وزارة التعليم العالي و البحث العلمي

BADJI MOKHTAR-ANNABA UNIVERSITY

UNIVERSITE BADJI MOKHTAR-ANNABA

جامعة باجي مختار ـ عنابة

Faculté des sciences de l'ingénieur

Département d'informatique

Année : 2013

MEMOIRE

Présenté en vue de l'obtention

du diplôme de MAGISTER en Informatique

Utilisation de l'approche basée agent pour la gestion de l'information de contexte dans le domaine de l'intelligence ambiante afin d'assister des personnes handicapés dans leur maison

Option

Informatique Embarqué

Par

Berraouna Abdelkader

DIRECTEUR DE MEMOIRE : Dib Lynda , Professeur, Université d'Annaba.

DEVANT LE JURY

PRESIDENTE: **Dr Labiba Souici-Meslati** Professeur, Université d'Annaba.

EXAMINATEURS: **Dr Mohame Tarek Khadir** Professeur, Université d'Annaba.

Dr Nora Bounour MC, Université d'Annaba.

.

Abstract

People with disabilities compose a segment of the population that can benefit by sensitive ambient intelligence contexts (context-aware).

In this context sensitive environment the system should collect contextual through different ways depending on availability of technology and process this information appropriately to perform actions or alarms or modify the environment information.

In this new research we will try to propose a new approach for modeling context information, and that to advise and assist a disabled person in a **Smart Home** as well as ensuring an automatic management of the assistance of the disabled in smart home.

Our work will focus on the study of the entire intelligent environment, particularly:

- make a complete management of ambient intelligent environment
- Energy House
- user comfort (air conditioning, heating)
- safety of the user (if a gas leak, accessing the handicap).
- Suggest actions near the user to better control the Smart Home.
- remember the time of taking the medication user (disabled) and that according to his state of health at a given time
- detect unwanted events (buyers) that can occur at a given moment and the disabled come to its aid by sending an alarm to his assistant

Keyword: context-aware, Smart Home, ambient intelligence, Ontology.

Résumé

Les personnes handicapées composent un segment de la population qui peut profiter beaucoup de ***l'intelligence ambiante sensible au contexte (context-aware)***.

Dans cet environnement sensible au contexte le système devrait recueillir des informations contextuelles par le biais de différentes manières en fonction de la disponibilité de la technologie et du traitement de ces informations de manière adéquate et cela dans le but d'exécuter des actions ou des alarmes ou modifier l'environnement de l'intelligence ambiante.

Notre contribution dans cet axe de recherche est de proposer une nouvelle approche pour la modélisation de ***l'information de contexte*** afin de conseiller et d'assister une personne handicapée dans une maison intelligente (***Smart Home)***

Notre travail de recherche sera focalisé sur l'étude de tout l'environnement intelligent Smart Home particulièrement sur :

- Réaliser une gestion complète de l'***environnement ambiant intelligent***
- Controller l'énergie de la maison
- Assurer le confort de l'utilisateur (climatisation, chauffage)
- Assurer la sécurité de l'utilisateur (en cas de fuite de gaz, accident de l'handicapée).
- suggérer des actions de la gestion de la maison intelligente à de l'utilisateur et cela pour meilleur contrôle.
- rappeler l'handicapée son heure de prise des médicaments selon son état de santé au cours du temps
- détecter les événements non souhaités (accidents) qui peuvent arriver à un moment donné à l'handicapée et venir à son aide en envoyant une alarme à son assistant.

Mots clés : ***Sensible au contexte***, ***Intelligence ambiante***, **Maison intelligente, Ontologie.**

DEDICACE

Je dédie ce modeste travail

Tout d'abord à mes chers parents, pour leur sens du devoir et soutien moral pendant toute ma vie et mes années d'éludes

A mes sœurs Radia et Fatma et mon petit frère Chouaib et toute ma grande famille de loin ou de proche

Et à tous mes amis et collègues d'étude

Et à tous ceux qui j'ai oublie de mentionner

Remerciement

Je remercie Dieu « Allah »le tout puissant, qui « nous aime, nous protège, nous guide vers le bien, nous donne la chance et le bonheur » de m'a aidé à mener à bout ce travail, et à le fait concrétiser.

Je tien à remercier Mlle Dib lynda qui a bien voulu diriger ce mémoire ; de m'a accordé sa confiance pour son aide et ses précieux conseils.

Mes remerciements et considérations aux membres du jury, qui ont bien voulu accepter d'examiner et juger mon travail.

Ma grande reconnaissance a Me Rachid boudour qui nous beaucoup aider durant notre formation de magister

Mes remerciements aussi à mes collègues dans la formation de magister INEM2011 ,Ahmed Malek Nada ,Nabet Aida et toutes les autres

En fin, à tous ceux qui m'ont soutenus de prés ou de loin, notamment : à ma familles et à touts personnes qui ont contribués à l'élaboration de ce mémoire.

Table des matières

Liste des Figures

Chapitre 1

Introduction générale

1.1. Contexte

Les personnes à mobilité réduite et les personnes âgées rencontrent beaucoup de difficulté dans leur vie quotidienne et ne peuvent pas, parfois difficilement de commander la multitude d'équipements présents dans leur habitat. Beaucoup d'entre eux ont besoin d'aide pour réaliser leurs activités quotidiennes, telle que la prise de médicament, le suivi de la santé, le contrôle des équipements électroniques se trouvant dans leur maison (climatiseur le chauffage), …etc.

La domotique présente une solution envisageable pour compenser les personnes à mobilité réduite et les personnes âgées en leur permettant de réaliser les activités de la vie quotidienne sans l'aide systématique d'une tierce personne. Pour se faire, le système doit être capable de faciliter voire d'automatiser l'activation de certains services et de proposer aux utilisateurs des services adaptés.

La domotique consiste à réaliser des habitats intelligents se composant d'objets communicants capables de traiter de l'information et de proposer des services de manière indépendante.

La domotique intelligente rejoint le domaine de l'informatique ubiquitaire et les systèmes sensibles au contexte. Ces systèmes prennent en compte de manière dynamique les besoins des utilisateurs ainsi que des informations sur le contexte.

1.2. Problématique

Plusieurs modèles et techniques sont proposés pour modéliser les informations de contexte dans un environnement domotique tel que les ***Modèles Clé-Valeur, graphes contextuels, langages de balisage et les ontologies.*** *Chacun de ces modèles a des avantages et des limites mais l'*approche de modélisation du contexte basée sur l'ontologie reste le meilleur choix par rapport aux autres approches et cela grâce aux avantages que la représentation ontologique présente comme l'e***xpressivité***, ***le partage des connaissances, l''inférence logique, la réutilisation des connaissances, …etc.***

Au cours des dernières années beaucoup d'ontologies génériques ont été développés pour modéliser et représenter les concepts généraux de contexte, mais elles ne sont pas spécifiques pour le domaine de l'assistance des personnes handicapées à domicile et les maisons intelligentes *SmartHose.*

1.3. Une approche de modélisation du contexte basée ontologie

L'objectif de notre travail est double.

La première partie consiste à ***proposer une modélisation de l'information de contexte dans un environnement domotique en utilisant les ontologies comme modèle de représentation*** et cela afin d'assister les personnes handicapés dans leur maison.

La deuxième partie nous développons un outil qui sera basé sur l'ontologie créée ainsi que sur un mécanisme de raisonnement basé sur le contexte de cette dernière.

Le modèle envisagé dans notre travail prend en charge le contexte de l'utilisateur et agit sur l'environnement pour le conseil et l'assistance de ce dernier en lui suggérant des actions qu'il juge les mieux appropriées.

De plus le système suggère des actions au prés de l'utilisateur et cela faire une gestion complète de l'environnement et des différentes alarmes en cas de problème détecté dans la maison, pour mieux contrôler sa maison ainsi que pour contrôler l'énergie de la maison, du confort des utilisateurs (climatisation, chauffage), de la sécurité de l'utilisateur (fuite de gaz, accident de l'handicapée, ...etc)

1.4. Organisation du mémoire

Ce mémoire de magistère s'articule en 5 chapitres qui nous permettront de présenter les différents aspects de notre travail.

Le premier chapitre présente une introduction à l'intelligence ambiante, l'informatique sensible au contexte dans un environnement domotique, l'architecture de ces applications, plus la définition de notion du contexte et des différentes techniques utilisées pour capturer et modéliser les informations du contexte.

Le second chapitre présente la notion d'ontologie, ses composants, le cycle de son développement et comment elle est utilisée dans la modélisation de contexte dans un environnement sensible au contexte. Une démonstration de quelques ontologies génériques de modélisation de contexte sera présentée dans ce chapitre.

Le troisième chapitre s'intéresse à la description des étapes suivies dans le développement de notre ontologie et cela après une présentation du scenario de l'assistance de l'handicapée et les différents concepts et composants de l'environnement, la conceptualisation avec le langage UML et la formalisation de l'ontologie avec le langage OWL.

Le quatrième chapitre présente le mécanisme de raisonnement sur le contexte que nous avons proposons pour le développement de notre système ainsi que les différents outils utilisés. Ensuite Il présente une démonstration des résultats avec l'environnement SWRLTab.

Enfin, nous terminons ce mémoire de magistère par une conclusion qui résume l'apport essentiel de l'approche proposée et nous introduisant quelques perspectives de recherche.

Chapitre 1

Introduction générale

1.1. Contexte

Les personnes à mobilité réduite et les personnes âgées rencontrent beaucoup de difficulté dans leur vie quotidienne et ne peuvent pas, parfois difficilement de commander la multitude d'équipements présents dans leur habitat. Beaucoup d'entre eux ont besoin d'aide pour réaliser leurs activités quotidiennes, telle que la prise de médicament, le suivi de la santé, le contrôle des équipements électroniques se trouvant dans leur maison (climatiseur le chauffage), …etc.

La domotique présente une solution envisageable pour compenser les personnes à mobilité réduite et les personnes âgées en leur permettant de réaliser les activités de la vie quotidienne sans l'aide systématique d'une tierce personne. Pour se faire, le système doit être capable de faciliter voire d'automatiser l'activation de certains services et de proposer aux utilisateurs des services adaptés.

La domotique consiste à réaliser des habitats intelligents se composant d'objets communicants capables de traiter de l'information et de proposer des services de manière indépendante.

La domotique intelligente rejoint le domaine de l'informatique ubiquitaire et les systèmes sensibles au contexte. Ces systèmes prennent en compte de manière dynamique les besoins des utilisateurs ainsi que des informations sur le contexte.

1.2. Problématique

Plusieurs modèles et techniques sont proposés pour modéliser les informations de contexte dans un environnement domotique tel que les ***Modèles Clé-Valeur, graphes contextuels, langages de balisage et les ontologies.*** *Chacun de ces modèles a des avantages et des limites mais l'*approche de modélisation du contexte basée sur l'ontologie reste le meilleur choix par rapport aux autres approches et cela grâce aux avantages que la représentation ontologique présente comme l'e***xpressivité***, ***le partage des connaissances, l''inférence logique, la réutilisation des connaissances, …etc.***

Au cours des dernières années beaucoup d'ontologies génériques ont été développés pour modéliser et représenter les concepts généraux de contexte, mais elles ne sont pas spécifiques pour le domaine de l'assistance des personnes handicapées à domicile et les maisons intelligentes *SmartHose.*

1.3. Une approche de modélisation du contexte basée ontologie

L'objectif de notre travail est double.

La première partie consiste à ***proposer une modélisation de l'information de contexte dans un environnement domotique en utilisant les ontologies comme modèle de représentation*** et cela afin d'assister les personnes handicapés dans leur maison.

La deuxième partie nous développons un outil qui sera basé sur l'ontologie créée ainsi que sur un mécanisme de raisonnement basé sur le contexte de cette dernière.

Le modèle envisagé dans notre travail prend en charge le contexte de l'utilisateur et agit sur l'environnement pour le conseil et l'assistance de ce dernier en lui suggérant des actions qu'il juge les mieux appropriées.

De plus le système suggère des actions au prés de l'utilisateur et cela faire une gestion complète de l'environnement et des différentes alarmes en cas de problème détecté dans la maison, pour mieux contrôler sa maison ainsi que pour contrôler l'énergie de la maison, du confort des utilisateurs (climatisation, chauffage), de la sécurité de l'utilisateur (fuite de gaz, accident de l'handicapée, ...etc)

1.4. Organisation du mémoire

Ce mémoire de magistère s'articule en 5 chapitres qui nous permettront de présenter les différents aspects de notre travail.

Le premier chapitre présente une introduction à l'intelligence ambiante, l'informatique sensible au contexte dans un environnement domotique, l'architecture de ces applications, plus la définition de notion du contexte et des différentes techniques utilisées pour capturer et modéliser les informations du contexte.

Le second chapitre présente la notion d'ontologie, ses composants, le cycle de son développement et comment elle est utilisée dans la modélisation de contexte dans un environnement sensible au contexte. Une démonstration de quelques ontologies génériques de modélisation de contexte sera présentée dans ce chapitre.

Le troisième chapitre s'intéresse à la description des étapes suivies dans le développement de notre ontologie et cela après une présentation du scenario de l'assistance de l'handicapée et les différents concepts et composants de l'environnement, la conceptualisation avec le langage UML et la formalisation de l'ontologie avec le langage OWL.

Le quatrième chapitre présente le mécanisme de raisonnement sur le contexte que nous avons proposons pour le développement de notre système ainsi que les différents outils utilisés. Ensuite Il présente une démonstration des résultats avec l'environnement SWRLTab.

Enfin, nous terminons ce mémoire de magistère par une conclusion qui résume l'apport essentiel de l'approche proposée et nous introduisant quelques perspectives de recherche.

