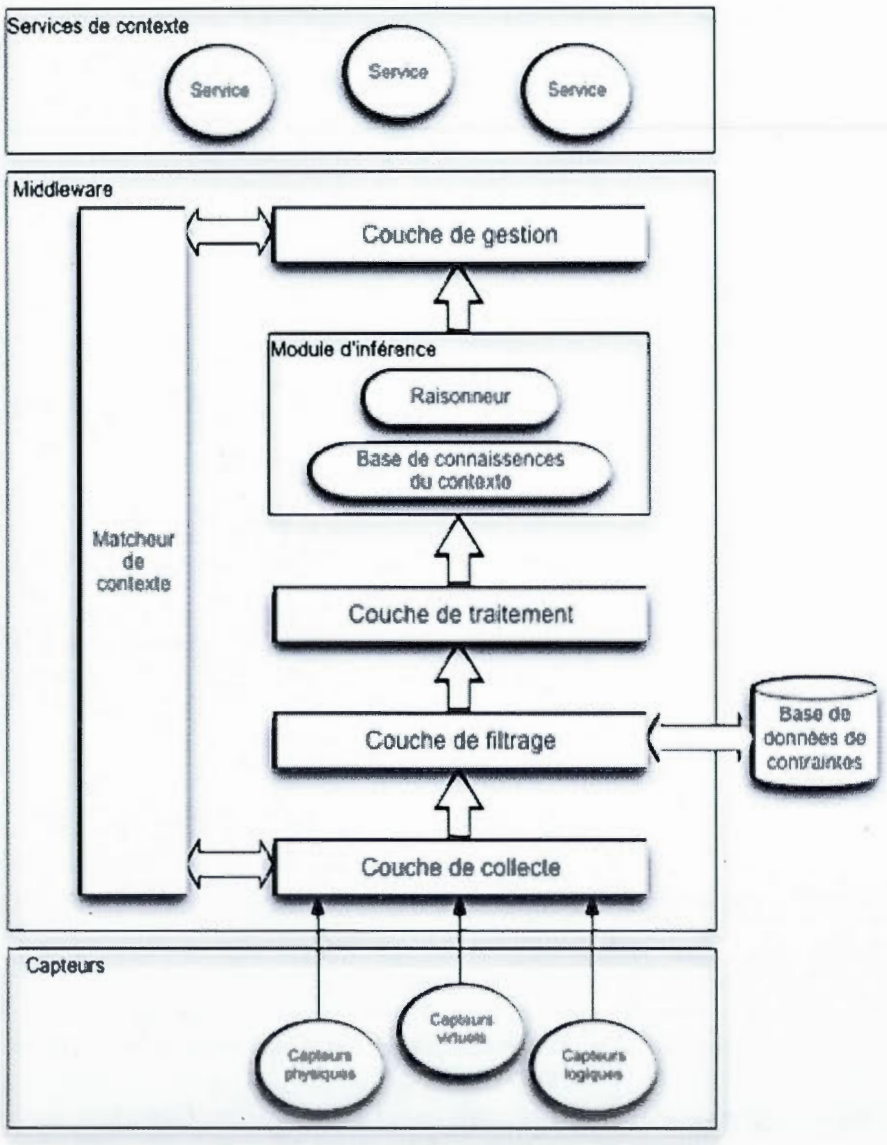

Figure 4.1 : Architecture de notre système dépendant au contexte proposé.

4.2.1 La couche capteurs

C'est la couche qui contient toutes les sources de données qui peuvent fournir des informations utiles pour le contexte.

On peut distinguer trois types de capteurs [11]:

- Les *capteurs physiques* qui sont plus fréquents et disponibles et fournissent généralement des données brutes. Pour les applications dépendantes du contexte, le plus important capteur est celui de la localisation (GPS) mais on peut inclure aussi d'autres capteurs physiques.
- Les *capteurs virtuels* qui sont disponibles essentiellement à travers les services Web. Par exemple on peut connaître la température en accédant en ligne à un site de météo, on peut aussi détecter l'activité de l'utilisateur par sa présence dans les réseaux sociaux (en ligne) ou en accédant à son agenda électronique.
- Les *capteurs logiques* qui combinent les deux types de capteurs énumérés précédemment afin d'en dériver l'information du contexte en utilisant la logique.

Dans notre travail, nous exploiterons les capteurs corporels comme capteurs physiques. Ces capteurs corporels sont des petites composantes attachées dans différentes région du corps de l'utilisateur et qui fournissent des interfaces de communication entre cet utilisateur et un système distant [21]. Ce système est implémenté à l'intérieur de l'appareil mobile dans notre cas. Cette configuration permet de connecter tous les capteurs afin de construire un réseau de capteurs corporels « *Body Sensor Network* » et afin de collecter les informations d'une manière centralisée. Les capteurs peuvent être interconnectés par des technologies comme le Zigbee, le Blutooth et le Wifi (IEEE 802.11). Ces mêmes technologies peuvent être utilisées pour la transmission des données vers l'appareil mobile. Nous pouvons aussi utiliser comme capteurs de données physiologiques des appareils de mesures comme le thermomètre pour la température, le tensiomètre pour la pression artérielle et le glucomètre pour le taux de glycémie. De cette façon, les capteurs sont utilisés d'une manière occasionnelle et ne sont plus continuellement attachés au corps de l'utilisateur. Cependant, ces appareils doivent être équipés d'une des technologies citées plus haut pour pouvoir envoyer les données au mobile.

Le captage des données peut se faire d'une façon manuelle, c'est-à-dire au moment des prises de mesures faites par l'utilisateur lui-même (dans le cas d'une prise sanguine pour la glycémie par exemple), ou d'une façon automatique par le réseau de capteurs (température corporelle, rythme cardiaque). La fréquence du captage automatique est définie et programmée selon les besoins médicaux de chaque utilisateur.

Nous utilisons aussi *Facebook* comme capteur virtuel. Les données de cette plateforme sont collectées d'une manière dynamique et envoyées a la base de connaissance a chaque traitement.

4.2.2 Le middleware

Notre middleware est structuré en couches superposées qui divisent le processus en sous-étapes dépendamment de la tâche à réaliser (Figure 4.1). La disposition en couches permet d'organiser les processus de gestion de l'information et la génération du contexte.

Le middleware a pour but de transformer les données fournies par différentes sources hétérogènes en une information de contexte utilisable par les services dépendants au contexte. Depuis leur acquisition à partir de divers capteurs, les données suivent un processus prédéfini. L'information est tout d'abord récupérée, ensuite filtrée, traitée, inférée et finalement fournie au service approprié. L'architecture du middleware est ainsi organisée en couches qui effectuent des tâches spécifiques :

4.2.2.1 La couche de collection des données

C'est la couche responsable de la récupération des données des différents capteurs. Elle fournit un ensemble d'interfaces et méthodes abstraites qui utilisent les pilotes pour les capteurs physiques et les APIs pour les capteurs physiques et implémente des requêtes afin de récupérer les données. Cette couche doit s'enregistrer à une autre composante du système appelée module de correspondance « *Context Matcher* » pour qu'elle puisse détecter l'ensemble des capteurs. Ce mécanisme facilite l'ajout de nouveaux capteurs ou la modification de capteurs déjà existants. Dans notre application, nous avons décidé de placer

cette couche dans l'appareil mobile. Ainsi, la collection des données se fait à travers les technologies de communication énumérées dans la section précédente. Cette étape implique aussi la mise en place de mécanismes de collection de données qui tiennent en compte les différentes limites des appareils mobiles comme la capacité de stockage et l'autonomie de la batterie. Quelques recherches ont été menées dans le but de présenter des techniques efficaces de collecte des données à partir réseau de capteurs sans-fil [22] [23].

4.2.2.2 La couche de filtrage des données

Les données captées et ensuite récupérées peuvent contenir des erreurs de mesures (bruits). Les capteurs peuvent fournir des données erronées ou des données qui sont inutiles pour les services du contexte. Afin d'améliorer la qualité des données qui seront fournies à la couche supérieure et dans le but d'économiser nos ressources, nous avons ajouté cette couche à notre architecture. Elle sera liée à une base de données qui contient l'ensemble des contraintes sur les données. Ces contraintes sont prédéfinies par le programmeur et peuvent être modifiées ou mises à jour d'une manière manuelle ou dynamique.

4.2.2.3 La couche de traitement des données

Les données reçues de la couche de filtrage sont issues de diverses sources distribuées et sont donc hétérogènes. Les données sont aussi très techniques puisqu'elles représentent des valeurs numériques avec des unités de mesures différentes. Ces données sont donc non appropriées et non compréhensibles pour les services dépendants au contexte. Alors, la couche de traitement des données est responsable de la transformation de toutes les données captées dans un langage formel, partageable et qui peut fournir des données utilisables par la couche suivante.

Les données reçues dans différents formats sont transformées à des descriptions OWL. Cette opération nécessite la mise en place d'un système qui en premier temps convertit les données brutes en documents XML contenant une description complète de la mesure incluant sa source, sa valeur et l'heure de la capture. Ces documents XML seront par la suite

automatiquement transformés en descriptions OWL (*Individuals*). Plusieurs mécanismes de transformation sont disponibles dont celui proposé dans [24].

Ces descriptions sont par la suite utilisées par le module d'inférence.

4.2.2.4 Module d'inférence

Cette couche est responsable d'inférer le contexte à partir données récupérées de la couche précédente en utilisant le raisonnement logique. Elle permet d'élever l'information à un niveau supérieur d'abstraction. L'information du contexte peut être classée sur deux nivaux dépendamment de son niveau d'abstraction.

On peut distinguer l'information de bas niveau déduite d'une simple donnée brute capturée comme la température, le niveau de luminosité, etc. Cependant, cette information est très souvent insuffisante. Alors elle est combinée avec d'autres informations capturées afin d'en déduire une information de contexte plus pertinente. Par exemple, un utilisateur qui est chez lui peut être dans le jardin ou couché dans la chambre, ce qui n'est pas du tout le même contexte. L'information de localisation (GPS) nous permet simplement de savoir qu'il est à la maison. Par contre, en utilisant d'autres capteurs, comme le thermomètre pour la température ou un micro pour le niveau de son, combinés avec la localisation une déduction plus précise du contexte.

Autre que dériver le contexte de haut niveau du contexte bas niveau, les tâches de cette couche consistent aussi à interroger la base de données du contexte et vérifier la cohérence des informations du contexte [25]. La couche d'inférence encapsule divers modules qui permettent la réalisation de ces tâches.

4.2.2.5 La base de connaissance du contexte

Cette composante assure l'interrogation et la modification des connaissances contextuelles. Elle contient les ontologies du contexte d'un domaine donné et ses instances. Ces ontologies représentent les objets, les relations qui existent entre eux et leurs instances.

Le modèle ontologique a été préféré aux autres approches puisqu'il fournit une forte expressivité qui permet de spécifier les concepts et leurs relations et qu'il fournit une formalisation des objets du monde réels dans un langage compréhensible par la machine [11]. Les ontologies sont aussi extensibles et permettent de rajouter d'autres concepts ou d'être couplées avec d'autres ontologies. De plus, nous pouvons y appliquer des techniques de raisonnement dont l'utilité est décrite dans la section suivante.

Les instances des ontologies peuvent être prédéfinies par l'expert du domaine et sont alors sauvegardées dans la base ainsi que les ontologies lors de l'initiation du système. Elles peuvent aussi être récupérées de la couche précédente en cas de données fournies par les capteurs. Elles sont alors chargées dans la base pendant l'exécution du système [15].

4.2.2.6 Le raisonneur

Le raisonneur est le module responsable de l'inférence des informations de contexte de bas niveau (contexte direct) à des informations de contexte de haut niveau de complexité. Les raisonneurs utilisent un modèle de logique de premier ordre et un raisonnement basé sur les règles, c'est-à-dire que tous les contextes sont représentés sous la forme d'expression logiques de premier ordre [25]. Ces règles d'inférence prédéfinies sont spécifiées et intégrées dans le système par le programmeur. Cependant, et compte tenu de l'hétérogénéité des diverses sources de données et des applications, le module de raisonnement doit intégrer différents raisonneurs et d'autres mécanismes de raisonnement [26]. Ce module doit supporter et définir l'ensemble des APIs de raisonneurs et permettre d'intégrer et de modifier facilement les raisonneurs.

Autre que l'inférence des informations du contexte le raisonneur a la tâche de vérifier et maintenir la cohérence des informations du contexte et résoudre les conflits contextuels.

4.2.2.7 La couche de gestion du contexte

C'est la dernière couche du middleware. Elle constitue la passerelle entre les services du contexte et le middleware. Cette couche stocke les informations du contexte et constitue un outil de courtage pour permettre la récupération des ces informations par les services du contexte. Cette couche est aussi reliée au module de correspondance, décrit ci-dessous. La couche de gestion du contexte est composée d'interfaces qui sont capables de faire la liaison entre l'information du contexte et l'application correspondante.