Chapitre 2

Introduction à l'intelligence ambiante et aux différentes approches de modélisation de l'information du contexte

2.1 Introduction à l'intelligence ambiante et l'informatique sensible au contexte

Dans ce chapitre on va définir les concepts importants qui sont traités dans notre tâche de recherche et cela pour comprendre le contexte de notre travail et pour centrer l'effort de recherche. On va également définir et décrire la notion de l'intelligence ambiante, l'informatique sensible au contexte, le contexte de l'utilisateur ainsi que les différentes approches actuelles utilisées pour le représenter.

2.1.2 Définition de l'intelligence ambiante

Le mot intelligence ambiante (AmI) est apparu au début des années 90. La, définition de l'AmI n'est toujours pas encore claire car elle est assez récente et l'avancé de la recherche fait évaluer les points de vue potentiels de ce concept [24].

Pour Bergamann l'intelligence ambiante s'insère dans une vision futuriste des télécommunications et d'informatisation de la vie courante. Notre environnement futur sera entouré par différents systèmes et applications qui seront portés par les technologies réseaux et l'informatique et cela pour fournir une aide intelligente sans l'intrusion humaine [26].

Pour Bermejo Nieto l'intelligence ambiante est une mise en relation avec des systèmes de services, c'est-à-dire, avec les technologies pour automatiser les actions et avec les dispositifs pour personnaliser et adapter leurs comportements. Aujourd'hui on fabrique des objets de plus en plus petits qu'on peut les intégrer dans des appareils électroniques. Ceci accroît leur utilisation et entraîne des nouvelles recherches. De plus, la recherche et le progrès en communication enrichissent le développement et l'utilisation de l'intelligence ambiante. Le succè de l'AmI dépend en grande partie du développement de la technologie des capteurs, d'actuateurs, etc, et également des logiciels et de leur intelligence pour la prise de décision [27].

Lorsqu'on parle d'intelligence ambiante, on retrouve souvent les mots suivants : informatique ubiquitaire, pervasive, context aware computing, technologies persuasives et domotique.

Dans ce qui suit nous introduirons la définition de chacun de ces concepts.

2.1.3 Informatique ubiquitaire omniprésente et préavise

C'est le model d'interaction homme-machine dans lequel le traitement d'information est intégré aux activités quotidiennes (intégration de l'informatique dans l'environnement habituel de la personne). Le point clé de l'informatique ubiquitaire est la disponibilité de ses outils et préférences (notion de profil) où que l'on soit.

Dans la réalité, Il existe une différence entre informatique ubiquitaire et informatique pervasive. L'informatique pervasive est plus générale et utilisée pour décrire des services omniprésents alors que l'informatique ubiquitaire est plus concrète, se retrouve partout, et aussi fait référence au profil de l'utilisateur [24].

2.1.4 informatique sensible au contexte (Context Aware Computing)

C'est la classe des systèmes mobiles qui peuvent percevoir et ressentir leur environnement physique (leur contexte d'utilisation) et adapter leurs comportements en conséquence. Les aspects les plus importants à connaitre sur le contexte sont trois : « où est l'utilisateur », « avec qui est l'utilisateur » et « quels sont les moyens dont il dispose » [24].

Notre travail de recherche sera focalisé sur ces notions de base ainsi que sur le cycle de vie d'une application sensible au contexte.

2.1.5 Technologies persuasives :

Ce sont les techniques qui changent intentionnellement les attitudes ou les comportements par la persuasion et l'influence sociale. Ces technologies sont utilisées dans beaucoup de domaines comme la politique, la religion, la gestion, la direction d'entreprises, …etc.

2.1.6 Domotique :

Les développements technologiques des dernières décennies ont introduit progressivement l'électronique dans notre quotidien dans des proportions devenues importantes. Cette incursion permet aujourd'hui d'imaginer, de manière sociologiquement naturelle, l'immersion complète des personnes dans des environnements comme des cloisons ou du mobilier comportant des systèmes électroniques cachés avec lesquels il est possible d'interagir. L'immersion dans un environnement interactif peut avoir plusieurs intérêts pour la personne tel que :

- une interaction plus naturelle avec l'environnement,
- une facilité d'utilisation de certains systèmes électroniques ou informatiques élargissant l'accès à ces technologies,
- une meilleure expérience lors de l'utilisation de systèmes multimédia,

- une amélioration de la qualité de vie pour chacun,
- l'aide et le maintien à domicile des personnes dépendantes (personnes âgées, malades chroniques, personnes handicapées),
- le maintien d'une vie sociale pour les personnes dépendantes au travers des liens électroniques avec l'extérieur (famille, communautés),
- le suivi médical permanent à domicile des personnes.

2.1.7 Environnement domestique

L'environnement domestique c'est l'environnement qui contient des capteurs, des actionneurs et des systèmes de diffusion de l'information intégrés, discrets et non-intrusifs.

Les capteurs sont les réseaux de microphones, des caméras orientables haute-résolution, des capteurs infrarouges, des capteurs de température, des capteurs de pression dans le sol (capteurs pour positionner les personnes, pour détecter les chutes, etc.) ; des caméras embarqués sur des assistants personnels, téléphones ou des Tablets PCs, etc.

Les systèmes domotiques sont, pour la plupart, basés sur des mécanismes d'horloge. Quelques-uns autorisent des manipulations à distance, par téléphone ou smartphone, et certaines solutions commerciales, comme le logiciel HAL (Home Automated Living), permettent un pilotage par commande vocale [44].

Plusieurs projets de recherche sont lancés pour le développement d'environnement domotique. On peut citer par exemple :

- ***Le projet Iroom*** [45] : Il se fonde sur le besoin de populations hétérogènes d'utilisateurs à évoluer dans des environnements riches. Ce projet met en œuvre des solutions d'adaptation multimodales.
- ***Le projet ANR Sweet Home :*** Il a pour but l'accompagnement des personnes dans leur lieu de vie en s'appuyant sur la notion de bâtiment sensoriel.
- ***Le programme Adream*** (LAAS, Toulouse). Il propose d'étudier conjointement le bâtiment intelligent optimisé en énergie (panneaux photovoltaïques notamment) et les objets communicants en vue d'une meilleure assistance des personnes.
- ***Le projet MavHome*** [46] : Il utilise plusieurs disciplines de l'intelligence artificielle comme la fouille de données et les systèmes multi-agents. Ce projet est développé à l'université du Texas basé sur la construction d'une maison intelligente avec diverses fonctionnalités. Les capacités de cette intelligente maison sont : le contrôle adaptatif de la température, de l'eau, de la lumière, du nettoyage et du loisir multimédia (audio et vidéo).
- ***Le Projet iDorm*** : ce projet teste l'informatique ubiquitaire et l'intelligence ambiante.

2.2 Caractéristiques de l'intelligence ambiante

L'intelligence ambiante est caractérisée par quatre notions importantes :

- L'ubiquité qui est la capacité d'interaction entre l'homme et la machine de façon invisible, adéquate et personnelle ;
- La perception qui est la faculté du système de connaître et de percevoir la localisation des objets, des appareils et des personnes au moyen des capteurs pour établir le contexte ;
- L'interaction naturelle qui doit être intuitive et naturelle puisqu'on utilise des appareils utilisés quotidiennement ;
- L'intelligence qui est l'aptitude d'analyser le contexte perçu et l'ajustement dynamique aux utilisateurs et aux situations pour trouver une bonne réponse.

2. 3 Domaines de l'intelligence ambiante

L'AmI est une technologie de pointe multidisciplinaire, en plein essor et s'améliore tous les jours. Elle est utilisée dans beaucoup de domaines comme : la santé, l'éducation, la sécurité, la vidéo surveillance, la domotique, le transport public les chaînes de production, etc. Elle apporte une aide personnalisée, améliore le fonctionnement et les fonctionnalités des systèmes comme elle permet d'économiser le facteur temporel.

La domotique, ou la construction de maisons intelligentes, est le champ de recherche le plus exploré vu qu'il est appliqué dans notre vie quotidienne. De plus, il y a des appareils dans les maisons qui peuvent embarquer des capteurs pour obtenir des informations sur leur utilisation et qui peuvent automatiser les tâches. Les bénéfices espérés par utilisation de l'AmI c'est la sécurité, le confort et l'économie.

Les maisons intelligentes (Smart Home) offrent un environnement plus adéquat et une très bonne solution pour assister les personnes qui ont des besoins spéciaux comme les personnes âgées et les personnes handicapées. Grâce aux capteurs installés on peut contrôler tout l'environnement (la maison), détecter les situations irrégulières et alerter les services d'urgences.

L'intelligence ambiante aussi est utilisée dans la santé et les hôpitaux pour améliorer l'efficacité des services. Il existe, aussi, des services pour augmenter la sécurité comme permettre l'accès seulement aux personnes autorisées aux lieux, aux appareils, aux données qui ont différentes priorités. Les centres médicaux sont plus sophistiqués et par conséquent il y a plus de génération, de transaction d'information et d'activités de connaissance.

On emploie aussi l'AmI dans l'éducation. Le but dans ces applications est la surveillance du progrès des élèves, la surveillance de leur présence et aussi la mémorisation de leurs habitudes pour donner des conseils sur leur santé.

Dans le secteur des transports routiers, on cherche à rendre le trafic plus fluide, le transport plus efficace, plus sûr grâce aux technologies GPS de recherche du meilleur chemin, en évitant les problèmes routiers (travaux, accidents, …), en obtenant des informations sur l'itinéraire (route, temps, etc.), en utilisant le paiement électronique sur l'autoroute .De même, des nouvelles techniques de stationnement automatisées [28].

L'AmI est aussi employé dans le tourisme pour la personnalisation et l'adaptation des voyages pour différents touristes, ainsi que la prolifération des systèmes de vidéosurveillance.

2.3.Définition du contexte

Le contexte a été pris en compte dans les différents domaines de l'informatique, y compris le traitement du langage naturel, l'apprentissage machine, la vision par ordinateur, l'aide à la décision, la recherche d'information et l'informatique ubiquitaire. Par analogie au raisonnement humain, le but derrière l'utilisation du contexte est d'ajouter la capacité d'adaptation et la prise de la meilleure décision.

Dans la littéraire on trouve beaucoup de définition du contexte parmi lesquels on peut citer :

- Dey et Abowd [29], proposent la première définition générique du contexte. Ils précisent la nature des entités en question : le contexte couvre toutes les informations pouvant être utilisées pour caractériser la situation d'une entité. Une entité est une personne, un lieu, ou un objet qui peut être pertinent pour l'interaction entre l'utilisateur et l'application, y compris l'utilisateur et l'application. L'information peut concerner le lieu d'interaction, l'identité et l'état des groupes et des objets de l'environnement.

- contexte se réfère à «ce qui entoure le centre d'intérêt, fournit d'autres sources d'information et augmente la compréhension » [30].

En général dans la définition du contexte de nombreux auteurs ne font que regrouper une ou plusieurs des informations suivantes :

- localisation de l'utilisateur à un instant donné,

-caractéristiques sociales de l'utilisateur (identité, statut, fonction, groupe, communauté, appartenance, etc.)

- personnelles (intérêts, préférences etc.)

- état émotionnel et physiologique de l'utilisateur

- caractéristiques de l'environnement physique de l'utilisateur (objets, espace géographique, lumière et bruit ambiants),

- social (personnes se trouvant dans l'entourage de l'utilisateur, leur identité, etc.)

- l'heure et la date.

2.3.1- Informations du contexte

Les informations contextuelles sont classées en plusieurs catégories:

- Contexte de l'utilisateur (profil utilisateur, localisation de l'utilisateur, situation sociale, etc.).
- Contexte informatique (connectivité réseau, coûts de communication, bande passante de communication).
- Contexte physique (éclairage, bruit, etc.)

D'autres auteurs ont suggéré d'autres types de contexte citons comme exemple: l'emplacement, l'identité, le temps et l'activité, en plus les informations du contexte peuvent être déduites à partir d'autres informations [31].

2.3.2 Attributs de contexte

Les attributs de contexte désignent les informations définissants un élément du contexte, par exemple, la location d'activité, le nom de personne, la durée d'activité. Chaque attribut du contexte a au moins une valeur à un moment donné et cela en fonction de la valeur de plusieurs entités auxquelles l'attribut se rapporte.

Une entité est une instance d'une « personne, objet ou d'un lieu». Elle peut aussi être une activité ou un concept d'organisation (rôle, groupe, autre agent, etc.).

Une entité de contexte peut être décrite avec un ensemble d'attributs [32], tels que:

- ***Le type du Contexte***: Il se réfère à la catégorie du contexte tels que la température, le temps, la vitesse, etc. Ce type d'information peut être utilisé comme un paramètre pour une requête contextuelle ou un abonnement à un service.
- ***La valeur du contexte***: Elle désigne les données brutes recueillies par un capteur. L'unité dépend du type du contexte et le capteur appliqué. Dans la plupart des cas, le type du contexte et des valeurs du contexte ne sont pas suffisants pour établir un groupe de travail pour un système sensible au contexte.

- ***Horloge du contexte*** : Cet attribut contient le temps (date / heure) de détection de l'information du contexte. Il est nécessaire pour garder l'historique du contexte et pour gérer les conflits de détection.
- ***La Source du contexte*** : un attribut contenant des informations sur la source de l'information du contexte. Par exemple dans le cas d'un capteur matériel il contient l'identifiant du capteur utilisé par une application pour récupérer toutes ses informations.
- ***Le degré de Confiance***: L'attribut confiance décrit l'incertitude de ce type de contexte car les sources du contexte peuvent fournir des informations non précises, par exemple, les données de localisation sont imprécises et cela en fonction de l'outil utilisé dans la localisation.

2.3.3 Représentation de l'information du contexte

Afin de réaliser un système sensible au contexte il est essentiel d'utilisé un modèle efficace permettant le traitement, le partage et le stockage des données du contexte. Ils existent plusieurs techniques de représentation du contexte qui sont :

A. Modèles Clé-Valeur

Le modèle de pair *clé-valeur* est la structure de donnée la plus simple pour la modélisation de l'information contextuelle. L'approche de modélisation clé-valeur est très utilisée dans les frameworks de service distribué décrits sous forme d'une liste simple d'attributs clé-valeur.

Malgré que, les modèles clé-valeur sont faciles à gérer, ils n'ont pas assez de capacités pour une structuration sophistiquée pour réaliser des algorithmes efficaces d'extraction du contexte [34].

B. graphes contextuels

Un graphe contextuel est un graphe orienté acyclique avec un unique point d'entrée et un unique point de sortie [21]. Les nœuds du graphe représentent des actions simples ou parallèles, des nœuds contextuels, des nœuds de recombinaison ou bien encore des d'activités. Un chemin du graphe contextuel correspond à la solution d'un problème dans un contexte donné voir la Figure 2.1

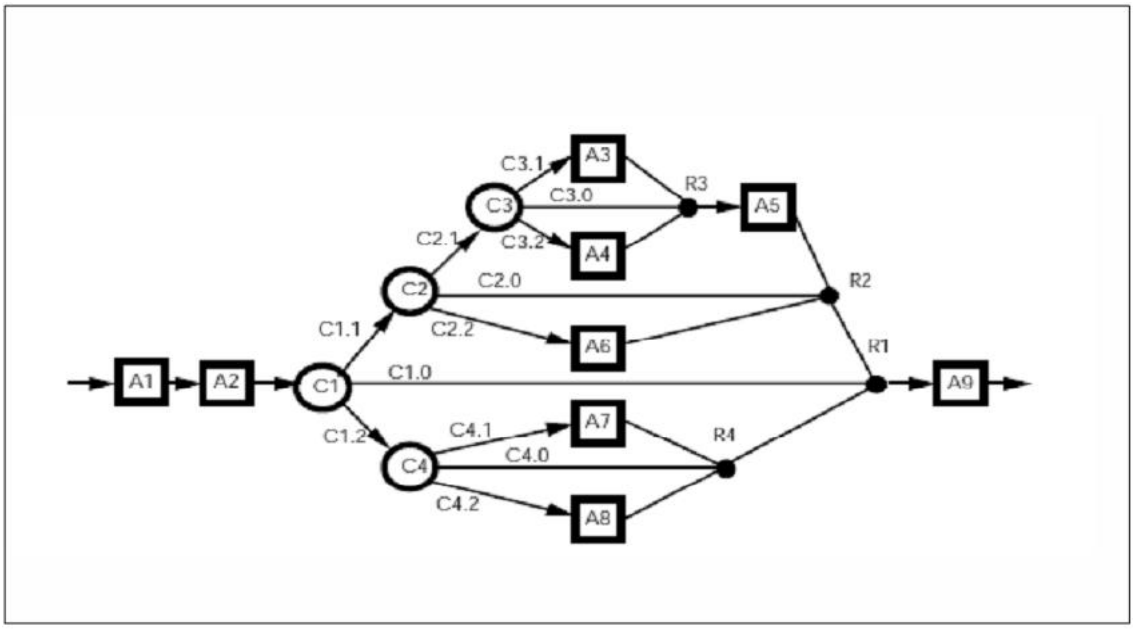


Figure 2.1 un exemple de graphe contextuel.

Les actions sont des méthodes exécutables alors que les activités représentent un assemblage

C. Modélisation par les langages de balisage (Markup Scheme Models).

Les langages de balisage sont des structures de données hiérarchiques constituées de balises avec attributs et contenus. Généralement, le contenu des balises est défini récursivement par d'autres balises. Le langage de balisage le plus utilisé et le plus populaire c'est le langage XML, il exit d'autres langages comme SGML, RDF, etc. [34].

D. *Contexteurs*

Un contexteur est une abstraction logicielle qui fournit la valeur d'une variable du contexte de l'environnement. Il comprend trois classes d'éléments Figure 2.2 :

– Les entrées de type données ou contrôle. Les premières correspondent aux informations du contexte, elles sont associées à des métadonnées qui expriment leurs qualités. Les entrées de contrôles permettent à d'autres contexteurs d'agir sur le fonctionnement interne d'un contexteur.

– Les sorties de type données ou contrôle. Les sorties de données sont les informations du contexte, associées à une métadonnée qualifiant la qualité de ces informations, destinées aux autres contexteurs ou à des applications. Les sorties de contrôle sont exclusivement destinées à d'autres contexteurs afin de leur transmettre des ordres.

– Le corps fonctionnel désigne la fonction remplie par le contexteur. Les auteurs proposent plusieurs classes de base du contexteur : contexteur élémentaire, contexteur à mémoire, contexteur à seuil, contexteur de traduction, contexteur de fusion et contexteur d'abstraction [33].

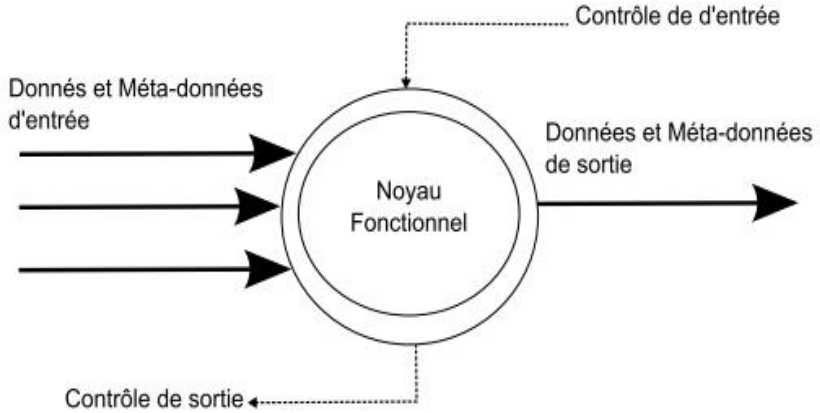


Figure 2.2 Modélisation du contexte avec les contexteurs

E. Modélisation Orientés Objet

Dans cette modélisation on utilise l'approche orienté objet afin de bénéficier de ses principaux avantages à savoir l'encapsulation et la réutilisation pour couvrir les problèmes posés par la dynamique du contexte dans les environnements ubiquitaires. Les détails du traitement du contexte sont encapsulés au niveau d'objet et sont cachées aux autres composants.

L'accès à l'information contextuelle est fourni via des interfaces spécifiées.

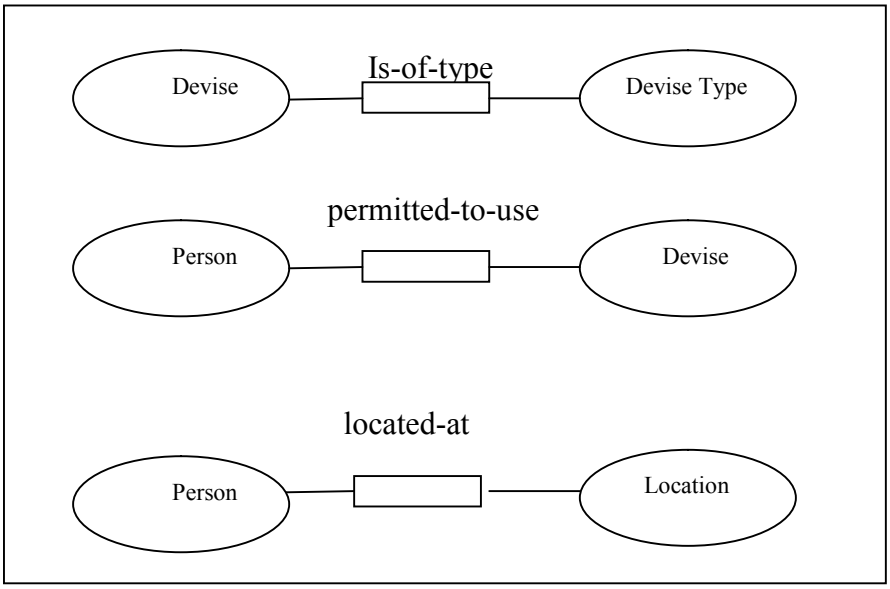


Figure 2.3 une modélisation du contexte avec l'approche Orienté Objet

F. Modèle Graphique

Parmi les outils de modélisation les plus connue est le langage de modélisation unifié (UML), qui a une forte composante graphique (diagrammes UML). Grâce à sa structure générique, UML convient

pour modéliser le contexte. Ceci est illustré par l'exemple de Bauer dans [35], où les aspects contextuels liés à la gestion du trafic aérien sont modélisés comme des extensions UML.

Garce à ce type de modèle on peut dériver un modèle Entité-Relation [12], ce qui est très utile comme instrument de structuration d'une base de données relationnelle dans un système d'information à base de gestion du contexte [34].

G. Modélisation formelle (Logic Based Models)

Dans cette modélisation on définit les conditions dans lesquelles une expression concluante ou des faits peuvent être inférés à partir d'autres expressions ou des faits. Ces conditions sont décrites par un ensemble des règles. Dans ce modèle formel, le contexte est défini comme des faits, des expressions et des règles. Par la suite les informations contextuelles seront ajoutées, mis à jour et supprimées à partir des règles logiques .

Comme tous les modèles logiques l'avantage principal de cette modélisation est qu'elle a un grand degré de formalité.

L'une des premières approches de modélisation du contexte basées sur la logique a été publiée au début de 1993 par McCarthy et son équipe de Stanford [37]. McCarthy a présenté le contexte comme des entités mathématiques abstraites avec des propriétés utiles en intelligence artificielle. La relation de base de cette approche est ***ist (c, p)***, qui affirme que la proposition ***p*** est vrai dans le contexte ***c***.

H. Modélisation à base d'ontologie.

Le terme ontologie a une longue histoire dans la philosophie, où il se réfère à l'objet de l'existence. Les ontologies représentent une connaissance complète ou partielle sur les concepts, leurs attributs et les relations entre ces concepts [36].

L'utilisation de l'ontologie pour la modélisation du contexte de l'utilisateur a plusieurs avantages comme :

-***Partage des connaissances :*** L'utilisation de l'ontologie permet à l'entité informatique telle que les agents et les services dans des environnements informatiques omniprésents d'avoir un ensemble commun de concepts sur le contexte, tout en interagissant avec les autres entités.

-Inférence logique. Sur la base de l'ontologie, un système sensible au contexte peut exploiter différents mécanismes de raisonnement logique et cela dans le but de déduire des informations du contexte de haut niveau à partir des informations du contexte de bas niveau, et à partir de l'information du contexte initiale (brute), ainsi que pour vérifier et résoudre les connaissances du contexte incompatibles en raison de la mauvaise détection de ces informations .

-Réutilisation des connaissances : grâce à la réutilisation des ontologies déjà existant, nous pouvons créer des nouvelles ontologies sans partir de zéro.

Toutes ces techniques de représentation du contexte ont leurs forces et leurs faiblesses. Le manque de généralité est la faiblesse la plus fréquente: certaines représentations sont adaptées à un type d'application et d'exprimer une vision particulière du contexte. Il y'a aussi un manque de bases formelles nécessaires pour capturer le contexte d'une manière cohérente et soutenir un raisonnement sur ses différentes propriétés.

Par ailleurs on peut conclure que la modélisation du contexte par les ontologies est la meilleure approche pour la modélisation du contexte grâce aux avantages qu'elles apportent par rapport aux autres représentations.

Les conclusions de l'évaluation présentée dans Strang et Linnhoff-Popien [37], reposant sur six exigences, montrent que les ontologies sont les modèles les plus expressifs et remplissent la plupart de leurs besoins.

Dans le Chapitre -3- on va présenter les ontologies en détaille et comment elles sont utilisées pour la modélisation du contexte.

2.3.4 Apprentissage et raisonnement dans les applications sensibles au contexte

Lorsque le contexte est modélisé par une approche formelle il peut être traité avec des mécanismes de raisonnement logique. Le raisonnement sur le contexte a deux utilités [16] :

- La vérification de la cohérence du contexte.
- La déduction du contexte de haut niveau (ou implicite) à partir du contexte de faible niveau (ou explicite).

Par exemple dans le scénario d'un téléphone intelligent (dans lequel ce dernier peut s'adapter avec les situations actuelles de l'utilisateur, en définissant des profils de préférences,) les utilisateurs peuvent personnaliser le comportement du téléphone mobile.

Lorsque l'utilisateur dort dans sa chambre ou qu'il est en train de prendre une douche dans la salle de bain, les appels entrants sont renvoyés vers la boîte aux lettres vocale; lorsque l'utilisateur cuit dans la cuisine ou regarder la télévision dans le salon, le volume de la sonnerie est activé ; lorsque l'utilisateur est en train de dîner avec la famille dans la salle à manger, le téléphone est réglé sur le mode vibration [16].

En évidence, le contexte de haut niveau ne peut pas être acquis directement par les capteurs mais il est inféré par le contexte à faible niveau comme l'emplacement physique et l'information environnementale [16].

L'inférence des nouvelles connaissances (par exemple le moyen transport) à partir des données détectées (par exemple la position GPS) est importante pour la sensibilité au contexte et l'adaptation aux changements du contexte de l'utilisateur [39].