4.2.2.8 Le module de correspondance de contexte

Ce module (aussi appelé *Context Matcher*) est responsable de la localisation et de la gestion des sources du contexte. Il utilise un mécanisme de correspondance pour trouver les sources du contexte dont le service a besoin. Pour cet effet, ce module est connecté à la couche de collection. Il sert à localiser les capteurs appropriés. Le module de correspondance interagit aussi avec la couche de gestion du contexte qui lui fournit les informations du contexte qu'elle pourvoit.

4.2.3 Les services du contexte

Ce sont les différents services qui sont dépendants du contexte. Ces services, qui peuvent être aussi des applications, ont besoin de s'approvisionner en informations du contexte afin qu'ils s'adaptent à la situation actuelle de l'utilisateur et son environnement. Les serveurs demandent les informations du contexte qui leur sont nécessaires par le biais du *Context matcher*. Ce dernier, trouve les capteurs correspondants et communique les informations du service de contexte en question à la couche de gestion, qui à son tour fournit l'information de contexte de haut niveau à ce service de contexte. Dans notre projet, nous travaillons une application mobile. Cette application serait en mesure de réagir aux informations de contexte fournies par la couche de gestion.

4.3 Plan de validation de l'architecture

Nous proposons le plan suivant pour la validation de l'architecture présentée ci-dessus :

- Implémentation du middleware et de ses différents modules (dans un environnement JAVA par exemple)

- Développer un prototype de réseau de capteurs corporels (*body sensors network*) incluant ceux évoqués dans notre mémoire en l'occurrence : thermomètre, glucomètre, tensiomètre, oximétre.

- Développer un prototype d'application mobile (sur environnement Androïde par exemple) qui fournit les interfaces nécessaires pour la communication avec l'utilisateur, le réseau de capteurs et le middleware). Cette application offre aussi la connexion sur le réseau Facebook en tant que capteur virtuel permettant la collecte des informations de contexte sur ce réseau d'une manière dynamique (la configuration de l'application inclut une interface permettant à l'utilisateur de connecter cette application à son compte Facebook personnel).

- Développer des services de contexte utilisant les informations fournies par le middelware.

- Intégrer les ontologies et les règles d'inférence dans le middelware en permettant l'interaction avec le module de traitement et le module de gestion du contexte.

- Effectuer des simulations dans différentes situations (contextes) et évaluer les performances de notre architecture (les performances incluent le temps d'exécution, le raisonnement sur le contexte, la détermination du contexte et les scénarios de réaction selon le contexte).

CHAPITRE V

MODÉLISATION DE L'ONTOLOGIE FACEBOOK

5.1 Introduction

D'un point de vue centré sur l'utilisateur, le contexte social est l'environnement de cet utilisateur, c'est-à-dire tout ce qui l'entoure et qui influence son comportement ou ses préférences [27]. Le contexte social d'un utilisateur peut inclure ses relations et contacts, son humeur, ses préférences, ses habitudes et plus encore. Le contexte social est alors très important pour définir le contexte d'un utilisateur.

Nous avons déjà souligné que certains modèles de contexte tels que *CONON* [15]., *CobraOnt* [10], *SmartM* [20] ont négligé le contexte social de l'utilisateur qui devrait être une entité à part entière du modèle du contexte. Les systèmes dépendants du contexte doivent tenir compte du contexte social d'un utilisateur afin d'améliorer ses interactions avec les périphériques.

De ce fait, nous avons travaillé sur cet aspect du contexte et développé un modèle capable de nous fournir des informations sur le contexte social en ligne de l'utilisateur. Ce modèle sera intégré, dans le prochain chapitre, dans un modèle de contexte afin de l'enrichir.

Dans ce chapitre, nous présentons notre réalisation qui consiste en :

- L'étude des fonctionnalités de *Facebook* et leur éventuelle exploitation pour le contexte de l'utilisateur.
- L'analyse de l'architecture du réseau *Facebook*.

- L'extraction des éléments de *Facebook*, c'est-à-dire les objets qui le composent et les relations entre ces objets.
- La conception d'un modèle ontologique représentant ces objets et ces relations. Cette forme nous permettra de faire un lien entre *Facebook* et l'ontologie du domaine médical que nous allons présenter dans le chapitre suivant.
- L'implantation sur Protégé du modèle ontologique obtenu.

5.2 Contexte social et réseaux sociaux en ligne, cas de Facebook

Nous pouvons analyser les réseaux sociaux afin d'y extraire des informations sur le contexte social d'un utilisateur, ses préférences, ainsi que ses données personnelles. Ces informations enrichissent notre modèle de contexte et améliorent les performances des systèmes dépendant du contexte. Nous analysons ci-dessous le réseau *Facebook*.

Le réseau *Facebook*, avec ses 800 000 utilisateurs actifs[8]est aujourd'hui le plus populaire de tous les réseaux sociaux. De ce fait, il est le réseau le plus prometteur à étudier et analyser. Indépendamment des relations entre les utilisateurs, *Facebook* est une source très riche en informations personnelles : nom, prénom, date de naissance, email, adresse, date de naissance, école, employeur, état matrimonial, orientation politique et bien d'autres... La collecte et l'analyse de ces données peut nous amener à cerner le profil d'un utilisateur afin de lui fournir une information qui lui correspond le mieux. Ces techniques sont utilisées pour le marketing mais peuvent aussi être utilisées dans le cadre de la définition du contexte d'un utilisateur.

Nous avons alors analysé ce réseau et modélisé l'ontologie qui représente ces objets. Nous avons utilisé les différentes informations disponibles sur ce réseau social pour y extraire celle qui peuvent être exploitées pour notre modèle de contexte présenté dans le prochain chapitre. Les liens entre les utilisateurs de la plateforme nous permettent de représenter leurs différentes relations sous forme de graphes. Ce modèle graphique formel nous permet de faire le lien entre l'ontologie du domaine médical et les éléments de *Facebook*.

---

[8] http://newsroom.fb.com/content/default.aspx?NewsAreaId=22 visitée le 28/02/2012

Les diverses fonctionnalités de *Facebook* sont une riche source de données sur le contexte et nous fournissent les informations ci-dessous. Nous avons énuméré tous les éléments et non seulement ceux que nous avons utilisé pour permettre leur utilisation dans d'autres travaux de recherche.

5.2.1 Les liens

Les relations en ligne sont importantes puisqu'elles sont souvent basées sur les relations du monde réel [28]. Les relations d'amitié sur *Facebook* sont bilatérales, ce qui nous mène à croire que la représentativité de ces liens est très élevée. De plus, *Facebook* nous permet de différencier le genre de relation entre les utilisateurs en définissant les liens de parenté familiale, les camarades de bureau et les différentes affiliations. Nous aurions pu exploiter cet aspect pour tracer des liens biologiques entre différents patients ou pour la réaction aux cas d'urgences médicales.

5.2.2 Les événements

Facebook permet de planifier la participation à des événements en y incluant l'adresse de la place, l'heure de début et l'heure de fin de chaque événement. *Facebook* permet aussi de dresser une liste des personnes qui participent à un même événement. Ceci nous permet d'anticiper sur le contexte de l'utilisateur. Nous avons utilisé cet élément pour déduire l'emplacement et l'occupation de l'utilisateur.

5.2.3 Les préférences

En plus des informations personnelles comme équipe favorite, athlète favori, religion, orientation politique, la fonctionnalité *like* de *Facebook* permet d'avoir des informations sur les préférences de l'utilisateur comme ses artistes favoris, ses restaurants ou clubs favoris, ses marques de vêtement préférées.

Facebook nous permet ainsi d'avoir une meilleure idée sur les préférences et les intentions de l'utilisateur [29].

5.2.4 Les emplacements

La fonction *Checkin* permet à un utilisateur de marquer son emplacement sur son profil *Facebook*. Cette fonction définit la localisation de l'utilisateur, la date et l'heure de la visite ainsi que les personnes qui se trouvent dans le même endroit et les liens entre elles. Cet aspect est exploité pour définir l'emplacement du patient et sa proximité par rapport à d'autres personnes de son réseau.

5.2.5 La présence en ligne

Le module de chat peut nous permettre de détecter la présence en ligne d'un utilisateur à partir de la messagerie instantanée. Nous pouvons par la suite déduire l'occupation de l'utilisateur et sa disponibilité. La présence en ligne a fait l'objet de quelques recherches comme le projet *Online Presence* [30] qui modélise une ontologie afin de mettre en place un modèle commun de présence en ligne sur tous les réseaux sociaux.

5.2.6 La personnalité

Analyser la personnalité de l'utilisateur est essentiel dans le modèle d'utilisateur [31]. La manière d'utiliser des réseaux sociaux reflète la personnalité dans le monde réel. En se basant sur *Facebook* et en analysant l'interaction de l'utilisateur avec ce réseau social, comme le nombre d'amis, le nombre de publications sur son mur, nous pouvons inférer sa personnalité. Ce paramètre pourrait être utilisé dans le cadre d'une application médicale relative à la psychologie.

5.2.7 Le marquage

Le marquage des photos sur *Facebook* nous permet de définir les liens entre les utilisateurs tout en fournissant le lieu et la date de l'événement. Ces liens peuvent nous aider à capter des informations sur les personnes dans l'environnement social de l'utilisateur dans la vie réelle. Cet élément a aussi été utilisé pour avoir l'emplacement de l'utilisateur.

5.3 Le graphe Facebook

Les relations entre utilisateurs peuvent être représentées sous formes de graphes dont les nœuds sont les utilisateurs et les arcs les relations qui les relient. *Facebook* ne connecte pas seulement les utilisateurs mais connecte aussi des objets et des modules extérieurs à ces personnes. Ces connexions sont modélisées par le Graphe Social (*Social Graph*) [9] et le protocole *Open Graph*[10].

5.3.1 Le protocole *Open Graph*

Le Protocole Open Graphe permet à *Facebook* d'intégrer des objets extérieurs, d'avoir la même fonctionnalité que les objets du site et de le connecter au graphe (figure 4.1). Cette fonctionnalité a permis à *Facebook* de connecter des millions d'objets sous une seule plateforme. Le Open Graph représente les connections de *Facebook* aux objets extérieurs du site sous la forme d'un graphe comportant des millions de nœuds.

[9] https://developers.facebook.com/socialdesign/
[10] https://developers.facebook.com/docs/opengraph/

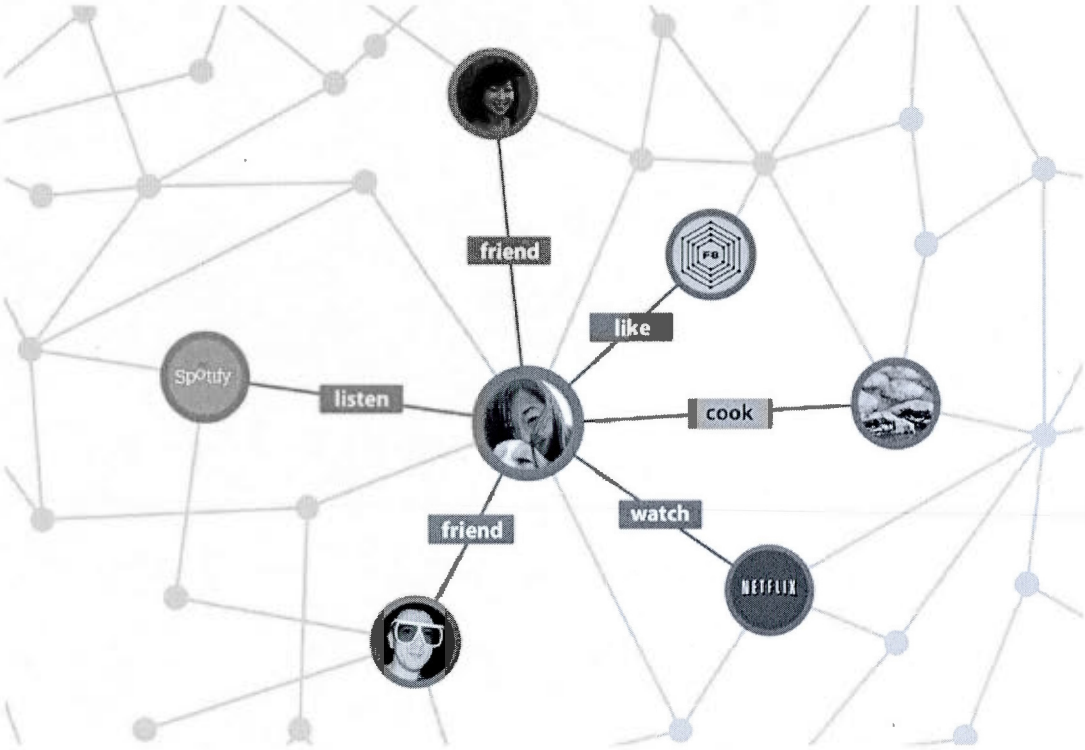

Figure 5.1 : Open Graph de Facebook.