Le traitement du contexte peut être divisé en agrégation et interprétation. La première réfère à la composition des informations du contexte brute. La deuxième réfère à l'abstraction de données du contexte pour les rendre lisibles par l'utilisateur.

L'inférence peut être faite par plusieurs techniques de raisonnement selon la technique de représentation du contexte. Par exemple on peut utiliser la technique de raisonnement à base de SPARQL ou la représentation du contexte est faite par OWL[39].

On peut utiliser aussi le technique de raisonnement à base d'apprentissage d'ontologies pour dériver de nouveaux faits données à partir d'une base de faits spécifiques si le contexte est représenté par une ontologie [39].

Les techniques d'apprentissage machine (*Machine learning techniques*) (par exemple réseaux bayésiens, la logique floue) peuvent être utilisées pour construire encore plus d'attributs du contexte de haut niveau (high-level context) à partir de contexte détecté. La combinaison de ces deux techniques peut être intéressante comme démontré dans [40].

Les systèmes expert (par exemple JESS, CLIPS) ou le raisonnement à base de règles peut être utilisé dans le raisonnement sur le contexte, on peut aussi déduire des nouvelles connaissances en utilisant la framework Jena [41] qui fournit le service d'inférence à base d'ontologie, et Jess (Java Expert System Shell) pour mettre en œuvre l'inférence en chaînage-avant.

2.3.5 Architecture des applications sensibles au contexte

Les systèmes sensibles au contexte peuvent être mis en œuvre par plusieurs manières. La démarche de mise en œuvre dépend des exigences et des conditions spéciales telles que l'emplacement des capteurs (locale ou distante), le nombre d'utilisateurs possibles (un seul utilisateur ou plusieurs), les ressources disponibles des appareils utilisés (haut de gamme-PC ou de petits appareils mobiles) ou de l'installation d'une nouvelle extension du système. En outre, la méthode de l'acquisition de données du contexte est très important lors de la conception des systèmes sensibles au contexte, car elle permet de prédéfinir le style architectural du système, il existe trois méthodes d'acquisition des données du contexte qui sont [38]:

- Accès directe aux capteurs (Direct sensor access)
- Utilisation des infrastructures intermédiaires (Middleware infrastructure) .
- Serveur du contexte (Context server).

Un système sensible au contexte contient en générale cinq couches comme illustré dans la Figure suivante :

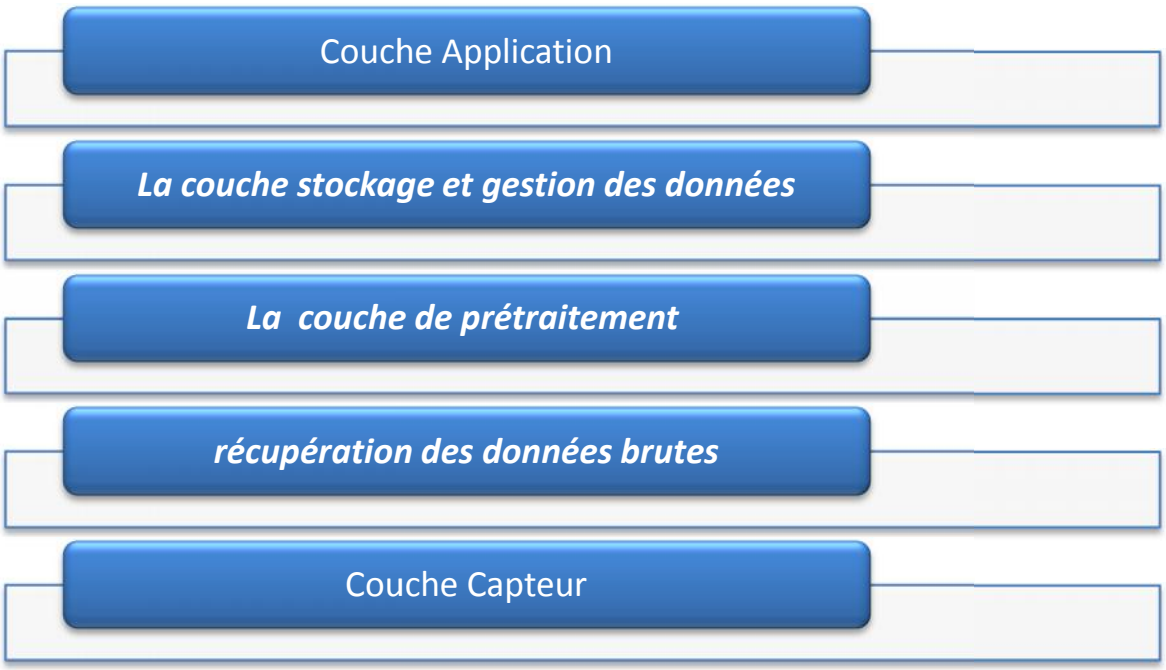


Figure 2.4 architecture d'un Système sensible au contexte

-La couche capteur (Sensors) :

Dans cette couche on trouve l'ensemble des capteurs chargés de récupérer les données brutes de l'environnement (par exemple dispositif utilisateur, réseau social). Les capteurs du contexte peuvent être classés en trois catégories [39] :

- Les capteurs physiques : ou capteurs matériels qui sont chargés de capter les mesures physiques comme la lumière, le son, l'emplacement, la température.
- Les capteurs virtuels qui sont chargés de capter les données des logiciels ou des services (par exemple, la détection des rendez-vous d'agenda);

- Les capteurs logiques : qui sont chargé de regrouper les informations provenant de différentes sources (combiner les informations des capteurs physiques ou virtuels avec d'autres sources telles que les bases de données) pour effectuer des tâches complexes.

-La couche récupération des données brutes (Raw data retriveral)

Cette couche fait appel à des APIs ou des protocoles spécifiques pour récupérer les données de la couche capteur. Ces requêtes doivent être autant que possible mis en œuvre d'une manière générique, ce qui rend possible le remplacement des capteurs (par exemple le remplacement d'un système RFID par un GPS un) [*39]*.

-La couche de prétraitement (Preprocessing):

La couche de prétraitement est responsable du raisonnement et de l'interprétation des informations contextuelles. Les capteurs interrogés dans la couche Capteur(Sensor) le plus souvent renvoient les données techniques qui ne conviennent pas à l'utilisation par les concepteurs d'applications, pour cela cette couche soulève les résultats des deux couches à un haut niveau d'abstraction.

Les transformations incluent les opérations d'extraction et de quantification. Par exemple, il transforme une position GPS à un poste comme à la maison ou au travail en plus cette information pourrait ne pas être de valeur pour une application, mais le nom de la salle de la personne est une information importante pour la même application [39].

Dans les systèmes sensibles au contexte qui ont plusieurs sources du contexte, les informations de contexte peuvent être combinées dans une seule information de haut niveau. Ce processus est également appelé «**agrégation**» ou «**composition**». Les informations d'un seul capteur est souvent pas importantes, alors qu'une information combinée de plusieurs capteurs pourrait être plus précieuse et précise.

De cette manière, un système est capable de déterminer, par exemple, si un client se trouve à l'intérieur ou à l'extérieur par l'analyse des différentes données physiques telles que la température et la lumière ou si une personne est actuellement assistant à une réunion et cela en capturant le niveau de bruit et l'emplacement.

-La couche de stockage et de gestion des données (Storage / Management) :

Cette couche organise les données recueillies et les offre au client par une interface public intermédiaire. Les clients peuvent les accéder de deux manières différentes « synchrones/ asynchrones ».

Dans le mode synchrone, le client interroge le serveur pour les modifications via des appels de méthodes distantes. Par conséquent, il envoie un message de demande de certains types de données et reste en attente jusqu'à ce qu'il reçoive la réponse.

Dans le mode asynchrone on utilise le mécanisme d'abonnement. Chaque client s'abonne à des événements spécifiques, qui les intéressent.

Dans la majorité des cas, l'approche asynchrone est plus appropriée en raison de l'évolution rapide du contexte.

-***La couche Application(Application)*** :

Cette couche représente l'endroit où les réactions aux changements du contexte sont mises en œuvre (par exemple, l'affichage du texte dans un contraste de couleur plus élevée si l'éclairage tourne mal).

2.4 Conclusion

Dans ce chapitre on a présenté la définition de l'intelligence ambiante, l'informatique sensible au contexte et la définition du contexte de l'utilisateur. De plus, on a étudié les différents modèles de représentation du contexte, les techniques de raisonnement sur le contexte, la description d'architecture des applications sensibles au contexte ainsi que le rôle de chaque couche. .

A partir de cette étude nous avons conclu que la modélisation du contexte par les ontologies est la meilleure approche de modélisation grâce aux avantages qu'elle apporte par rapport aux autres techniques de représentations.

Le prochain chapitre décrira en détaille la notion d'ontologie, son cycle de vie et les étapes à suivre pour bien modéliser le contexte par cette approche.

Chapitre 3

Les Ontologies et la modélisation de l'information de contexte

3.1 Introduction

Dans les premières phases de la recherche dans les systèmes ubiquitaires et l'informatique sensible au contexte, il y avait une grande faiblisse et beaucoup de limite dans la modélisation et le partage des connaissances. La cause de cette faiblesse est le fait que ces systèmes ne sont pas construits à base d'ontologies communes, avec une représentation sémantique explicite.

Les derniers chercheurs ont découvert un grand potentiel en utilisant les ontologies comme un moyen de représentation des connaissances de contexte de l'utilisateur.

Dans ce chapitre, nous allons évaluer la faisabilité d'utilisation des ontologies dans la modélisation des informations contextuelles. Nous décrirons aussi quelques exemples sur la façon d'utilisation du langage (OWL) pour la modélisation des différents types de contexte dans les environnements informatiques sensible au contexte.

3.2 Les Ontologies

C'est à partir des années 90 que les ontologies sont apparues dans le domaine de l'Ingénierie des Connaissances (IC), et plus largement en Intelligence Artificielle (IA), comme une approche de modélisation et de représentation des connaissances. Elles se sont introduites dans le cadre des démarches d'acquisition des connaissances pour les Systèmes à Base de Connaissances (SBC) et ont évolué vers la représentation des connaissances et leur organisation.

Durant ces dernières années, l'utilisation des ontologies pour la gestion des connaissances s'est avérée avantageuse dans le domaine de la recherche en intelligence artificielle où la gestion des connaissances est basée sur la représentation des connaissances de façon à simuler le raisonnement humain afin de modéliser les connaissances d'une façon utilisable par la machine [62]. Ainsi, les ontologies sont utiles pour construire des systèmes de gestion des connaissances. Elles permettent la représentation des connaissances et la modélisation des raisonnements qui sont des caractéristiques fondamentales des SBC. Elles ont pour rôle de fournir un système de concepts fondamentaux du domaine, afin de construire une base de connaissances. La conceptualisation permise par les ontologies représente en effet, une base solide sur laquelle sont construites des bases de connaissances partageables et utilisables. Une ontologie modélise à l'aide d'un formalisme approprié les connaissances spécifiques

d'un domaine d'application. Elle permet ensuite, la recherche et la restitution des connaissances dans des bases de connaissances.

3.2.1 Définition des ontologies

Le terme ontologie est un terme grec composé des mots «ontos» et «logos» qui veulent dire respectivement l'essence de l'être. Ce terme, hérité d'une tradition philosophique qui s'intéresse à la science de l'Etre, est apparu dans le domaine informatique grâce notamment au projet ARPA Knowledge Sharing Effort[71].

La première définition pour les ontologies dans le domaine de l'intelligence artificielle est donnée par Gruber : « une ontologie est une spécification explicite d'une conceptualisation. Le terme est emprunté de la philosophie, où l'ontologie est un ensemble de choses existantes. Pour les systèmes d'IA, ce qui existe est ce qui peut être représenté.».

Retenons qu'une ontologie est un vocabulaire commun qui définit le sens des concepts et les relations entre ces concepts. Ce vocabulaire peut être associé à un modèle qui décrit le contenu d'une base de connaissances, ses propriétés, la manière dont elle peut être utilisée ainsi que la syntaxe et les contraintes fournis par le langage de représentation. L'objectif est d'assurer la spécification explicite des connaissances au niveau conceptuel en utilisant un langage formel offrant une sémantique qui peut être plus ou moins rigoureuse, permettant une utilisation non ambiguë des connaissances du domaine.

3.2.2 Les composants d'une ontologie

Toutes les définitions citées dans [63] s'accordent sur le fait qu'une ontologie est formée par des concepts et des relations entre ceux-ci et qu'elle se réfère à un domaine. Ainsi, une ontologie peut être vue comme un treillis de concepts et de relations entre ces concepts destinés à représenter les objets du monde sous une forme compréhensible aussi bien par les hommes que par les machines. Si, certaines divergences relatives à la structure (degré de la formalisation) de l'ontologie ont été constatées, les composants d'une ontologie sont toujours les mêmes : une ontologie est constituée des concepts et des relations ainsi que des propriétés et des axiomes.

Les concepts : sont des notions (ou objets) permettant la description d'une tâche, d'une fonction, d'une action, d'une stratégie ou d'un processus de raisonnement, etc. Ils peuvent être abstraits ou concrets, élémentaires ou composés, réels ou fictifs. Habituellement, les concepts sont organisés en taxonomie. Une taxonomie est une hiérarchie de concepts (ou d'objets) reliés entre eux en fonction de critères sémantiques particuliers. A représenter un type d'interaction entre les concepts d'un domaine.

Les relations : sont les liens organisant les concepts de façon Elles sont formellement définies comme tout sous-ensemble d'un produit de n ensembles, c'est-à-dire R: C1xC2x … x Cn. Des exemples de relations binaires sont : (sous-concept-de, connecté-à, sorte-de, etc).

Les propriétés : (ou attributs) sont des restrictions des concepts ou des relations.

Les fonctions : sont des cas particuliers de relations dans lesquelles le nième élément de la relation est unique pour les n-1 précédents. Formellement, les fonctions sont définies ainsi, F: C1xC2x … x Cn-1, Cn. Comme exemple de fonction binaire, nous avons la fonction mère-de.

Les axiomes : de l'ontologie permettent de définir la sémantique des termes (classes, relations), leurs propriétés et toutes contraintes quant à leur interprétation. Ils sont définis à l'aide de formules bien formées de la logique du premier ordre en utilisant les prédicats de l'ontologie.

Les instances : sont utilisées pour représenter des éléments.

3.2.3 Les types d'ontologies

On distingue deux classes d'ontologies qui sont :

- Celle liée au type de la structure de la conceptualisation ;

- et celle liée au domaine à conceptualiser.

Concernant la première classe, l'ontologie se présente de façon différente selon le degré de formalisation du langage utilisé pour définir la signification des termes [64]. Une ontologie est dite hautement informelle dans le cas d'utilisation du langage naturel sans aucune restriction, semi-informelle lorsque le langage est de type langage naturel mais structuré avec un vocabulaire limité afin de restreindre les ambiguïtés. Si l'ontologie est représentée à l'aide d'un langage artificiel défini de façon formel tel qu'Ontolingua, l'ontologie est alors semi-formelle. Enfin, lorsque les termes possèdent une sémantique formelle dans un système tel que le calcul des prédicats du premier ordre comme c'est le cas de l'ontologie TOVE l'ontologie est dite rigoureusement formelle.

Dans la deuxième classe, l'ontologie se définit selon le domaine étudié et le degré de généralité ou de précision des connaissances représentées, les types d'ontologie les plus utilisées sont :

Les ontologies générales

Les ontologies générales portent sur des concepts généraux qui se veulent indépendants d'un domaine ou d'un problème particulier, tels que les concepts de temps, d'espace, de notion mathématiques, « elles sont prévues pour être utilisées dans des situations diverses et variées et peuvent servir une large communauté d'utilisateurs, ainsi elles peuvent être utilisées dans l'organisation des parties substantielles des connaissances humaines, comme la compréhension du langage naturel. » [63]. Plusieurs ontologies ont été développées pour décrire des concepts généraux ou des domaines particuliers. Parmi les ontologies générales il y a « Cyc » développée avec le modèle logique, en utilisant le langage CycL. Cette ontologie a la possibilité de construire des applications pour l'extraction des connaissances, la recherche intelligente et la traduction, etc.

Les ontologies de domaine

Ce type d'ontologies exprime des conceptualisations spécifiques à des ontologies de domaines particuliers tout en étant générique pour ce domaine. Ces conceptualisations mettent des contraintes sur la structure et les contenus des connaissances du domaine. De nombreuses ontologies de domaine ont été développées notamment dans le domaine médical UMLS.

Les ontologies d'application

Ces ontologies sont les plus spécifiques. Les concepts correspondent souvent aux rôles joués par les entités du domaine tout en exécutant une certaine activité [65]. Elles peuvent contenir des extensions spécifiques telles les méthodes et tâches. Elles contiennent toutes les définitions nécessaires pour décrire la connaissance requise pour une application particulière.

3.2.4 La construction d'une ontologie

Il existe trois méthodes possibles de création d'une ontologie. Une ontologie peut être construite d'une façon manuelle, automatique ou mixte.

Dans le mode manuel, les experts réalisent l'ontologie en s'appuyant sur des techniques classiques de collecte et d'analyse des connaissances comme c'est le cas pour les niveaux supérieurs des ontologies «Cyc » et « Wordnet ».

La création d'une ontologie d'une manière automatique se base sur des méthodes formelles et des techniques d'extraction des connaissances en employant des outils linguistiques et statistiques. Parmi les méthodes qui peuvent être utilisées, citons l'utilisation des méthodes de classification automatique issues de la théorie de l'information ou encore les méthodes de regroupement conceptuel (conceptual clustering) qui segmentent automatiquement les textes en unités thématiquement homogènes, et regroupent ces unités en fonction d'une mesure de similarité fondée sur la fréquence des mots.

Enfin, dans le mode mixte, les ontologies sont construites par des techniques automatiques tout en intégrant des méthodes permettant d'étendre des ontologies ayant été construites manuellement. Quel que soit le mode choisi, l'élaboration de toute ontologie doit s'appuyer sur un certain nombre de règles qu'il est nécessaire de respecter et une méthodologie de construction d'ontologies.

3.2.5 Les principes de construction d'une ontologie

Le processus de construction d'une ontologie doit respecter certains principes de bases qui permettent d'obtenir une ontologie susceptible de répondre aux objectifs de l'ontologie. Le concepteur de l'ontologie, se doit donc de garder à l'esprit ces principaux critères tout au long du cycle de développement de son ontologie.

- La cohérence, afin de pouvoir formuler des inférences cohérentes en offrant des définitions objectives ainsi que de la documentation associée en langage naturel.
- La clarté et objectivité : l'ontologie doit fournir le sens des termes définis en offrant des définitions objectives ainsi que de la documentation associée en langage naturel.
- L'exhaustivité : une définition exprimée par une condition nécessaire et suffisante est préférable à une définition exprimée seulement par une condition nécessaire ou par une condition suffisante.
- L'extensibilité monotone maximale : les nouveaux termes, qu'ils relèvent de la langue générale ou d'une langue de spécialité, devraient être inclus dans l'ontologie sans entraîner de modifications dans les définitions existantes.
- L'intervention ontologique minimale: intervenir le moins possible sur le monde en phase de modélisation. L'ontologie devrait spécifier le moins possible le sens de ses termes, de façon à ce que les parties impliquées dans l'ontologie aient les mains libres pour spécialiser et instancier l'ontologie à leur guise.

3.2.6 Méthodologies et cycle de développement d'une ontologie

L'ingénierie ontologique ne propose à l'heure actuelle, aucune méthode normalisée ou méthodologie générale de construction d'ontologies, ce qui rend le processus d'élaboration des ontologies long et coûteux. Cependant, certains auteurs ont proposé des méthodologies inspirées de leur expérience de construction d'ontologies.

Ces méthodologies proposent à travers un ensemble d'étapes, un cycle de développement d'ontologies qui peut être adopté lors de la construction d'une nouvelle ontologie.

Il en ressort que le cycle de développement général d'une ontologie se déroule en quatre principales étapes qui sont:

1. l'évaluation des besoins,

2. la conceptualisation,

3. la formalisation (appelée également ontologisation),

4. l'implémentation.

Ces étapes sont généralement précédées par une étape d'évaluation des besoins et de délimitation du domaine de connaissances à modéliser.

L'évaluation des besoins :

L'évaluation des besoins a pour objectif de cerner le but visé par la construction de l'ontologie. Cette évaluation des besoins, se décline en trois aspects qui correspondent à l'objectif opérationnel de l'ontologie, le domaine de connaissances à modéliser et les utilisateurs potentiels. En effet, il est indispensable de bien préciser l'objectif opérationnel de l'ontologie, en particulier à travers des scénarios d'usage [66]. Aussi, le domaine de connaissances doit être délimité aussi précisément que possible, et découpé si besoin en termes de connaissances du domaine, connaissances de raisonnement et connaissances de haut niveau. La délimitation du domaine de connaissances repose sur l'utilisation de ressources textuelles et/ou multimédia, constituant le corpus du domaine et au travers desquelles peuvent être appréhendées la terminologie du domaine et les significations des concepts.

La conceptualisation

La conceptualisation est un processus d'abstraction qui consiste à identifier les concepts essentiels du domaine de connaissances et d'établir les relations entre ces concepts, au sein d'un corpus représentatif du domaine.

Il s'agit donc de décrire le domaine de connaissances grâce à des concepts plus ou moins précis et aux relations qui peuvent exister entre ces concepts. L'identification des concepts et relations peut se faire selon l'analyse des textes (documents, notes, comptes rendus d'interviews, etc.) Cette analyse est généralement « une analyse informelle des textes qui peut être doublée par une analyse automatique permettant de détecter les termes et structures sémantiques (définitions, règles) présentes dans le corpus de documents » [68].

Néanmoins, cette analyse ne peut suffire à elle seule à spécifier sémantique du domaine, car certaines connaissances ne prennent sens que lorsqu'elles sont lues par un expert ou un spécialiste du domaine.

La sémantique doit donc être précisée ou validée par les experts du domaine. Ces derniers peuvent soit contribuer directement à la construction de l'ontologie, soit fournir des spécifications conceptuelles permettant de contraindre la sémantique exprimée. De plus, les experts peuvent faciliter l'indication des instances connues des concepts, de leurs propriétés, leur sens ainsi que les contraintes éventuelles. Celles-ci sont établies en collaboration avec le concepteur de l'ontologie. Par exemple, en décidant qu'une notion est une instance, on pose la contrainte qu'une instance ne peut pas être un concept. Pour identifier les concepts, [64] suggère trois stratégies qui se déclinent en :

- une approche descendante, où il s'agit de partir des concepts les plus généraux et de les spécialiser par la suite ;
- une approche ascendante qui consiste à considérer tous les termes spécifiques et ensuite à trouver les termes génériques associés.
- une approche intermédiaire dans laquelle les concepts se structurent autour des concepts importants du domaine (ni trop généraux, ni trop spécifiques). Ces concepts centraux sont ensuite reliés avec les concepts proches soit par spécialisation soit par généralisation.

Au cours de ce processus, les connaissances sont structurées grâce notamment à l'utilisation des relations et aux définitions spécifiant le sens associé à chaque concept. « La conceptualisation structure les connaissances identifiées, c'est à dire, les concepts, instances et relations en représentations intermédiaires semi-formelles, comme le dictionnaire des données, les arbres de classification de concepts, et bien d'autres » [63].

Ainsi, le processus de la conceptualisation mène à l'élaboration d'un modèle conceptuel qui décrit, au travers des éléments terminologiques et sémantiques les connaissances du domaine. Ce modèle conceptuel peut être complètement informel (exprimé en langage naturel et présenté sous forme de tableau par exemple) ou semi-formel, combinant langage naturel et propriétés formelles telle que les diagrammes de classes UML. Dans Fürst la structuration des connaissances, le consensus ontologique et la formalisation de l'ontologie font partie d'un processus appelé l'ontologisation.

La formalisation

La formalisation concerne le modèle conceptuel obtenu à l'étape précédente. Elle peut être vue comme une représentation explicite et formelle de la conceptualisation. Elle est réalisée par le biais d'un langage formel ou formalisme qui est un ensemble de composants sémantiques (contenu), de règles structurelles (mode d'emploi) et d'une notation formelle particulière (forme) destinée à organiser les relations entre les éléments constituant l'ontologie. L'objectif de l'utilisation d'un langage de formalisation d'ontologies, est de permettre d'une part de réduire les ambiguïtés du langage naturel en offrant une plus grande expressivité ; et d'une autre part de rendre l'ontologie compréhensible par les machines.

« Les formalismes offrent un support formel à la composition des concepts et à leur comparaison» [66].

Différents formalismes tels que les logiques de description, les réseaux sémantiques et les frames (schémas) peuvent être employés pour représenter formellement une ontologie.

- Les logiques de description représentent la connaissance sous forme de propositions ou affirmations sur le domaine.
- Les réseaux sémantiques, tout en gardant cette approche propositionnelle, tiennent compte de la structure et les relations entre ces propositions.
- Les frames représentent le domaine en termes de ses objets et leurs propriétés et relations.

Ces paradigmes se différencient les uns des autres par leur formalisme de représentation des connaissances et leurs mécanismes d'inférence permettant de raisonner sur les représentations.

A l'issue de l'étape de formalisation, on obtient une ontologie formelle. Cependant, certaines connaissances du domaine peuvent être abandonnées, du fait de l'impossibilité de lever certaines ambiguïtés, ou du fait des limitations de l'expressivité du langage utilisé.

L'implémentation

L'implémentation consiste à outiller une ontologie pour permettre à une machine de manipuler des connaissances du domaine via cette ontologie. La machine doit donc pouvoir utiliser des mécanismes opérant sur les représentations de l'ontologie. Or, si beaucoup de langages réifiant les modèles cités précédemment autorisent l'expression des inférences des connaissances, peu sont outillés pour rendre possible la manipulation de ces connaissances. Les graphes conceptuels sont des modèles qui font exception car, la représentation des connaissances sous forme de graphes permet de mettre en œuvre des raisonnements par des opérations formelles sur les graphes (comparaison, fusion, etc.). La façon de mener ces opérations dépend cependant de l'objectif opérationnel du système envisagé.

Dans le cas où le formalisme utilisé n'est pas opérationnel, il est nécessaire, soit d'outiller ce langage, dans la mesure du possible, soit de transcrire l'ontologie dans un langage opérationnel. Mais certains langages offrent des possibilités de raisonnement limités qui peuvent convenir à certaines applications limitées. Par exemple, les langages à base de frames et les logiques de description permettent de savoir si une connaissance donnée, ou une connaissance plus spécifique qu'une connaissance donnée, est présente dans une base de connaissances en utilisant la relation de subsomption. Dans le cas d'un simple système de stockage et de consultation de connaissances, de tels langages sont donc suffisants. Une fois l'ontologie opérationnalisée, elle peut être intégrée en machine au sein d'un système manipulant le modèle de connaissances utilisé via le langage opérationnel choisi. Mais avant d'être livrée aux utilisateurs, l'ontologie doit être testée par rapport au contexte d'usage pour lequel elle a été conçue.