(Source http://developers.facebook.com/docs/opengraph/)

5.3.2 Le Graphe Social

Le graphe social (Figure 5.2) est le noyau de la plateforme *Facebook*, réseau social en ligne. Les connexions entre les utilisateurs sont l'élément le plus important du réseau. Ils permettent de représenter les relations sous entre les objets sous formes de graphes. Les objets sont les utilisateurs, les événements, les pages, les photos, les groupes, les applications, etc. Les connexions sont les liens d'amitié entre les utilisateurs, les marquages, les inscriptions, etc.

Les personnes (profils) de *Facebook* sont interconnectées. Ces éléments permettent d'avoir une meilleure vue du modèle utilisateur et l'exploitation de ses données nous offre la possibilité d'en extraire des informations pertinentes de son contexte. Il serait alors

intéressant de modéliser le graphe social de *Facebook* afin de le rendre profitable pour les systèmes dépendants du contexte.

Figure 5.2 : Graphe Social des réseaux d'amis de Facebook généré par *touchgraphe.net.*[11]

[11] http://touchgraph.net

Nous avons utilisé un générateur de graphes en ligne Touchgraphe.net pour l'affichage du graphe qui représente les relations en ligne d'un utilisateur donné. Ce graphe pourrait être exploité pour des modèles de contexte qui prennent en compte les relations d'un utilisateur. Nous n'avons pas pris en considération cet aspect dans notre modèle. Cependant, cela pourrait être réalisé dans d'autres champs d'applications.

5.4 L'ontologie de Facebook

Après avoir opté pour le modèle de contexte basé sur les ontologies, nous avons alors représenté le modèle de *Facebook* sous forme ontologique. Dans notre réalisation, nous nous concentrerons sur les objets internes du réseau social *Facebook*. Nous définirons un modèle global pour représenter les différents objets et les relations qui les lient. L'ontologie *Facebook* jouera un rôle important dans la définition du model de contexte. Pour le langage et l'outil d'édition des ontologies, nous avons utilisé respectivement OWL et Protégé.

5.4.1 Modélisation de l'ontologie

Facebook nous permet de modéliser son architecture par deux approches différentes :

- La première consiste à déréférencer les URIs du graphe de Facebook à partir d'un URL d'un compte utilisateur.
- La deuxième consiste à utiliser le Graph API de Facebook afin de concevoir le modèle ontologique de ce réseau social. L'API est accessible à l'adresse https://developers.facebook.com/docs/reference/api et décrit tous les objets de *Facebook* ainsi que les connexions qui existent entre eux. Le Graph API nous permet non seulement de définir les objets et les relations mais aussi les propriétés de chaque objet.

Nous avons opté pour la deuxième méthode pour sa simplicité et la précision des informations qui y sont énoncées. Nous avons eu recours à la première méthode pour des fins de vérification de l'exactitude et de la cohérence des informations dégagées.

L'analyse du Graph API de *Facebook* nous permet de dégager les objets (classes) et leurs propriétés, et leurs connexions que nous énumérons ci-dessous. Nous avons, par la suite, conçu un modèle relatif à ces éléments sous forme d'ontologie. Finalement nous avons implémenté l'ontologie sur Protégé. Nous avons joint des captures d'écrans de cette implémentation pour chaque élément.

5.4.1.1 Les objets de Facebook

Nous nous sommes basé sur la description des objets fournie par Facebook à partir de l'adresse mentionnée ci-dessus. Nous avons parcouru chaque objet et dégagé ses propriétés décrites par des tableaux incluant leurs propriétés, leur description et les permissions qui y sont affectées. Facebook fournit aussi les connexions relatives à chaque objet.

Les objets de Facebook, dégagés après notre analyse, sont représentés dans le Tableau 5.1.

Table 5.1 : Objets de Facebook.

Objet	**Nom**	**Description**
Utilisateur	*User*	Élément central de la structure de Facebook.
Compte	*Account*	Pages et applications Facebook appartenant à l'utilisateur.
Page	*Page*	Page des *fans*.
Albums photo	*Album*	Albums regroupant différentes photos d'un utilisateur.
Applications	*Application*	Applications développées sur Facebook.
Marquage de localisation	*Check-In*	Inscription de la localisation courante de l'utilisateur sur son mur.
Événement	*Event*	Événements programmés.
Fil d'actualité	*Home*	Page principale de la plateforme à travers laquelle l'utilisateur peut visualiser les activités de ses amis.
Listes d'amis	*Friendlists*	Listes d'amis crées par l'utilisateur.
Demandes d'ajout	*Friendrequest*	Demande d'ajout envoyé par un utilisateur à un autre.
Groupes	*Group*	Groupes de personnes partageant un intérêt commun.
Mur	*Feed*	Mur personnel d'un utilisateur où on peut trouver les inscriptions qui lui ont été destinées et les inscriptions qu'il soumet.
Boîte de réception	*Inbox*	Boîte de réception de la messagerie de Facebbok.
Boîte d'envoi	*Outbox*	Boîte d'envoie de la messagerie de Facebook.
Note	*Note*	Articles personnels d'un utilisateur.
Notification	*Notification*	Notifications que l'utilisateur reçoit de la part de la plateforme d'une activité dans laquelle il est impliqué.
Inscription	*Post*	Inscriptions d'un utilisateur sur son mur ou le mur d'un ami. Une inscription peut être une photo, une vidéo, un statut (message texte) ou un lien.
Photo	*Photo*	Photo Facebook (enregistrée sur la plateforme) postée par un utilisateur.
Vidéo	*Video*	Vidéo Facebook (enregistrée sur la plateforme) postée par un utilisateur
Statut	*Status*	Message texte posté par utilisateur.
Lien	*Link*	Lien vers un site ou objet externe de Facebook posté par un utilisateur.

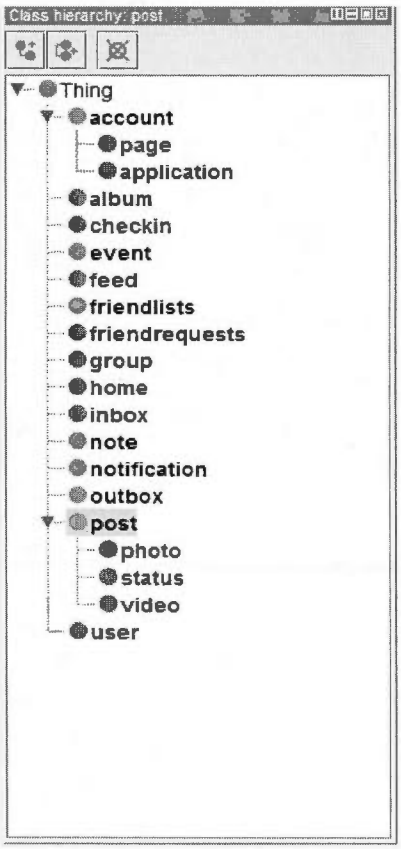

Figure 5.3 : Capture écran des classes de Facebook sous Protégé.

Les classes de l'ontologie que nous avons développée avec Protégé sont représentées dans la figure 5.3. Nous pouvons voir la hiérarchie de ces objets à travers cette implantation des classes sur Protégé. Ainsi *Page* et *Application* sont des sous classes de *Account* . Une publication (*post*) est soit une photo (*photo*), soit une vidéo (*video*), soit un statut (*status*).

5.4.1.2 Les propriétés des objets

Notre analyse de *Facebook* nous a permis de dégager les propriétés des objets. Ces propriétés sont représentées dans le tableau 4.2.

Table 5.2 : Propriétés des objets de Facebook.

Objet	Propriétés
Utilisateur	id, description, lien personnel, localisation, nom, prénom, intimité (*privacy*), type, catégorie, biographie, date de naissance, sexe, nom d'utilisateur, adresse courriel, éducation, ville d'origine, intéressé par, citations, orientation politique, religion, langue.
Page	id, nom, catégorie, lien, administrateur, localisation, téléphone.
Album photo	id, description, lien, localisation, nom, intimité (*privacy*), type.
Application	id, catégorie, compagnie, nom, description.
Marquage de localisation	id, localisation, marquages, date de création
Événement	id, nom, description, localisation, créateurs, visibilité, date et heure de début, date et heure de fin.
Listes d'amis	id, nom, type.
Groupe	id, nom, description, lien, version, intimité.
Note	id, message, sujet, date de création.
Photo	id, nom, source, date de création, hauteur, largeur, marquqges, date de mise en ligne.
Vidéo	id, nom, description, date de création, date de mise en jour, marquages.
Statut	id, message, date de création, marquages.
Lien	id, URL, date de création, message.

5.4.1.3 Les relations

Les relations (*Object Properties*) entre les objets de *Facebook* sont représentées dans le tableau 5.3.

Table 5.3 : Relations entre les objets de Facebook.

Propriété d'objet	**Domaine**	**Rang**
submits	User	*Post*
isPartOf	User	*Group*
ownsAlbum	User	*Album*
attends	User	*Event*
placedAt	User	*Checkin*
sends	User	*Friendrequests*
hasAccount	User	*Account*
hasFeed	User	*Feed*
hasHome	User	*Home*
hasInbox	User	*Inbox*
hasOutbox	User	*Outbox*
hasLists	User	*Friendlists*
hasNote	User	*Note*
notifiedBy	User	*Notification*
shares	Page	*Post*
processesHome	Page	*Home*
processesFeed	Page	*Feed*
creates	Page	*Events*
includes	Groupe	*Feed*
submitAppPost	Application	*Post*
hasAppHome	Application	*Home*
hasAppFeed	Application	*Feed*
hasWall	Event	*Feed*

Notre implantation de ces relations sur Protégé est représentée dans la figure 5.4.

Figure 5.4 : Capture d'écran Les relations sous Protégé.

5.4.1.4 L'ontologie

Finalement, l'analyse des objets, de leurs propriétés ainsi que les liens entre eux nous a permis de modéliser l'ontologie présentée dans la figure 5.5. Cette figure est une reproduction du modèle ontologique obtenu par Protégé. Nous avons repris ce modèle manuellement puisque l'outil de visualisation graphique sur Protégé a une visibilité limitée qui ne permet pas de voir les relations entre les objets.

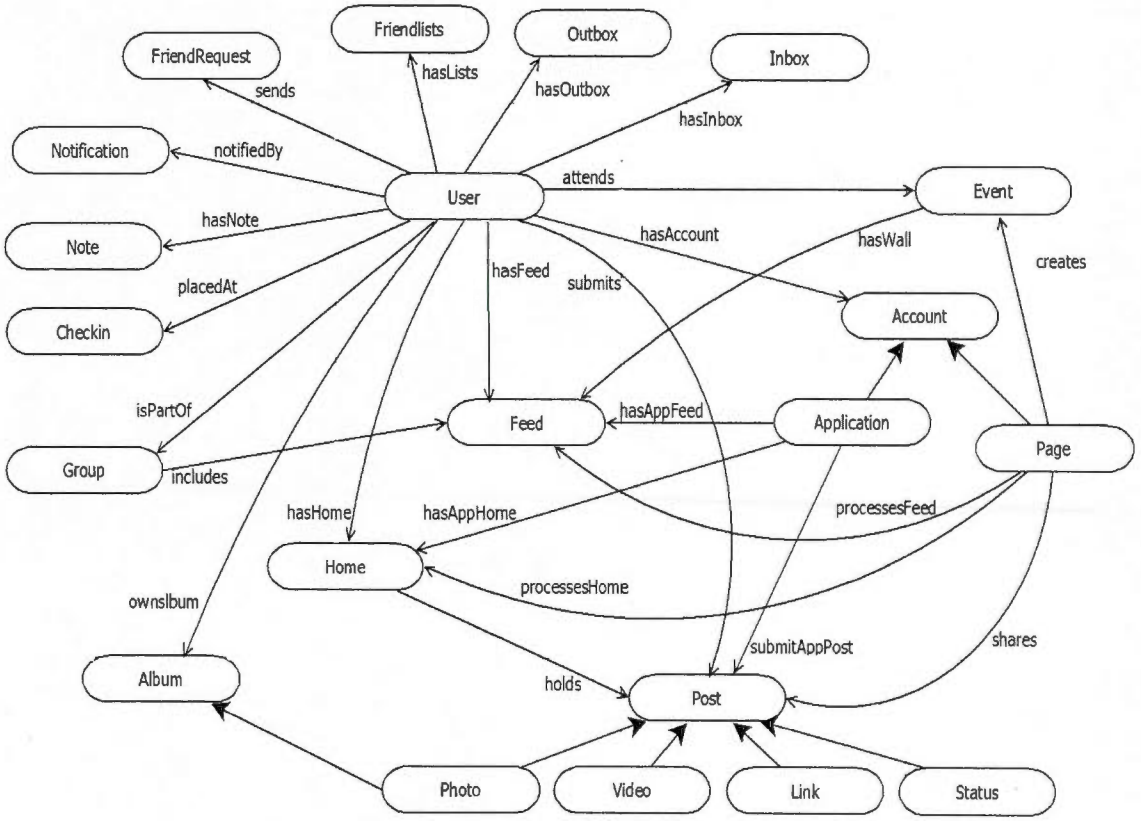

Figure 5.5 : Ontologie de Facebook.

Nous pouvons remarquer que l'utilisateur est l'élément principal de la structure de *Facebook* et que tous les autres objets y sont connectés. Nous n'avons pas mis les propriétés des différentes classes pour alléger le schéma.