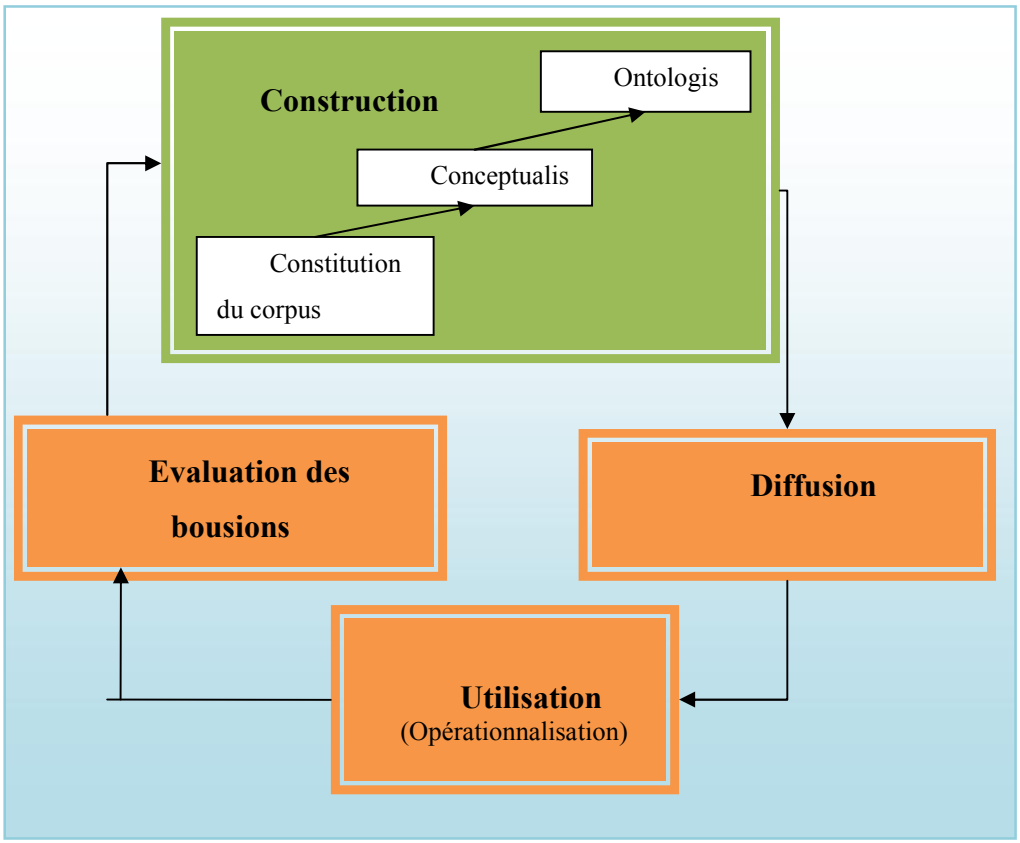


Figure 3.1 Processus de développement Ontologie

3.2.7 Langages de spécification d'ontologies

Plusieurs langages de spécification d'ontologies (ou langage d'ontologies) ont été développés pendant ces dernières années, et ils deviendront sûrement des langages d'ontologie dans le contexte du Web sémantique. Certains d'entre eux sont basés sur la syntaxe de XML, tels que XOL (Ontology Exchange Language), SHOE (Simple HTML Ontology Extension - qui a été précédemment basé sur le HTML), OML (Ontology Markup Language), RDF (Resource Description Framework), RDF Schéma. Les 2 derniers sont des langages créés par des groupes de travail du World Wide Web Consortium (W3C). En conclusion, trois langages additionnels sont établis sur RDF(S) pour améliorer ses caractéristiques: OIL (Ontology Inference Layer), DAML+OIL et OWL (Web Ontology Language).

La Figure 3.2 représente les rapports principaux entre tous ces langages sous la forme d'une pyramide des langages. [59]

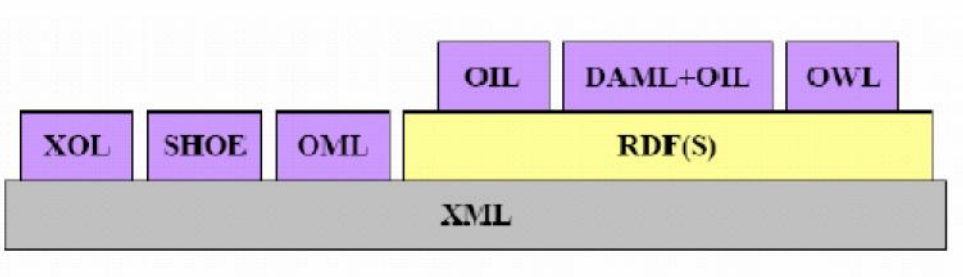


Figure 3.2 La pyramide des langages. [59]

3.2.7.1 XML (eXtended Markup Language) et XML Schema

XML est un langage de description et d'échange de documents structurés, issu de SGML (Standard Generalized Markup Language) et défini par le consortium Web. XML permet de décrire la structure arborescente de documents à l'aide d'un système de balises permettant de marquer les éléments qui composent la structure et les relations entre ces éléments. XML ne pose aucune contrainte sémantique sur la description des informations, il ne constitue donc pas un langage de modélisation d'ontologies à lui seul. Avec XML nous pouvons décrire une Personne Toto âgée de 37 ans qui habite 12 rue des pins.

```
<Personne id='Toto'>
<Adresse>12 rue des pins</Adresse>
<Age>37</Age>
</Personne>
```

XML Schema (XML-S) est un outil de définition de grammaires caractérisant des arborescences de documents (notion de validité syntaxique). Avec les schémas XML, il est possible de contraindre la structure arborescente d'un document mais pas la sémantique des informations contenues dans ce document [60].

3.2.7.2 RDF (Resource Description Framework) et RDF Schema

RDF est un modèle pour la représentation de métadonnées à propos de ressources. Cette représentation est faite sous la forme d'un triplet :

- Sujet : la ressource que l'on définit ;
- Prédicat : la propriété de la ressource, qui est une liaison étiquetée et orientée du sujet vers l'objet ;
- Objet : la valeur de la propriété pouvant être une autre ressource ou bien un littéral.

RDF est à la base exprimé sous la forme d'un graphe orienté mais on peut le décrire par la syntaxe XML. On parle alors de RDF/XML qui par abus de langage peut être appelé RDF. En prenant l'exemple défini pour XML, nous pouvons définir avec RDF (Figure 3.3) un graphe (Figure 3.4**)**.

Si RDF fournit une capacité d'échange de connaissances, il ne permet pas à l'utilisateur de définir le vocabulaire des termes à utiliser, ni d'établir la sémantique des objets utilisés

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

Figure 3.3 Représentation RDF /XML de Toto 37 ans qui habite au rue des pins. [60]

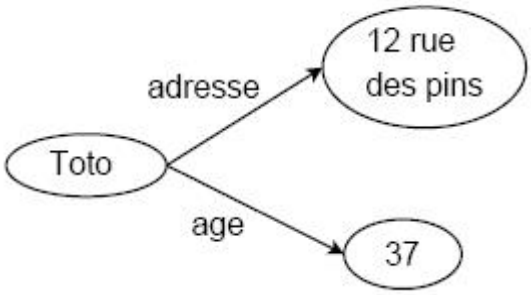


Figure 3.4 Représentation graphique de Toto 37 ans qui habite au rue des pins. [60]

RDF Schéma ou RDFS est un langage permettant de définir des propriétés sémantiques pour les ressources par un schéma. Dans un schéma on peut définir de nouvelles ressources comme des spécialisations d'autres ressources. Les schémas contraignent aussi le contexte d'utilisation des ressources. Avec RDFS de nouvelles notions sémantiques apparaissent. La principale est la distinction entre une classe (concept d'une ontologie) et une instance (individu d'une ontologie). Voici d'autres notions définies dans RDFS :

- La notion de classe (rdfs : Class) qui peut être rapprochée de la notion de concept d'une ontologie.
- La notion de sous-classe (rdfs : subClassOf) qui est une spécialisation d'une classe déjà définie.
- La notion de type (rdf :Type) : les instances d'une classe propriété et sous propriété (rôle de l'ontologie)
- Les notions de source (rdfs :domain) et de cible (rdfs :range) d'une propriété.

La hiérarchie de subsomption ou taxinomie peut être décrite en utilisant la propriété rdfs :subClassOf. Cette hiérarchie définit les relations de spécialisation d'une classe Ressource par rapport à une classe Objet. Cette relation s'applique également aux relations par la propriété rdfs :SubPropertyOf.

RDF et RDFS permettent de définir sous la forme d'un graphe de triplets des données ou des méta-données. Cependant, il est impossible de raisonner sur ces représentations car la sémantique reste très limitée. C'est ce manque de sémantique qu'OWL comble par l'apport d'un vocabulaire plus riche. [60]

3.2.7.3 OWL

Le standard proposé par le W3C pour la représentation d'ontologies, OWL (Web Ontology Language), a été imaginé comme une extension de RDF, dans le sens où, pour définir les domaines d'application, OWL utilise les primitives de base modélisées par les schémas RDF (les classes et les propriétés). A ceux-ci OWL ajoute d'autres structures de représentation dérivées des logiques de description qui augmentent l'expressivité. De cette façon, les grandes limitations de RDF (on peut citer l'absence de la notion de modularité, la portée globale plutôt que locale des propriétés, l'impossibilité de déclarer des classes disjointes, des définitions de classes complexes à l'aide des expressions ensemblistes, et des restrictions de cardinalité) sont palliées.

Les schémas RDF possèdent des primitives puissantes (*rdfs : Class* – la classe de toutes les classes et *rdfs :Property* – la classe de toutes les propriétés), qui font qu'une meilleure expressivité obtenue par une simple extension de RDF Schema a pour conséquence l'indécidabilité des raisonnements. Le W3C aborde cette question en définissant trois sous langages : OWL Lite, OWL DL et OWL Full, d'une expressivité croissante, permettant ainsi de répondre aux besoins variés des applications. [61]

3.3 La représentation des connaissances par les ontologies

De nombreuses ontologies ont été créées dans le but de représenter les connaissances dans des systèmes informatiques. Ces représentations formelles expriment les connaissances sous forme logique. Le travail de représentation des connaissances consiste dans un premier temps à construire une première modélisation partiellement cohérente, correspondant à une conceptualisation semi-formalisée.

Le résultat obtenu est alors un modèle conceptuel, semi-formel, dont il est nécessaire de le traduire dans un langage formel et opérationnel. Ce langage doit se doter d'un mécanisme de raisonnement et permettre ainsi de représenter les différents types de connaissances (connaissances terminologiques, faits, règles et contraintes).

L'objectif étant de rendre l'ontologie opérationnelle de façon à manipuler les connaissances représentées à travers des mécanismes adaptés à l'objectif du système conçu. Il existe plusieurs niveaux d'abstraction et de représentation des connaissances et par conséquent différents formalismes. Dans le

domaine de l'IA, de nombreux formalismes de représentation de connaissances sont proposés. Nous décrivons ci-après, quelques-uns de ces formalismes.

Les logiques de description

Les logiques de description (Descriptions Logics (DLs)) est une famille des langages de représentation des connaissances, elle peut être utilisé pour représenter les connaissances des applications de domaine d'une manière structurée et bien formée. Le nom DL est bien motivé par le fait, que par une seul fois, on peut décrire les notions importants de domaine par des concepts (descriptions).les expressions sont formé par des concepts atomiques (prédicats unaire) et rôles atomiques (prédicats binaires). Utilisant les constructeurs des concepts et rôles fourni par la DL. La DL diffère a l'autre formalisme (RS, Frames) par le fait quel est équipé par une logique formel a base du sémantique.

L'exemple ci-dessous illustre les bases de la DL :

Eléphant : c'est un mammifère de couleur grise possédant une trompe (partie du corps)

Concepts : Mammifere, Trompe, Gris

Relations (ou Rôles) : partie du corps, couleur

Constructeurs : Π, ▫ , ▫

Mammifère Π ∃partieducorps.Trompe Π ∀couleur.Gris

Dans le cas où une ontologie est formalisée à l'aide des logiques de description, cette ontologie permet de décrire les concepts d'un domaine à travers des concepts atomiques correspondant à des prédicats unaires spécifiant les objets du domaine; et des rôles atomiques, correspondant à des prédicats binaires et décrivant les relations entre les objets (concepts du domaine). Ces rôles sont spécifiés à l'aide de constructeurs fournis par le langage formel de description logique.

Ainsi, on distingue deux niveaux de représentation des connaissances: le niveau terminologique et le niveau factuel qui ont donné naissance aux notions de T-Box et A-Box, que l'on retrouve dans la plupart des logiques de description:

- La T-Box (Terminological Box) correspond au niveau descriptif qui permet de décrire les concepts en fonction d'autres concepts à partir des relations et des contraintes sur ces relations. Elle renferme les connaissances terminologiques.

Concepts : Personne, Homme, Femme, Pere, Mere, Frere, Soeur, Oncle, Tante.

Oncle = Homme Π (avoir_pour_frere.Pere avoir_pour_soeur.Mere)

Relations : avoir_pour_enfant, avoir_pour_frere, avoir_pour_soeur, sexe.

- La A-Box (Assertional Box) correspond au niveau des assertions, et est réservée à la description et la manipulation des individus.