CHAPITRE VI

MODÉLISATION DU CONTEXTE

6.1 Introduction

Après avoir modélisé la structure de *Facebook* sous forme d'ontologie, nous allons dans ce chapitre, expliquer notre technique pour la conception d'un modèle de contexte général. Ce modèle inclut l'intégration de l'ontologie de *Facebook* et l'ontologie du domaine médical.

La modélisation du contexte est *« la spécification des entités et les relations entre ces entités qui sont nécessaires à la description du contexte »* [Notre traduction] [32]. Le modèle de contexte couvre les besoins fonctionnels du système.

Nous avons modélisé une ontologie du contexte. Nous avons fait l'intégration de cette ontologie avec celle développée dans le chapitre précédant pour le réseau social *Facebook*. Nous avons implémenté l'ontologie résultante avec Protégé et nous y avons appliqué différentes règles d'inférence.

Ce chapitre décrit notre approche de la construction de l'ontologie expliquée étape par étape. Nous présentons les règles d'inférence applicables à notre modèle et nous avons effectuons quelques simulations pour tester le bon fonctionnement de ces règles.

6.2 Conception de l'ontologie du contexte

Dans le chapitre 5, nous avons justifié le choix de l'ontologie pour modéliser le contexte. Le développement des ontologies obéit à une certaine méthodologie qui définit les étapes et

les règles à suivre pendant le processus [8]. Pour une bonne modélisation, il faudrait aussi déterminer le volume de l'ontologie et son niveau de spécificité.

Pour l'approche ontologique, nous pouvons utiliser des ontologies, entières ou partielles, développées précédemment. Les ontologies de haut niveau (*upper ontologies*) sont celles qui sont le plus souvent utilisées puisqu'elles offrent des concepts universels, indépendants du domaine et qui sont flexibles à une adaptation pour différentes applications qui ne réalisent pas forcément les mêmes traitements et n'appartiennent pas au même domaine.

Nous travaillons sur la modélisation du contexte pour une simple application du domaine médical qui utilise des informations sur l'état physiologique de l'utilisateur et capable de réagir en fonction de cet état et d'autres éléments qui caractérisent le contexte de l'utilisateur. Nous utilisons l'ontologie de Facebook pour enrichir notre modèle de contexte avec des données triées de cette plateforme.

Nous avons décidé d'utiliser l'ontologie CONON comme ontologie de haut niveau pour notre modèle. Nous étendrons cette ontologie avec les concepts relatifs au domaine médical. Nous intégrerons l'ontologie obtenue avec celle que nous avons développée précédemment pour modéliser les éléments de *Facebook* afin d'avoir un modèle global capable de supporter les règles d'inférence. Cette approche est représentée dans la figure 6.1.

Nous partons alors de l'ontologie de haut niveau CONON (*O1*) comme point de départ. On y ajoute les concepts du domaine médical pour obtenir une ontologie (*O2*). Cette ontologie est couplée avec l'ontologie de Facebook (*O3*) afin d'obtenir notre modèle de contexte (*O4*). Ce modèle permet la déduction du contexte au moyen des règles d'inférences.

Nous avons réalisé cette ontologie en plusieurs étapes.

6.2.1 Étape A: Ontologie de haut niveau (*O1*)

L'ontologie de haut niveau CONON est un modèle générique qui présente un ensemble d'entités de haut niveau et qui offre une forte extensibilité et capacité d'adaptation à différents domaines spécifiques.

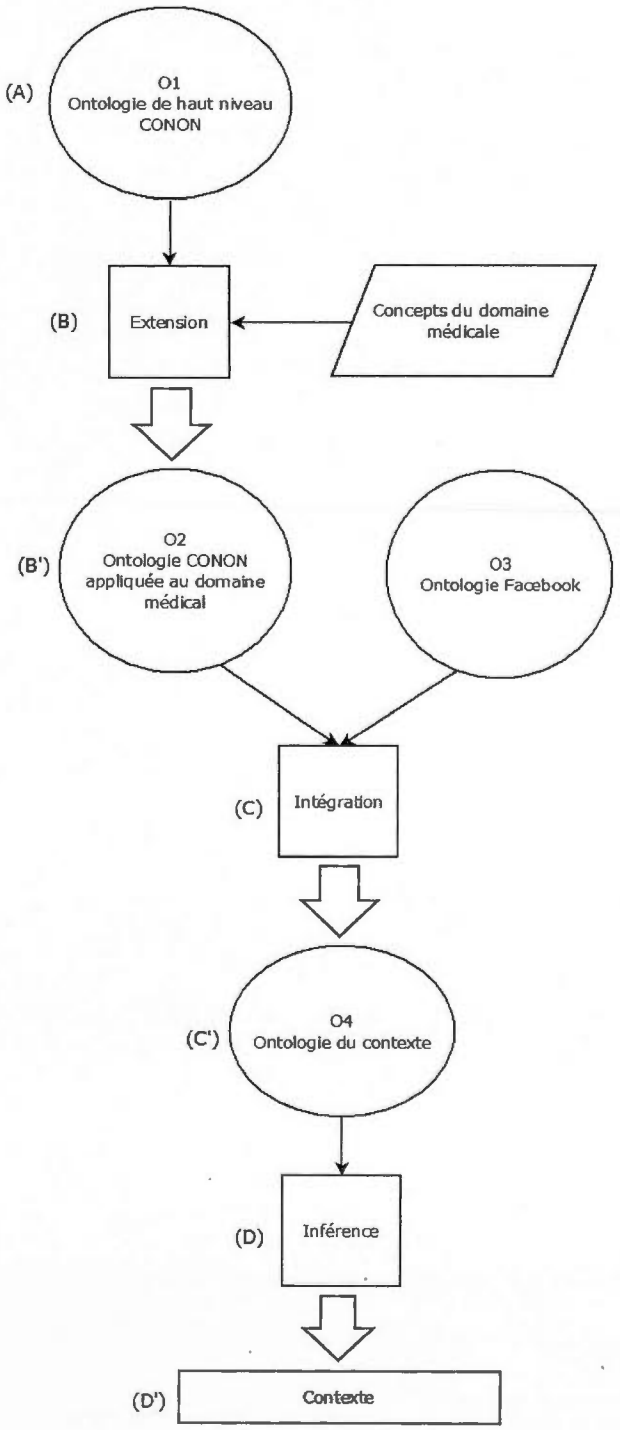

Figure 6.1 : Approche de modélisation du contexte.

L'ontologie représentée par la figure 6.2 est basée sur les concepts suivants : *Personne, Activité, Entité Computationnelle et Localisation*. Ces concepts sont reliés entre eux par des relations (*Object Properties*). Une activité se déroule (*locatedIn*) dans une localisation et utilise une entité computationnelle. Une personne est propriétaire (*ownsCompEnt*) d'une entité computationnelle, est engagée (*engagedIn*) dans une activité, est située (*situatedIn*) dans une localisation, et possède (*ownCompEnt*) une entité computationnelle.

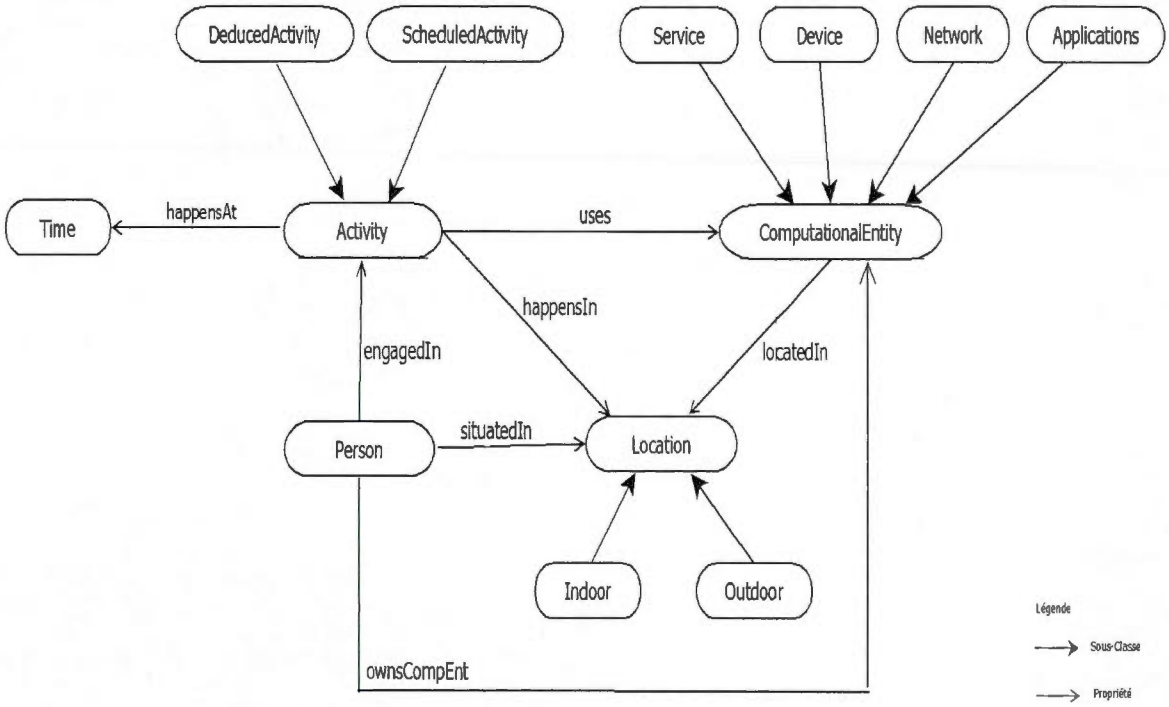

Figure 6.2 : Ontologie du haut niveau se basant sur CONON.

Pour des besoins d'application, nous avons ajouté un nouveau concept générique (*Time*) pour fournir la date et l'heure. Ce concept est lié au concept (*Activity*) par la relation se *déroule à* (*happensAt*).

Les concepts généraux de l'ontologie contiennent des sous entités ou sous classes qui présentent des extensions de plus haute spécificité et facilitent l'intégration des concepts propres au domaine. OWL permet la structuration des éléments de manière hiérarchique par la notion *subClasseOf*. Ainsi, la localisation est divisée en espace intérieur (*Indoor*) et

extérieur (*Outdoor*). L'élément *Activité* (*Activity*) distingue entre les activités programmées (*SchedualedActivity)* et activités déduites *(DeducedActivity).* L'entité computationnelle (*ComputationalEntity*) peut être un périphérique (*Device*), un réseau (*Network*), un service (*Service*) ou une application (*Applications*).

6.2.2 Étape B: Extension de l'ontologie de haut niveau par les concepts du domaine

Nous avons aussi choisi de concevoir notre modèle de contexte à une application médicale simple. L'étape suivante est donc de développer notre modèle de contexte en étendant l'ontologie de haut niveau par des concepts relatifs au domaine médical. De plus, nous devons prendre en considération dans notre modélisation les règles d'inférence qu'on souhaiterait exécuter.

Dans notre approche de modélisation nous avons pris en considération deux paramètres importants pour cette procédure. Le raisonnement nécessite d'importantes ressources et il est préférable de minimiser le plus possible ces délais de traitement [33]. Le domaine médical contient des terminologies spécifiques que nous avons respectées dans notre modélisation. L'utilisation des terminologies médicales spécifiques est un excellent point de départ pour la construction des ontologies [34]. Pour cela, nous avons utilisé un thésaurus de vocabulaire médical, *Medical Subject Headings* (*Mesh*)[12], de la bibliothèque nationale américaine de médecine. *Mesh* fournit un ensemble de termes médicaux et leur description, tout en les organisant d'une manière hiérarchique.

La meilleure façon de concevoir des ontologies serait une approche hybride qui combine les deux approches classiques de modélisation des ontologies (approche ascendante et approche descendante) [34]. Nous avons opté pour cette méthode dans notre approche de conception de l'ontologie du contexte.

[12] http://www.nlm.nih.gov/mesh/

Ainsi, nous avons déterminé les données corporelles à capter : la température corporelle, le pouls, la glycémie, la tension artérielle, etc. Nous les avons classées sous un super-classe *PhysiologicalState* qui pourrait contenir d'autres catégories.

La hiérarchisation du concept *PhysiologicalState* et ses différentes sous-classes est représentée dans la Figure 6.3. Pour chaque état physiologique, nous avons défini des sous-classes. Pour la glycémie, il pourrait y avoir une glycémie normale, une hypoglycémie, une hypoglycémie sévère, une hyperglycémie ou une hyperglycémie sévère. Pour la température corporelle, elle peut être normale, élevée ou très élevée. Pour la tension, on peut définir les états de tension normale, pré-hypertension, hypertension, hypertension sévère, hypotension et hypotension sévère. Les états sévères nécessitent une réaction d'urgence.

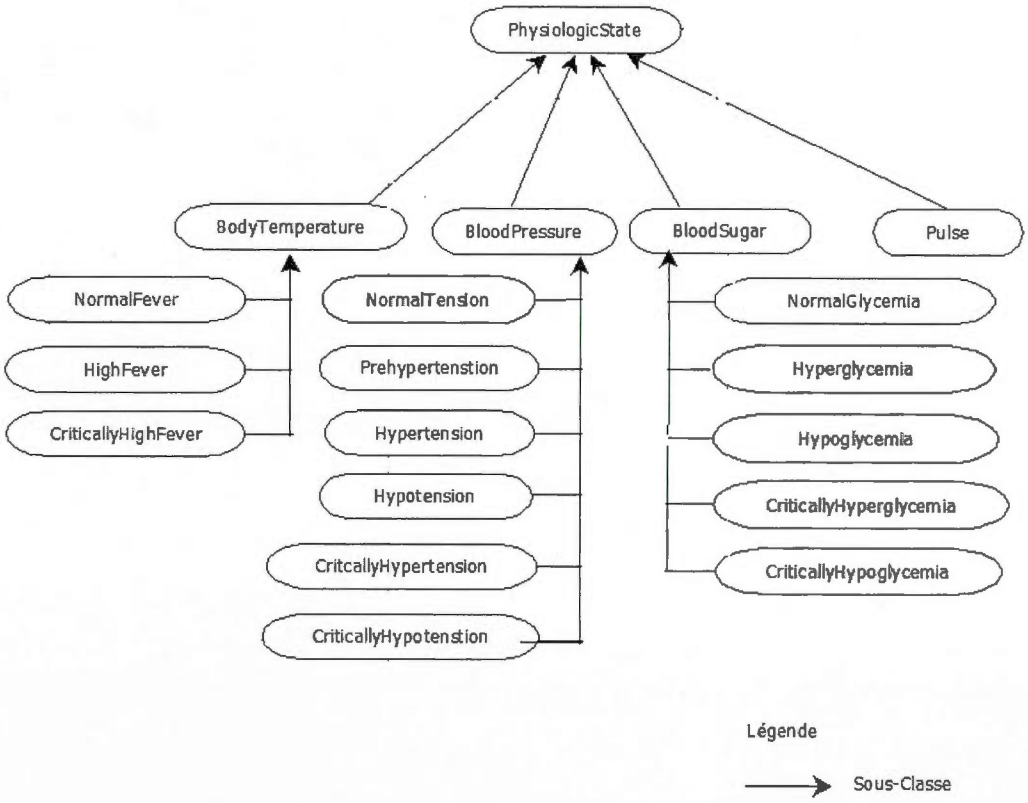

Figure 6.3 : Classification des différents états physiologiques.

Nous avons aussi organisé le concept *Device* en catégorisant les appareils par type : les appareils de communication (téléphone mobile) et appareils médicaux (tensiomètre, thermomètre, glucomètre, oxymètre, etc.) qui agissent comme capteur pour la mesure des données corporelles. Cette classification est représentée par la Figure 6.4.

Cependant, les valeurs définissant le domaine des valeurs de chacun des états physiques varient d'un genre à un autre (homme, femme), d'une catégorie d'âge à une autre (enfant, adulte, personne âgée) et de l'état de santé de l'utilisateur (personne normale, personne diabétique, etc.). Nous avons alors défini ces concepts qui sont reliés à la classe *Person*.

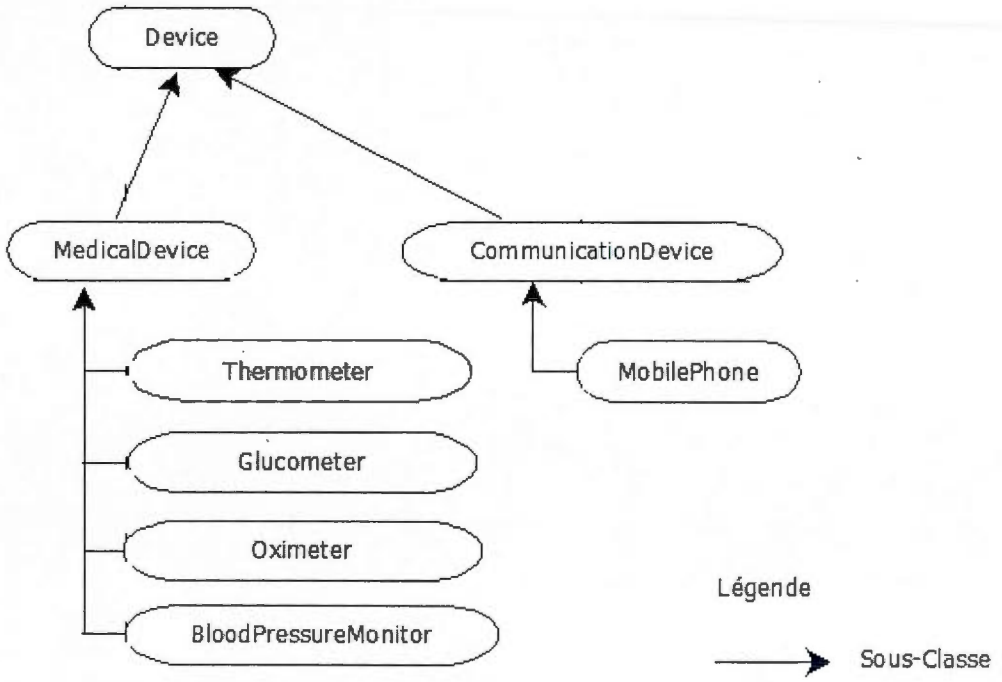

Figure 6.4 : Classification des appareils.

Les relations qui existent entre ces concepts du domaine ont été modélisées. Un état physique est capturé par (*assessedBy*) un appareil médical. Une personne a une catégorie d'âge et un genre et un état physiologique.

La figure 6.5 montre l'ontologie résultante (B') de l'adaptation de l'ontologie de haut niveau à nos besoins pour une simple application médicale.

6.2.3 Étape C: Couplage de l'ontologie du contexte et de l'ontologie Facebook

Nous devons coupler l'ontologie, précédemment développées pour *Facebook*, avec notre ontologie du contexte médical.

Le couplage entre deux ontologies veut dire « *trouver les relations qui existent entre ces deux ontologies*» [Notre traduction] [35] de façon à ce qu'elles représentent une ontologie unique et cohérente. Ce couplage pourrait être une fusion, un alignement ou une intégration [36]. Dans notre cas, il s'agit de l'intégration qui est le procédé de génération d'une ontologie unique dans un seul sujet à partir de deux ou plusieurs ontologies de sujets différents.

Nous avons fait cette intégration d'une façon manuelle. Les deux ontologies à unifier sont assez simples et ne nécessitent pas le recours à des méthodes d'intégration automatique ou semi-automatiques. L'ontologie résultante (C') est tracée par la figure 6.6.

Nous avons ajouté une superclasse qui regroupe tous les profils en ligne d'un utilisateur, *OnlineSocialContext* qui est reliée *à Person*. Le choix d'ajouter cette entité a pour but de faciliter n'importe quelle extension future par d'autres réseaux en ligne.

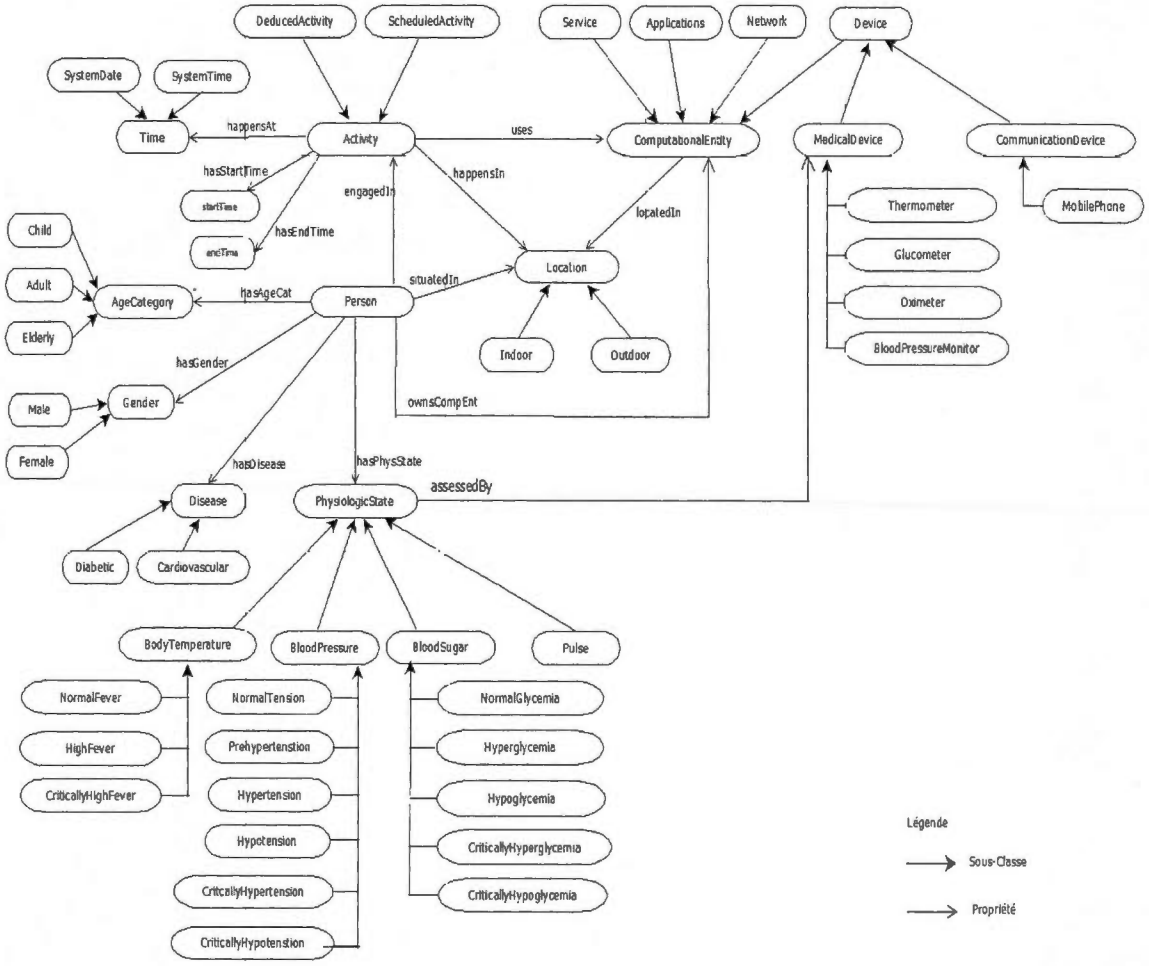

Figure 6.5: Ontologie du domaine.

Nous avons comparé chaque concept des deux ontologies en vue de trouver des correspondances potentielles. Par exemple, la classe *Event* de l'ontologie Facebook (*O3*) peut être une sous classe de *ScheduledActivity*de l'ontologie du domaine médical *(O2)* (un événement qui est confirmé sur *Facebook* est une activité programmée). Les classes *Checkin (O3)* et *Location (O2)* sont aussi liées par la relation *ProceedsAt*.

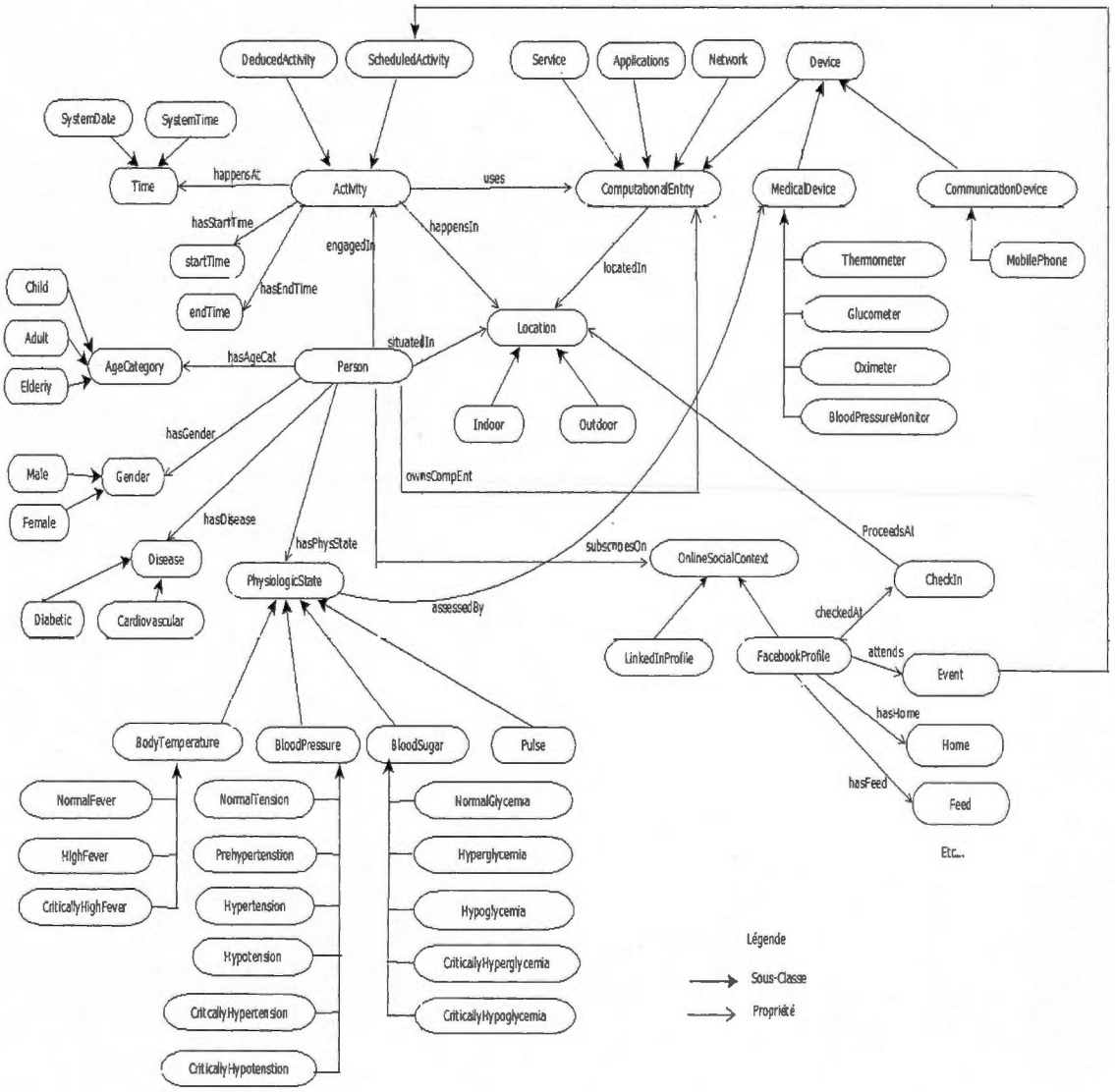

Figure 6.6 : Ontologie du contexte.