Pierre : Homme

Sophie : Mere

Jean : (Homme Π avoir_pour_soeur Sophie)

=> Jean : instanceDe Oncle

=> Sophie : instanceDe (Mere Π Soeur)

Concernant le raisonnement permis par les logiques de description, c'est un raisonnement déductif qui essaie d'expliciter les informations implicites contenues dans une base de connaissances ; vérifier si une assertion est vraie dans un univers représenté par une théorie (l'ensemble de formules de la base), consiste à vérifier si la formule qui représente l'assertion est une conséquence logique de cette théorie. Le raisonnement logique est associé à la technique logique des prédicats du premier ordre. Une base de connaissances en logique est constituée exclusivement d'un ensemble de formules logiques bien formées décrivant l'univers du discours. Par exemple, les expressions suivantes sont des formules bien formées d'un langage contenant les prédicats à un paramètre.

homme et voyage et la constante Pierre :

1) homme (Pierre)

2) x homme(x) => voyage(x)

De plus, la classification constitue un mécanisme d'inférence important des logiques de description. La classification est un processus permettant de déterminer la position d'un concept donné dans la hiérarchie de subsomption. Elle consiste à placer un concept dans la hiérarchie en spécifiant les concepts qui le subsument et les concepts qui sont subsumés par lui.

Réseaux sémantiques

Un graphe sémantique est un graphe composé d'un ensemble de nœuds, qui représente des concepts d'entité, attribut, objet, événement, état, etc., et un ensemble d'arcs orientés, étiquetés qui relient

deux nœuds en représentant les relations binaires entre les concepts. Nous pouvant structurer les connaissances dans une hiérarchie des concepts par les relations de « sorte-de » et « est-un » qui représente le lien de spécialisation de concepts.

Frames

Le frame est une structure dynamique représentant des situations prototypiques qui contient des informations sur une situation ou un objet standard et prend en compte toutes les formes possibles d'expression de la connaissance.

Un frame est composé d'un ensemble d'attributs qui sont les propriétés caractérisant le concept. Ces attributs contiennent des facettes qui décrivent l'ensemble des valeurs possibles pour cet attribut.

Deux mécanismes de raisonnement sont assurés par les frames : le filtrage et la classification.

Graphes conceptuelle

Les réseaux sémantiques donnèrent lieu à de nombreux modèles de représentation des connaissances parmi lesquels les graphes conceptuels. Ainsi, le modèle des graphes conceptuels est un modèle de représentation de connaissances du type réseaux sémantiques fondés sur la logique, développés par [69].

Le sens d'un concept se réduit a sa position relative par rapport aux autres concepts dans le réseau sémantique modélisant les connaissances de système. A la différence des réseaux sémantiques représentant plusieurs propositions et définitions de concepts dans un même réseau, les graphes conceptuels organisent les différents types d'élément dans des structures déférentes avec la hiérarchie de spécialisation des graphes. Cela permet de manipuler et vérifier chaque structure par des règles de cohérences.

Chaque graphe conceptuel possède également une représentation équivalente en logique des prédicats grâce au format d'échange de connaissances KIF (Knowkledge Interchange Format).

3.4. Ontologies et raisonnement

La représentation des connaissances par les ontologies peut s'accompagner des mécanismes de raisonnement. Le raisonnement concerne la manipulation des connaissances déjà acquises pour produire de nouvelles connaissances. Il utilise des mécanismes d'inférence qui permettent la résolution des problèmes pour lesquels il n'existe pas de procédures explicites dans le programme. Différents

mécanismes de raisonnement sont utilisés selon les objectifs du système à mettre en place : raisonnement logique, raisonnement par classification, filtrage, héritage et raisonnement à base de règles.

Dans la suite de ce chapitre on va présenter quelques mécanismes de raisonnement.

Le raisonnement logique

Le raisonnement logique se base sur un mécanisme de déduction qui utilise un ensemble de règles d'inférence pour déduire des nouveaux faits à partir des faits connus. Ces règles, le modus ponens, le modus tollens.

La règle de modus ponens affirme que si un fait entraîne un deuxième et que le premier est vrai, alors le deuxième est vrai aussi. Appliquer le modus ponens aux formules 1) et 3) donne 4) :

Prémisses : 1) homme (Pierre)

3) homme (Pierre) 2) voyage(Pierre)

Conclusion : 4) voyage (Pierre)

La logique fournit un formalisme clair et non ambigu. Cette clarté vient d'une part du fait que la signification d'une formule ne dépend que de sa structure et de la signification donnée à ses composants atomiques et d'autre part du fait que le langage d'expression logique est proche du langage naturel [70]. De plus les connecteurs logiques (et, ou, implication et négation) et les quantificateurs permettent une riche description du monde. Les inférences faites avec la logique du premier ordre sont correctes, complètes et fondées.

Le raisonnement à base de règles

Le raisonnement à base de règles est également un mécanisme de raisonnement sur les connaissances. L'élément de base des systèmes à base de règles est la règle de production ; une règle a la forme suivante

SI <condition> ALORS <action>

La partie condition est exprimée par un prédicat logique correspondant à une affirmation sur la base de connaissances qui doit être vraie au moment de valider la règle pour que l'action soit déclenchée ; la partie action, qui est la partie exécutable de la règle indique des ajouts ou modifications à faire à la base.

Un système à base de règles comporte trois parties: une base de règles, un contexte ou base de faits et un moteur d'inférence.

Les systèmes à base de règles permettent en général de bien résoudre les problèmes de causalité ou de diagnostic traitant des objets simples. Ils offrent un cadre déclaratif pour exprimer des connaissances procédurales, de "savoir-faire", ce qui permet de voir clairement les conditions dans lesquelles une règle est applicable. Pour répondre à une question, le moteur d'inférence suit un cycle de détection des règles applicables, de choix de la règle à déclencher et d'exécution de l'action associée à cette règle. Le moteur d'inférence peut fonctionner dans deux modes différents, chaînage avant et chaînage arrière.

3.5 Utilisation des ontologies dans la modélisation de contexte de l'utilisateur

Une approche de modélisation du contexte basée sur l'ontologie est le meilleur choix par rapport aux autres approches et cela pour les avantages qu'elle présente, parmi lesquelles on trouve [8] :

- ***Expressivité*** : une ontologie est modélisée par une approche orientée objet, avec Expressivité puissante entraîné par leur classe / propriété des constructeurs et axiomes. Par conséquent, elle est plus expressive que les modèles existants et nous permet de capturer plus de caractéristiques de différents types de contexte.

-***Partage des connaissances*** : l'utilisation de l'ontologie permet aux entités informatiques telles que les agents et les services dans des environnements informatiques sensibles au contexte d'avoir un ensemble commun de concepts sur le contexte, afin de bien interagir entre eux.

- ***L'inférence logique*** :en utilisant les ontologies, une application sensible au contexte peut exploiter différents mécanismes d'inférence logique pour déduire le contexte de haut niveau à partir du contexte de faible niveau, et pour vérifier et résoudre les connaissances du contexte incompatibles à cause de mauvaise détection d'information de contexte.

-***Réutilisation des connaissances*** : En réutilisant les ontologies des différents domaines (par exemple, l'ontologie temporelle et spatiale), nous pouvons composer une ontologie de grande envergure sans partir de zéro.

-***Extensibilité*** : les concepts dans l'ontologie de contexte sont organisés sous forme des taxinomies ou des hiérarchies, les nouveaux concepts peuvent être facilement ajoutés à l'ontologie existante de manière hiérarchique.

En plus l'utilisation de l'ontologie offre des grandes opportunités pour les systèmes sensibles au contexte, une de ses possibilités est d'explorer la puissance de langages de description d'ontologie OWL pour développer des modèles de contexte formels et expressifs.

D'un point de vue formel, les deux versions du langage OWL (DL et Lite) peut être considérée comme équivalente à logiques de description (SHOIN(D) et SHIF(D) [8].

3.6 Exemple des ontologies de contexte

Dans la modélisation de contexte, certains types de contexte est plus important que les autres dans un domaine spécifique. Apres la lecture des différents littératures , nous avons constaté que les entités **Emplacement**(Location), **Utilisateur**(user), A**ctivité** (Activity)et de les **ressources de calcul**(computational) sont les entités contextuelles les plus fondamentaux pour capturer l'information de contexte dans un environnement intelligent[8].

Ces entités contextuelles ne forment seulement le squelette principal du contexte, mais agissent aussi comme des indices dans les autres informations associées. Les objectifs de la conception du modèle de contexte comprennent la modélisation d'un ensemble d'entités de haut niveau et offrir une extensibilité flexible pour ajouter des concepts spécifiques dans différents domaines d'application.

Dans les environnements informatiques ubiquitaires, les applications et les services sont généralement regroupés en une collection de sous-domaines (par exemple, à la maison, au bureau ou véhicule), le contexte dans ces domaines partage des concepts communs qui peuvent être modélisés à l'aide d'un modèle de contexte général, avec des caractéristiques spécifiques. Par exemple, les modèles de localisation définies pour un service de navigation automobile et le service d'une maison intelligente partagent en général les mêmes concepts, comme les relations spatiales entre les différents sites.

Par conséquent, la séparation des domaines d'application encourage la réutilisation des concepts généraux, et fournit une interface flexible pour définir des connaissances spécifiques.

En basant sur le concept de séparation des domaines on divise le modèle de contexte en deux niveaux ***l'ontologie de haut niveaux*** (***upper ontology***) et l'***ontologie spécifique (specific ontology).*** L'ontologie de haut niveau reflète les caractéristiques générales pour les entités contextuelles de base et l'ontologie spécifique est un ensemble d'ontologie qui définit les détails des concepts généraux et leurs caractéristiques dans chaque sous-domaine[8].

La Figure 3.5 montre l'ontologie de contexte d'haut niveau du projet **SOCAM [42]**, Le modèle de contexte est structuré par un ensemble d'entités abstraites, chacune décrivant un objet physique ou conceptuel, y compris la **Personne(Person)**, **l'Activité(Activity)**, **l'entité informatique** (**CompEntity**) et **Emplacement(Location)**, ainsi que d'un ensemble de sous-classes abstraites.

Chaque entité est associée à ses attributs (représentés par '***OWL :DatatypProperty*** ') et ses relations (représentés par '***OWL : ObjectProperty*** '). La propriété OWL '***OWL: subClassOf'*** permet une structuration hiérarchique des entités sous-classe, afin d'ajouter des concepts supplémentaires dans un domaine spécifique, la Figure 3.6 montre une sérialisation partielle de l'ontologie en langage OWL.

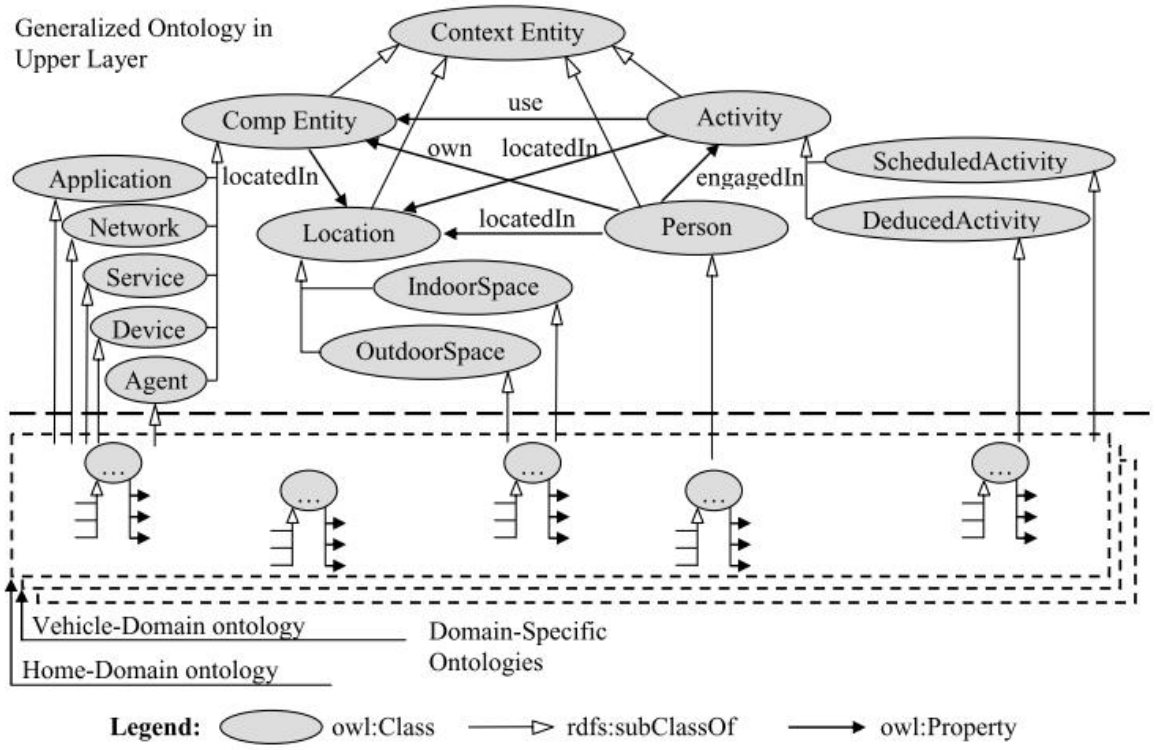


Figure 3.5 Exemple d'une ontologie de contexte d'haut niveau du projet **SOCAM.**

```
<owl:Class rdf:ID="ContextEntity"/>
<owl:Class rdf:ID="Location">
<rdfs:subClassOf rdf:resource="#ContextEntity"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="longtitude">
<rdf:type rdf:resource="FunctionalProperty">
<rdfs:domain rdf:resource="Location">
<rdfs:range rdf:resource="xsd:double">
</owl:ObjectProperty>
<owl:Class rdf:ID="IndoorSpace">
<rdfs:subClassOf rdf:resource="#Location"/>
<owl:disjointWith rdf:resource="#OutdoorSpace"/> </owl:Class>
<owl:ObjectProperty rdf:ID="locatedIn">
<rdf:type="owl:TransitiveProperty"/>
<rdfs:domain rdf:resource="#Entity"/>
<rdfs:range rdf:resource="#Location"/>
<owl:inverseOf rdf:resource="#contains "/>
</owl:ObjectProperty>
```

Figure 3.6 une sérialisation partielle de l'ontologie en langage OWL

L'architecture ***CoOL*** (Ontology Language Contexte) [43] adopte une architecture en deux couches

- ***CoOL Core*** : basé sur plusieurs langages: OWL, DAML + OIL, F-Logic

- ***CoOL Integration*** : une collection de schéma et des extensions de protocole.