Par ailleurs, certaines propriétés du profil Facebook (*FacebookProfil*) pourraient être les mêmes propriétés de la classe *Person* (adresse, nom, prénom, anniversaire, etc.). Nous avons choisi de les laisser sous une seule classe afin d'éviter une redondance dans l'ontologie unique résultante (*O4*).

6.2.4 Étape D: Raisonnement et inférence du contexte

Le choix de l'approche ontologique dans la modélisation du contexte était motivé par la possibilité d'appliquer du raisonnement sur le modèle. En effet, le raisonneur peut jouer le rôle de classificateur en inférant la hiérarchie des classes. Le raisonneur est responsable de vérifier la cohérence de l'ontologie. De plus, le raisonneur aide à déduire (inférer) un contexte de haut niveau à partir d'un ensemble de données brutes captées (informations de bas niveau). Par exemple, pour déduire l'état d'hyperglycémie pour une personne, il faut qu'un ensemble de conditions soient vérifiées. Ces conditions sont définies par un ensemble de données captées (âge, genre, taux de glycémie, etc.).

Le raisonnement est important pour une bonne exploitation de l'ontologie et peut être fait sur deux niveaux ou de deux manières différentes, on parle du raisonnement ontologique et des règles d'inférence définies par le développeur. Le raisonnement ontologique est exécuté par des règles intégrées dans la sémantique d'OWL. Les règles d'inférence sont un ensemble de règles définies par le développeur afin de déduire le contexte de haut niveau.

6.3 Raisonnement ontologique

Les ontologies développées en OWL intègrent des règles logiques internes qui sont définies par la sémantique du langage. Ce raisonnement permet de déduire certains faits relatifs à la hiérarchie de l'ontologie : vérifier si une classe est une sous classe d'une autre classe ou si un individu (une instance) placé dans une classe appartient à d'autres classes en même temps.

Par exemple, *Thermometer* et une sous classe de *ComputationalEntity* puisque *MedicalDevice* est une sous classe de *ComputationalEntity* et *Thermometer* est une sous classe de *MedicalDevice* (*SubclasseOf*).

OWL permet aussi de dissocier les classes (*disjointWith*). Par exemple les classes *Child* et *Adult et Elderly* sont toutes disjointes. Une personne peut seulement être dans une et une seule catégorie d'âge.

La propriété *locatedIn* entre *ComputationalEntity* et *Location* est aussi une propriété inverse (*InverseOf*) c'est-à-dire un mobile est situé dans une maison, la maison contient le téléphone mobile [37].

Dans la version de *Protégé* que nous avons utilisé (4.1.0), il est possible de voir la nouvelle hiérarchie de classes après avoir exécuté le raisonneur. La hiérarchie initiale est appelée *assertedhierarchy* et la hiérarchie inférée est appelée *inferredhierarchy.*

L'inférence du contexte est la déduction d'un contexte implicite à partir de données implicites. Il s'agit de définir le contexte de haut niveau à partir de données brutes captés dites données atomiques. Cette inférence peut être réalisée grâce aux règles d'inférence qui peuvent être associées à l'ontologie et exécutées par le raisonneur. Ces règles sont définies par le programmeur après avoir modélisé le contexte et terminé l'implémentation de l'ontologie.

La définition de ces règles nécessite le choix d'un langage de règles, d'un outil d'édition des règles et d'un raisonneur. Pour définir ces règles nous avons utilisé le langage SWRL[13] et l'éditeur de règles SWRL fourni par Protégé ainsi que *RacerProTG* qui est une version gratuite *RacerPro*[14],

Les règles de la logique descriptive peuvent être [38]:

a) *Des règles déductives* : permettent de déduire un fait à partir d'un ensemble de conditions.
b) *Des règles réactives* : permettent d'exécuter une action si un ensemble de conditions sont vérifiées.

Les valeurs normales des paramètres physiologiques dépendent d'une catégorie à une autre. Nous avons alors classé les individus (*Person*) par catégorie d'âge (*Child, Adult, Elderly*). Les règles permettant de classifier une personne sont :

- Si une personne a moins de 16 ans alors c'est un enfant (Child).

[13] http://www.w3.org/Submission/SWRL/
[14] http://www.racer-systems.com/

- Si une personne a entre 16 et 66 ans alors c'est un adule (Adult).
- Si une personne a plus de 66 ans alors c'est une vielle personne (Elderly).

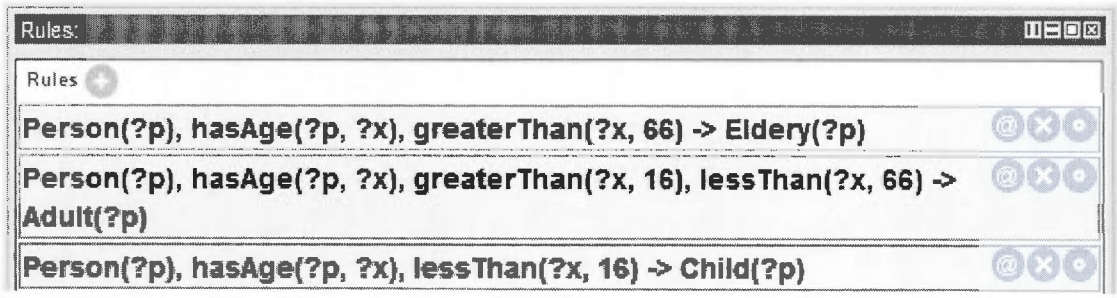

Figure 6.7 : Classification des personnes selon l'âge.

Cette classification se fait par, *greaterThan* et *lessThan* qui sont des fonctions intégrées (*Built-Ins*) de SWRL qui permettent d'effectuer des comparaisons numériques. La figure 6.7 est une capture d'écran, dans Protégé, des règles définies dans le but de classer les individus (*Person*) par catégorie d'âge (*Child, Adult, Elderly*).

La température corporelle ne dépend pas du genre ou de la catégorie d'âge de la personne. Une température corporelle normale (*NormalFever*), élevée (*HighFever*) ou très élevée (*CriticallyHighFever*) est déduite de la manière suivante:

- Si une personne a une température corporelle plus élevée que 39 degrés alors c'est une température très élevée.
- Si une personne a une température entre 38 et 39 degrés alors c'est une température élevée.
- Si une personne a une température corporelle moins élevée que 38 degrés alors c'est une température normale.

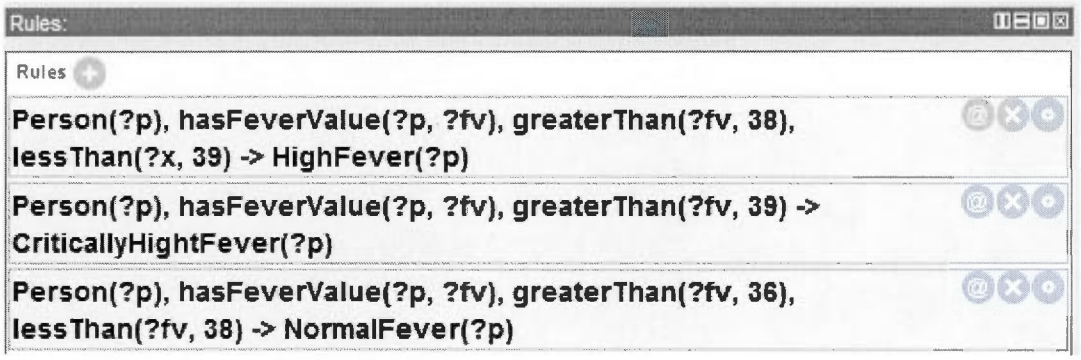

Figure 6.8 : Règles définissant les niveaux de température corporelle de l'utilisateur.

La figure 6.8 est une capture d'écran des règles qui permettent de déduire l'état physiologique de la personne afin de réagir en fonction de ce contexte.

Le taux de glycémie varie selon l'état de santé de la personne (personne normale, personne diabétique), le genre, la catégorie d'âge et l'horaire de prise de sang (le matin à jeun, le soir après le repas). Les règles suivantes s'appliquent sur un homme âgé (vieille personne) et diabétique ayant pris la mesure du taux de glycémie le matin. C'est-à-dire :

- Si la personne est un homme vieux et diabétique et la prise de sang a été faite le matin et le taux de glycémie est moins de 0.7g/l alors c'est une hypoglycémie sévère.
- Si la personne est un homme vieux et diabétique et la prise de sang a été faite le matin et le taux de glycémie est entre 0.7g/l et 0.8g/l alors c'est une hypoglycémie.
- Si la personne est un homme vieux et diabétique et la prise de sang a été faite le matin et le taux de glycémie est entre 0.8g/l et 1.1g/l alors c'est une glycémie normale.
- Si la personne est un homme vieux et diabétique et la prise de sang a été faite le matin et le taux de glycémie est entre 1.1g/l et 1.3g/l alors c'est une hyperglycémie.
- Si la personne est un homme vieux et diabétique et la prise de sang a été faite le matin et le taux de glycémie est plus de 1.3g/l alors c'est une hyperglycémie sévère.

La figure 6.9 est une capture d'écran sur Protégé des règles énoncées ci-dessus.

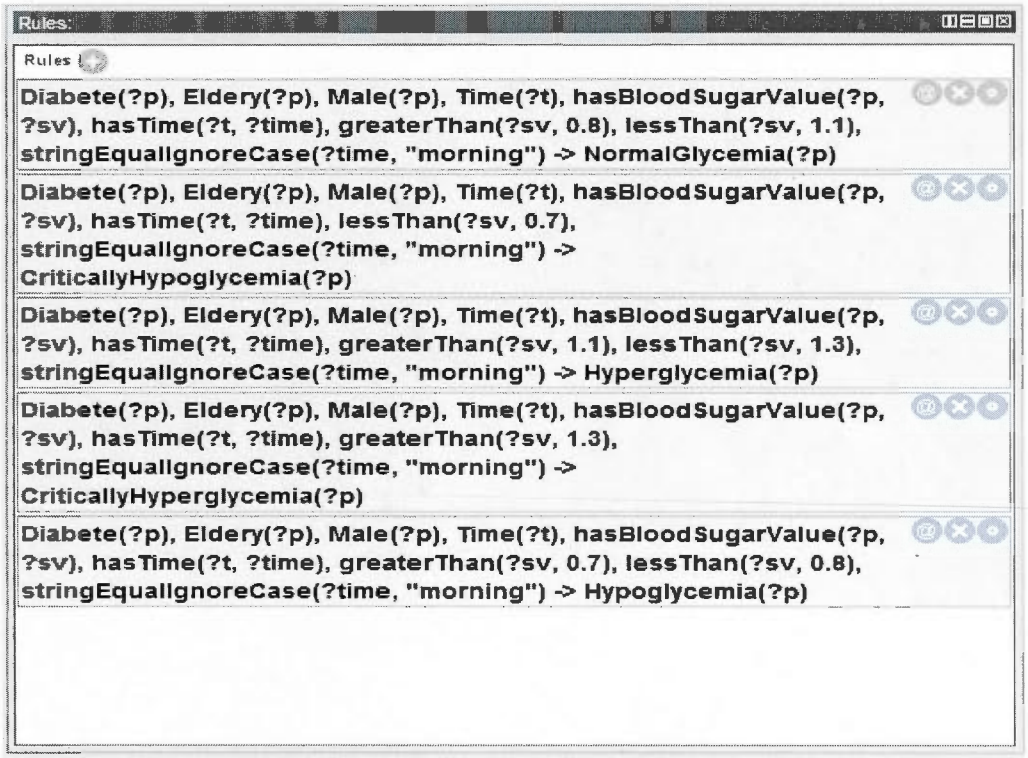

Figure 6.9 : Règles d'inférence des niveaux de glycémie de l'utilisateur.

Nous avons procédé de la même manière pour établir les règles en variant le genre de la personne, sa catégorie d'âge, son état de santé, les valeurs de la capture de glycémie et la période de la journée de la prise de la valeur (matin, soir).

6.4 Simulation des règles d'inférence

Nous avons effectué quelques simulations sur *Protégé* afin de s'assurer du fonctionnement correct des règles d'inférences. Ainsi, nous avons peuplé l'ontologie avec des individus (*Individuals*), nous avons défini les propriétés de ces individus et nous avons lancé le moteur d'inférence.

Nous avons défini une personne (*Ahmed*) avec les propriétés suivantes :

- Age = 20 ans.
- Genre = male.

- Temperature corporelle = 40.

Nous avons rentré manuellement cet individu sur Protégé sans « *Individuals* » et nous avons défini de la même manière ses propriétés avec « *Property Assertion* » en rentrant la propriété et sa valeur.

La figure 6.10 ci-dessous représente une capture écran de l'initiation d'un individu (*Ahmed*) et ses propriétés.

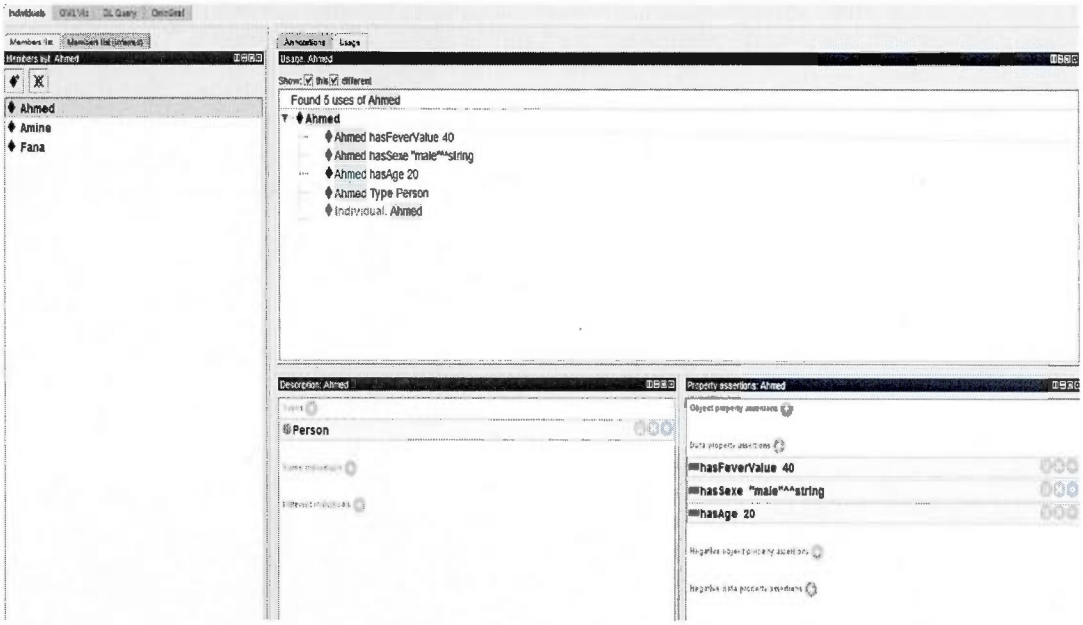

Figure 6.10 : Capture d'écran – Initiation de la description d'un individu.

Nous avons par la suite lancé le moteur d'inférence *RacerPro*. Après cette opération nous pouvons vérifier la classification de l'individu selon les propriétés qu'on lui a attribué. Nous avons procédé à cette vérification par la nouvelle description de l'individu après inférence. Comme le montre la figure 6.11 qui est une capture d'écran de Protégé de l'individu *Ahmed* après l'inférence. *Ahmed* est classé sous *Adult*, *Male* et *CriticallyHightFever* (température corporelle très élevée).

Nous avons aussi défini une deuxième personne (*Fana*) avec des propriétés différentes. Les propriétés attribuées à Fana sont ainsi :

- Age = 10 ans.
- Genre = female.
- Temperature corporelle = 37.

La figure 5.12 représente l'initiation de l'individu (*Fana*) et ses propriétés que nous avons saisies d'une façon manuelle.

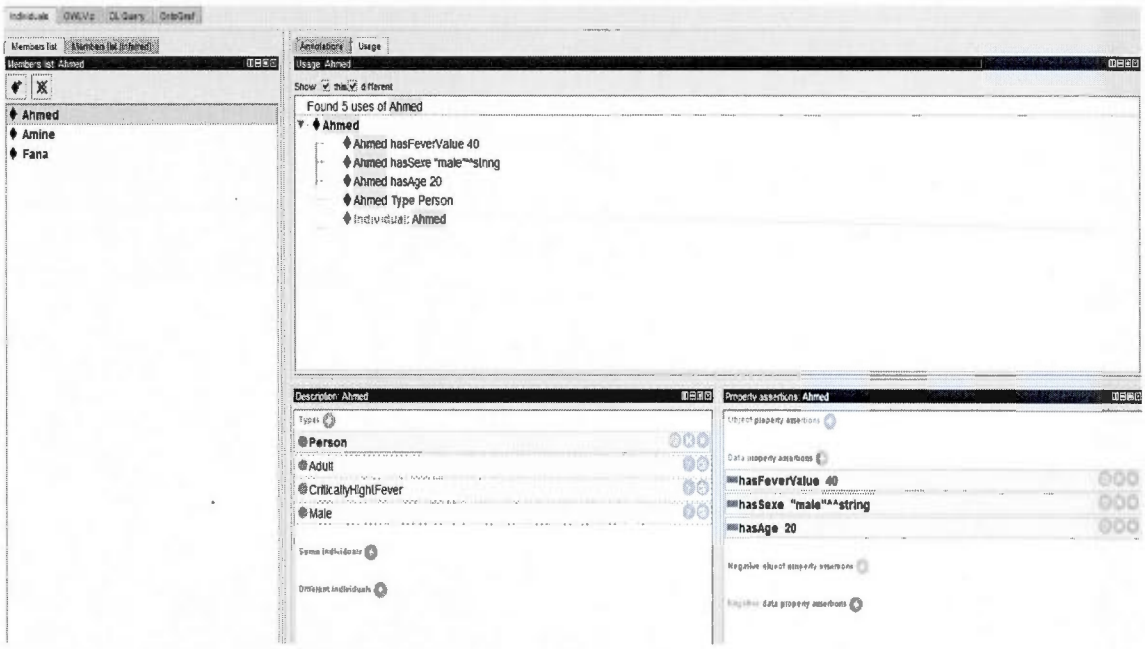

Figure 6.11 Individu Ahmed après inférence.

Nous avons par la suite lancé le moteur d'inférence. Après l'inférence, *Fana* est classée sous *Child, Female* et *NormalFever* (température normale). La figure 6.13 montre l'individu *Fana* après le lancement du moteur d'inférence.

Les simulations précédentes démontrent que l'inférence nous donne des résultats corrects et prouve que les règles d'inférence ont été convenablement définies et implémentées.

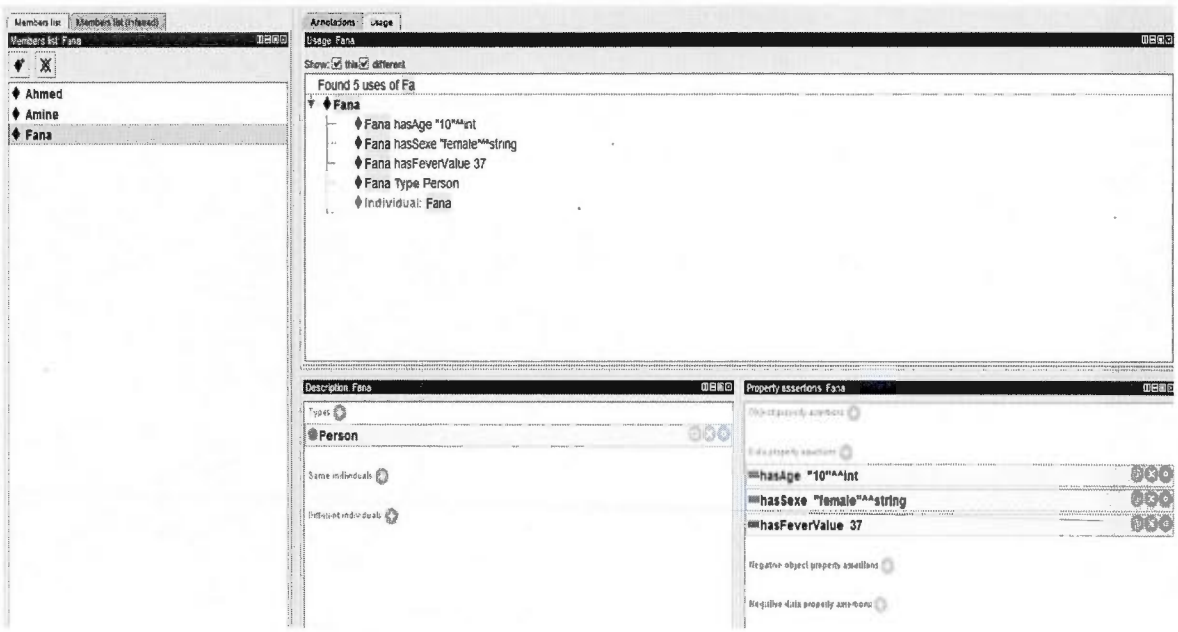

Figure 6.12 : Initiation de l'individu Fana.

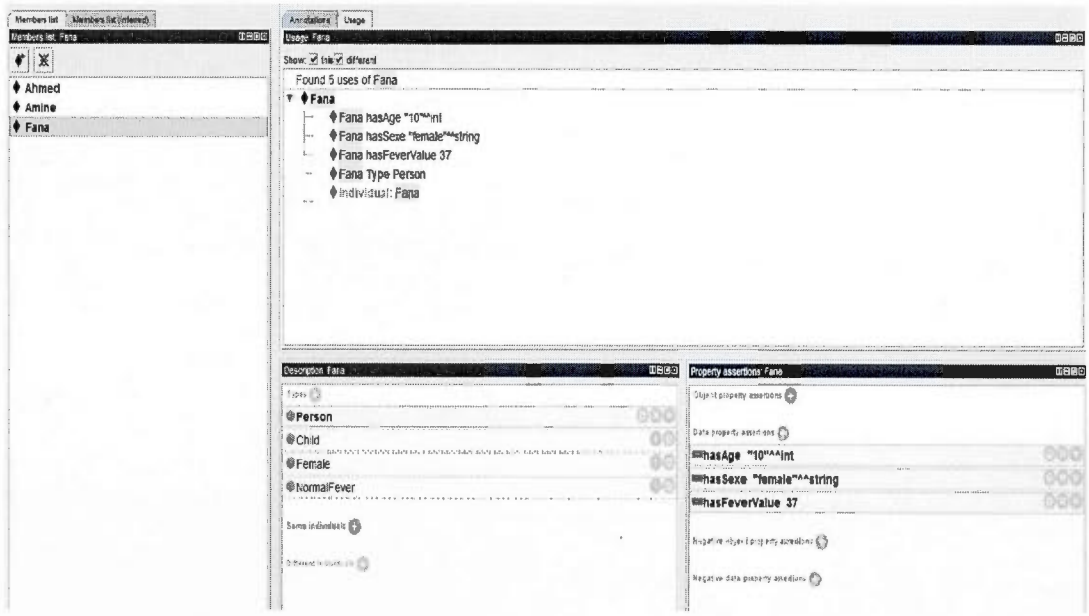

Figure 6.13 : Individu Fana après inférence.

Les autres règles d'inférence ont été développées de la même manière et sur le même principe afin d'avoir une modèle qui couvre tous les cas possibles pour chaque état physiologique. Cependant, dans le cas d'une application réelle de notre modèle, le peuplement de l'ontologie se fait automatiquement et la définition des propriétés se fait à travers l'extraction de ces données de *Facebook* ou la réception des informations des capteurs.

CHAPITRE VII

CONCLUSION

Nous avons présenté un modèle de contexte qui utilise les informations du réseau social en ligne *Facebook*. Nous avons présenté une architecture middleware dépendante du contexte formée de modules interconnectés et effectuant des traitements différents.

Dans l'architecture introduite, nous avons proposé des améliorations en se basant sur les systèmes existants en ajoutant un module de filtrage pour l'interception des données erronées et une couche de gestion responsable de la communication entre le middleware et les services du contexte.

Nous avons démontré l'utilité et l'apport des réseaux sociaux en ligne dans la définition du contexte de l'utilisateur dans le cadre d'une application du domaine médical.

Dans l'analyse des informations de contexte fournies par *Facebook,* nous avons extrait des informations pour les intégrer à une ontologie globale et l'avons appliquée dans le cadre d'une application médicale simple en y intégrant le modèle de *Facebook.*

Ce couplage entre ces deux éléments nous a permis d'exploiter les informations à l'intérieur du réseau *Facebook.* Nous avons défini différents contextes de l'utilisateur par le moyen de règles d'inférences que nous avons testées avec succès.

Nous avons pu réaliser presque tous les objectifs définis dans le cadre de ce mémoire. Cependant, et faute de temps, nous n'avons pas pu faire une implémentation complète d'une application capable d'aller chercher des informations de *Facebook* et des différents capteurs et de réagir dépendamment du contexte déduit. Cet aspect reste donc à compléter dans le

futur, sur quoi nous avons l'intention de travailler. Nous avons aussi négligé l'aspect d'évolutivité de la plateforme *Facebook*. Il serait intéressant de développer un mécanisme permettant de mettre à jour d'une manière dynamique le modèle ontologique dés que l'architecture de *Facebook* subisse des changements.

Ce travail nous a présenté quelques défis intéressants au niveau de la modélisation de Facebook:

- Cette plateforme offre une API à partir de laquelle nous avons pu construire notre modèle. Cependant, cette API ne décrit pas entièrement les éléments de la plateforme et nous avons investi plus de temps pour la réalisation de cette tâche.
- La version 4.1.0 de Protégé que nous avons utilisée, ne fournit pas un éditeur de règles d'inférence développé. L'éditeur disponible est très simple et les règles doivent être décrites manuellement. De plus, cet éditeur ne permet pas l'insertion de plus de 105 règles, une limite plus petite que le nombre de règles que nous avons.

Dans une perspective de développement futur, nous prévoyons implémenter une application complète qui utilise notre modèle de contexte. Nous pouvons aussi développer un programme de fouille automatique de *Facebook* qui pourrait extraire les données nécessaires à notre application. Au niveau du modèle, nous suggérons d'utiliser d'autres réseaux sociaux en ligne pour enrichir le modèle de contexte et obtenir d'autres informations qui ne sont pas disponibles sur *Facebook*. Enfin, nous pouvons proposer d'utiliser les informations de *Facebook* pour des modèles de contexte relatifs à des domaines autres que le domaine médical expérimenté dans ce travail.

Ce travail a été très enrichissant et nous a permis de travailler sur un sujet de l'heure et a ouvert des perspectives et des problématiques intéressantes dans le futur.

RÉFÉRENCES

[1] Zuraini Zainol and Keiichi Nakata. 2010. "Generic context ontology modelling: A review and framework". In *2nd International Conference on Computer Technology and Development (ICCTD 2010),* Piscataway, NJ : IEEE, p. 126-130.

[2] Anind K. Dey, Gregory D. Abowd, and Daniel Salber. 2001. "A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications". *Human-Computer Interaction*, vol. 16, no 2, p. 97-166.

[3] M. Mohsin Saleemi, Natalia Díaz Rodríguez, Johan Lilius, and Iván Porres. 2011. "A framework for context-aware applications for smart spaces". In *Proceedings of the 11th international conference and 4th international conference on Smart spaces and next generation wired/wireless networking (NEW2AN'11/ruSMART'11)*, Sergey Balandin, Yevgeni Koucheryavy, and Honglin Hu (Eds.). Berlin, Heidelberg: Springer-Verlag, Lecture Notes in Computer Science, vol. 6869, p. 14-25.

[4] Thomas R. Gruber. 1995. "Toward principles for the design of ontologies used for knowledge sharing". *International Journal of Human-Computer Studies - Special issue: the role of formal ontology in the information technology*, vol. 43, no 5-6 p. 907-928.

[5] Joachim Kleb and Raphael Volz. 2009. "Ontology based entity disambiguation with natural language patterns". In *4th International Conference on Digital Information Management (ICDIM 2009)*, Piscataway, NJ: IEEE, p.1-8.

[6] Kaushal Giri. 2011. "Role of ontology in semantic web". *DESIDOC Journal of Library & Information Technology*, vol. 31, no 2, p. . 116-120.

[7] Kaveh Bazargan. 2004. "Le rôle des ontologies de domaine dans la conception des interfaces de navigation pour des collections en ligne de musées : évaluation et

proposition". Mémoire de DEA en Management et Technologies des Systèmes d'Information (MATIS), Genève, Université de Genève. Retrieved from http://cui.unige.ch/~bazargan/PDF/Rapport-KB-DEA-MATIS.pdf (accessed may10, 2012).

[8] M. Poveda-Villalón, M. Carmen Suárez-Figueroa, R. García-Castro, and A. Gómez-Pérez. 2010. "A Context Ontology for Mobile Environments". In *Proceedings of the Second Workshop on Context Information and Ontologies (CIAO2010)*, Tilburg University, The Netherlands : CEUR-WS.org, vol. 626. Retrieved from http://ceur-ws.org/Vol-626/regular3.pdf (may 10, 2010).

[9] Nicola Guarino. 1998. "Formal Ontology in Information Systems". In *Proceedings of the First International Conference on Formal Ontology in Information Systems (FOIS'98)*. Amsterdam, The Netherlands : IOS Press. Retrieved from http://www.loa.istc.cnr.it/Papers/FOIS98.pdf (may 10, 2012)

[10] Harry Chen, Tim Finin, and Anupam Joshi. 2003. "An Ontology for Context-Aware Pervasive Computing Environments". The Knowledge Engineering Review, vol. 18, no 3, p. 197-207. Retrieved from http://ebiquity.umbc.edu/paper/html/id/60/An-Ontology-for-Context-Aware-Pervasive-Computing-Environments (accessed may 9, 2012).

[11] Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg. 2007. "A survey on context aware systems". Int. J. Ad Hoc Ubiquitous Comput. 2, 4 (June 2007), 263-277.

[12] Philipp Gutheim. 2011. "An ontology-based context inference service for mobile applications in next-generation networks". *IEEE Communications Magazine,* vol. 49, no 1, p. 60-66.

[13] Jean Le Feuvre, Cyril Concolato, and Jean-Claude Dufourd. 2009. "Widgets mobility". In *Proceedings of the 6th International Conference on Mobile Technology,*

Application & Systems (Mobility '09). New York, NY, USA : ACM, Article 25 , 4 pages.

[14] Xu Ying and Xu Fu-yuan. 2006. "Research on Context Modeling Based on Ontology". In *CIMCA '06 Proceedings of the International Conference on Computational Inteligence for Modelling Control and Automation and International Conference on Intelligent Agents Web Technologies and International Commerce*. Washington, DC, USA : IEEE Computer Society, p. 188-196.

[15] Tao Gu, Hung Keng Pung, and Da Qing Zhang. 2005. " A service-oriented middleware for building context-aware services ". *Journal of Network and Computer Applications*, vol. 28, no 1, p. 1-18.

[16] Michael Wooldridge and Nick R. Jennings. 1995. "Intelligent agents: theory and practice". *The Knowledge Engineering Review*, vol. 10, no 2, p. 115-152.

[17] Harry Chen. 2003. "An Intelligent Broker Architecture for Context-Aware Systems". PhD. Dissertation. Baltimore, MD, USA : University of Maryland Baltimore County. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.13.3938&rep=rep1&type=pdf. (accessed may 8, 2012)

[18] Patrick Fahy and Siobhan Clarke. 2004. "CASS – a middleware for mobile context-aware applications", paper presented at Workshop on Context Awareness, MobiSys 2004, Boston, Massachusetts, USA, June 6, 2004. Retrieved from http://www.sigmobile.org/mobisys/2004/context_awareness/papers/cass12f.pdf (accessed may 9, 2012)

[19] Thomas Hofer, Wieland Schwinger, Mario Pichler, Gerhard Leonhartsberger, Josef Altmann, and Werner Retschitzegger. 2003. "Context-awareness on mobile devices - the hydrogen approach". In *Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS '03)*, Washington, DC, USA : IEEE Computer Society, Track 9 – vol. 9, 10 pages.

[20] Natalia Diaz Rodriguez. 2011. "A Framework for Context-Aware Applications for Smart Spaces". In *IEEE/IPSJ 11th International Symposium on Applications and the Internet (SAINT), 2011*, Washington, DC, USA: IEEE Computer Society, p. 218-221.

[21] Andrew D. Jurik and Alfred C. Weaver. 2009. "Body Sensors: Wireless Access to Physiological Data".*IEEE Software*, vol. 26, no 1, p. 71-73.

[22] Hai Van Luu and Xueyan Tang. 2011. "An Efficient Multi-path Data Collection Scheme in Wireless Sensor Networks". In*Proceedings of the 31st International Conference on Distributed Computing Systems Workshops*, Washington, DC, USA : IEEE Computer Society, p. 198-207.

[23] Bilel Romdhani, Dominique Barthel, and Fabrice Valois. 2011. "Routing for Data-Collection in Heterogeneous Wireless Sensor Networks". In *IEEE 73rd Vehicular Technology Conference (VTC Spring)*, Piscataway, NJ : IEEE, p. 1-5,

[24] Pham Thi Thu Thuy, Young-Koo Lee, and Sung Young Lee. 2009. "DTD2OWL: automatic transforming XML documents into OWL ontology". In *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human (ICIS '09)*. New York, NY, USA : ACM,p. 125-131.

[25] Shanping Li, Zhaohui Yang, and Xin Lin. 2005 "RTCR: a soft real-time context reasoner," In *Second International Conference on Embedded Software and Systems(ICESS'05),*Washington, DC, USA : IEEE Computer Society, p. 386-391.

[26] Hung Quoc Ngo, Anjum Shehzad, Kim Anh Pham Ngoc, S.Y. Lee, and Manwoo Jeon. 2005. "Research issues in the development of context-aware middleware architectures". In *Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications RTCSA '05*, Washington, DC, USA : IEEE Computer Society, p. 459- 462.

[27] Gu Jun-Zhong. 2009."Contexte Aware Computing". *Journal of East China Normal University (Natural Science)*, vol. 5, p. 1-20.

[28] Kyle Chard, Kris Bubendorfer, Simon Caton, and Omer Rana. 2011. "Social Cloud Computing: A Vision for Socially Motivated Resource Sharing". *IEEE Transactions on Services Computing,* preprint.

[29] Salma Noor and Kirk Martinez. 2009. "Using social data as context for making recommendations: an ontology based approach". In *Proceedings of the 1st Workshop on Context, Information and Ontologies(CIAO '09)*, Jose Manuel Gomez-Perez, Peter Haase, Marcel Tilly, and Paul Warren (Eds.). New York, NY, USA : ACM, Article No. 7.

[30] Milan Stankovic. 2008."Modeling Online Presence", In *Proceedings of the First Social Data on the Web Workshop, Karlsruhe, Germany, October 27*, CEUR Workshop Proceedings. Retrieved fromhttp://milstan.net/papers/paper2.pdf (accessedmay 24).

[31] Alvaro Ortigosa, Jose Ignacio Quiroga, and Rosa M.Carro. 2011. "Inferring user personality in social networks: A case study in Facebook". In*11th International Conference on Intelligent Systems Design and Applications (ISDA),*Piscataway, NJ : IEEE,p. 563-568.

[32] Anusuriya Devaraju and Simon Hoh. 2008. "Ontology-based context modeling for user-centered Context-aware Services Platform". In *International Symposium on* Information *Technology (ITSim 2008)*, Piscataway, NJ : IEEE,vol. 2, p. 1-7.

[33] Feruzan Ay.2007. "Context Modeling and Reasoning using Ontologies". Unpublished work. Retrieved from http://www.ponnuki.de/cmaruo/cmaruo.pdf (accessed may 24, 2012).

[34] Alan Jovic, Marin Prcela, and Dragan Gamberger. 2007. "Ontologies in Medical Knowledge Representation". In *Proceedings of the 29th International Conference on Information* Technology *Interfaces (ITI 2007)*, Piscataway, NJ : IEEE, p. 535-540.

[35] Jérôme Euzenat and Petko Valtchev. 2003."An Integrative proximity measure for ontology alignement", In *Proceedings of the 1st International Workshop on Semantic*

Integration (ISWC-2003), CEUR-WS.org, vol. 82, p. 33-38. Retrieved from http://ceur-ws.org/Vol-82/SI_paper_06.pdf (accessed may 24, 2012)

[36] Namyoun Choi, Il-Yeol Song, and Hyoil Han. 2006. "A survey on ontology mapping". *ACM SIGMOD Record*, vol. 35, no 3, p. 34-41.

[37] Xiao Hang Wang, Da Qing Zhang, Tao Gu, and Hung Keng Pung.2004. "Ontology based context modeling and reasoning using OWL". In *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops,* Washington DC : IEEE Computer Society,p. 18-22.

[38] Petko Valtchev. Automne 2011."Thème 5 La couche logique et inférentielle: les règles". *Notes du cours DIC938I*