Les auteurs de cette architecture pensent que le choix de plusieurs langages pour la représentation des connaissances est favorable pour les développeurs, qui peuvent utiliser n'importe quel langage qu'ils jugent approprié.

Une autre ontologie proposée pour la modélisation de contexte intitulé ***SOUPA*** (Standard Ontology for Ubiquitous and Pervasive Applications) cette ontologie exprimé en langage OWL. Comme dans le projet ***SOCAM***, ***SOUPA*** contient deux ensembles d'ontologies:

- ***SOUPA Core*** : définissant le vocabulaire général pour différentes applications informatiques ubiquitaires.

- ***SOUPA Extension :*** qui définissent les autres concepts utilisés dans les applications spécifiques.

La stratégie utilisée par les développeurs de SOUPA pour sa mise en œuvre était d'adapter des ontologies existantes, en empruntant ses termes. Les termes importés sont empruntés à des ontologies bien connus tels que l'ontologie ***Friend-Of-A-Friend (FOAF), DAMLTime*** et ***Time Ontology***.

SOUPA Extension fournit des modèles d'ontologie qui étendent les ontologies de base [8].

Tous ces ontologies sont exprimées avec le langage standard OWL (Ontology Web language) et comprennent des composants de vocabulaires modulaires pour représenter les agents intelligents avec les croyances associées, désirs, intentions, temps, l'espace, les événements, les profils utilisateur, les actions, les politiques de sécurité et de protection privée.

3.8 Conclusion

Dans ce chapitre nous avons exposé la notion d'ontologie, son cycle de vie, son utilisation dans la modélisation de contexte de l'utilisateur dans les systèmes sensibles au contexte ainsi que son efficacité dans la modélisation par rapport aux autres approches de modélisation.

On a aussi présenté quelles que Ontologies pour la modélisation de contexte comme **SOCAM** ,**CoOL** et **SOUPA.** Bien que ces modèles saisissent les concepts généraux de contexte et fournissent un raisonnement sur le contexte, ils ne sont pas spécifiques pour le domaine de l'assistance des personnes handicapées à domicile et les maisonss intelligentes *SmartHose* ce qui nous nécessite l'adaptation de ces dernières ou le développement de nouvelles ontologies pour ce domaine d'application .

De plus, nous avons montré que les langages du Web sémantique sont bien adaptés aux besoins de nombreux environnements informatiques ubiquitaire pour les raisons suivantes:

- L'ontologie exprimée dans les langages du Web sémantique fournit des moyens pour développer les systèmes sensibles au contexte, le partage des connaissances du contexte et minimiser le coût de développement grâce à la réutilisation.
- RDF et OWL sont des langages de représentation des connaissances avec une riche expressivité. Ils sont suffisants pour modéliser les différents types d'informations contextuelles comme les informations associées aux personnes, événements, dispositifs, lieux, temps et espace.
- Comme l'ontologie de contexte a des représentations explicites de la sémantique, ils peuvent être exploités par les moteurs d'inférence logique, avec la capacité de raisonnement sur le contexte. Le système permet de détecter et de résoudre les connaissances incompatibles sur le contexte qui viennent souvent de la mauvaise détection de l'information contextuelle.

Enfin en utilisant les langages du Web Sémantique, un système sensible au contexte peut utiliser des règles logiques pour déduire les informations de contexte implicites et de haut niveau à partir des informations explicites et de bas niveau.

Chapitre 4

Développement de l'ontologie de modélisation de contexte dans un environnement domotique pour l'assistance des personnes handicapées

4.1 Introduction

Comme déjà décrit dans le chapitre précédent, la modélisation du contexte par utilisation d'ontologies est la meilleure façon de modélisation grâce à ses avantages multiples tel que : *l'expressivité, le partage des connaissances, l'inférence logique, la réutilisation des connaissances et l'extensibilité.*

Plusieurs ontologies génériques ont été développés pour capturer et représenter les concepts généraux de contexte comme les ontologies du projet **SOCAM** l'ontologie **CoOL** et **SOUPA** [42]. La principale limite de ces ontologies est qu'elles ne sont pas spécifiques pour le domaine de l'assistance des personnes handicapées à domicile et les maisons intelligentes *SmartHome.*

*L'objectif de notre étude est d'*assurer un contrôle de l'environnement de la maison ainsi que suggérer des actions à l'utilisateur l'aidant et lui facilitant ses tâches quotidiennes.

La réalisation de cet objectif nous mène aux développements :

- d'une ontologie spécifique pour ce domaine en se basant sur les ontologies génériques existantes ainsi que sur l'adaptation de ces dernières pour nos besoins.

- d'un mécanisme de raisonnement sur le contexte. .

4.2 Présentation de l'environnement Domotique et le Scenario pour l'assistance de la personne handicapée

4.2.1 Scenario de l'assistance :

Pour assister les personnes handicapées dans leur maison, dans notre système nous avons proposé la gestion des situations suivantes :

- Le control de l'environnement de la maison (fenêtre, portes, ….etc.),
- Réalisation d'un système d'alarme pour le contrôle de la maison :
  - alarme en cas de fumé,
  - alarme en cas de fuite de gaz,
  - alarme en cas d'accident ou de chute de l'utilisateur,
- Rappel des heures de prise de médicaments de l'utilisateur selon son état de santé,

- L'envoie des différents rappels aux différents personnes et cela selon leurs états dans l'environnement domotique,
- Allumer les lampes des chambres où l'utilisateur se trouve et assurer une gestion (activation/désactivation) intelligente de la lumière (par exemple dans le cas d'un éclairage naturel, par le soleil, dans une chambre spécifique, notre système doit être capable d'éteindre la lumière dans cette chambre),
- régler le climatiseur et la température de la maison selon les besoins de l'utilisateur,
- Détecter les accidents et les chutes de la personne handicapée puis, selon la situation actuelle de son assistant, le système informe ce dernier par
- un e-mail s'il est connecté à internet.
- un message SMS s'il est déconnecté.

4.2.2 Le système domotique proposé

Pour le développement de notre ontologie nous avons suivi les quatre phases du cycle de développement suivant (voir chapitre 2).

Phase 1 : évaluation des besoins

Dans cette étape on doit préciser :

- L'objectif opérationnel de l'ontologie :

  C'est l'utilisation de cette ontologie dans un système sensible au contexte pour l'assistance des personnes handicapées dans un environnement domotique.
- Le domaine de connaissances à modéliser :

  C'est la modélisation des connaissances de contexte de l'utilisateur dans un environnement domotique.
- Les utilisateurs potentiels : Qui sont :
  - la personne handicapée, son assistant ainsi que son médecin traitant.

Après une étude approfondie des nombreux travaux s'intéressant au développement d'un environnement domotique, l'environnement qu'on propose dans notre étude, pour l'assistance des personnes handicapées, se compose d'une chambre contenant tous les dispositifs et les capteurs nécessaires pour assurer l'objectif du système. Cette ***chambre*** regroupe une ***lampe*** (pour fournir un éclairage suffisant sur l'ensemble de la pièce), un ***radiateur***, un ***climatiseur*** et un ***téléviseur*** avec des chaines favorites.

a- L'ensemble des capteurs installés dans cette chambre sont les suivants :

-un capteur de luminosité.

-un capteur pour détecter la présence ou non de l'utilisateur dans la pièce en utilisant un tag RFID (Radio Frequence Identification).

-un capteur de température,

-un capteur de détection de feu.

-un capteur pour la détection de fuite de gaz.

-un capteur pour la détection des accidents.

b- De plus, la chambre est dotée d'un système d'alarme qui contrôle l'environnement et les actions de l'utilisateur, puis envoi ces alarmes appropriées à son assistant :

- alarme en cas de fumé,
- alarme en cas d'une fuite de gaz,
- alarme en cas d'accidentant ou de chute de l'utilisateur.

c- Les personnes impliquées dans le système sont les suivantes :

- L'handicapé: Toute Personne qui souffre d'un handicap, d'une déficience physique ou mentale [57].
- L'assistant : toute personne chargée du suivi et du contrôle de la santé de personne handicapée
- Le Médecin chargée du suivi médical de la personne handicapée.

d- Le système dispose aussi d'une entité nommée ***agenda*** contenant toutes les informations spécifiques à la gestion des rendez-vous et des rappels de prise de médicament.

Les fonctionnalités qui doivent être assurées par notre système sont les suivantes :

– La température de la pièce doit être maintenue dans un intervalle de valeurs donné. Si la température de la pièce est trop élevée le système doit intervenir pour régler le climatiseur dans une température approprie.

– Lorsqu'un utilisateur s'approche à la chambre, la lampe doit s'allumer. Pour cela, les capteurs de présence situés dans la chambre identifient l'existence ou non de la personne.

Remarque : Évidemment un humain peut également intervenir manuellement sur chacun des éléments indépendamment du système.

Phase 2 : Conceptualisation par la technique UML

La deuxième étape du cycle de développement de l'ontologie concerne la ***conceptualisation*** des concepts du domaine dans un modèle conceptuel. Dans cette étape les connaissances du domaine domotique sont décrites par des concepts et des relations.

Dans notre système on a utilisé le diagramme de classe UML pour représenter les différents concepts et relations entre concepts.

Après analyse de l'environnement domotique proposé dans le scénario, les concepts que nous avons identifiés pour le développement de notre ontologie sont les suivants:

- Lampes(Light)
- Climatiseur (Air-conditioner)
- Chambre (Room)
- téléviseur (TV)
- Personne (Actor)
  - Personne handicapée (DisabledPersone)
  - Médecin (Doctor)
  - Assistant(Assistant)
- Agenda (Agenda)
- Médicament (Medicament)
- Rappelle de prise de médicament (MedicalAlert)
- Smart Phone (SmartPhone)
- PC (PC)
- Alarme (Alert)
  - Alarme de fumé (FireAlert)
  - Alarme de fuite de gaz (GazAlert)
  - Alarme en cas d'accidentant (AccidentAlert)
- Les capteurs (sensor)
  - Capteur de luminosité (LightSensor).
  - Capteur d'existence de personne (RFIDSensor).
  - Capteur de température (TemperatureSensor).

- Capteur de détection de feu (FireSensor).
- Capteur de détection de fuite de gaz (GazSensor).
- Capteur de détection d'accident (AccidentSensor).

La représentation UML, dans le diagramme de classe, de ces concepts sont modélisés par des classes ou l'identité de chaque classe c'est le nom du concept qu'elle représente et ses attributs c'est les informations de même ce concept.

- ***Modélisation des informations de contexte***

Apres l étude des différentes ontologies existantes de haut niveaux(*upper- ontology*) qui modélisant les informations du contexte ; comme **SOCAM, CoOL** et **SOUPA ;** nous constatons que les entités **Emplacement**(Location), **Utilisateur**(user), A**ctivité** (Activity) et les **différentes ressources de calcul** représentent les entités contextuelles les plus fondamentales pour capturer l'information du contexte dans un environnement intelligent.

L'ensemble de ces entités héritent tous de l'entité ***ContextEntity*** qui est l'entité de contexte de base.

La représentation UML (dans le diagramme de classe) de ces concepts est décrite par les classes suivantes :

ContextEntity : cette classe décrit l'état actuel de l'utilisateur dans l'environnement qui comprend l'activité décrivant la tâche en cours, la localisation caractérisant l'emplacement de l'utilisateur et l'heure décrivant : années, mois, semaines, jours, heures et minutes.

Actor : cette classe a pour objectif de décrire toute personne impliquée dans l'environnement domotique tel que la personne handicapée, son assistant et son médecin.

Service : cette classe décrit une entité de service sollicitée par une personne.

Devise : cette classe vise à décrire tout dispositif ou une pièce d'équipement utilisée pour maintenir ou améliorer les capacités fonctionnelles d'une personne ayant un handicap, y compris les aides cognitives, les prothèses auditives, les aides visuelles, auditives et moteur ressources.

Sensor : cette classe vise à décrire tous les capteurs utilisés dans l'environnement et cela pour capturer les différentes informations de contexte y compris :***LightSensor, RFIDSensor , TemperatureSensor , FireSensor , GazSensor, AccidentSensor***

PhysicalObject : cette classe décrit les objets physiques intelligents et adaptables qui soutiennent l'assistance des services et des dispositifs appropriés au profil de l'utilisateur.

Agenda : cette classe décrit l'agenda utilisé par la personne handicapée et le temps de prise des médicaments

Medicament : cette classe décrit les médicaments que la personne handicapée doit les prendre.

Alert : cette classe décrit les différentes alarmes déclenchées par le système. On distingue deux types d'alarme ***SoftAlert*** et ***MaterialAlert.*** SoftAlert se déclenche dans le téléphone portable de la personne ou dans son PC, tandis que MaterialAlert c'est une alarme matériel installée dans l'environnement.C'est différentes classes sont représentées par le diagramme de classe UML suivant :

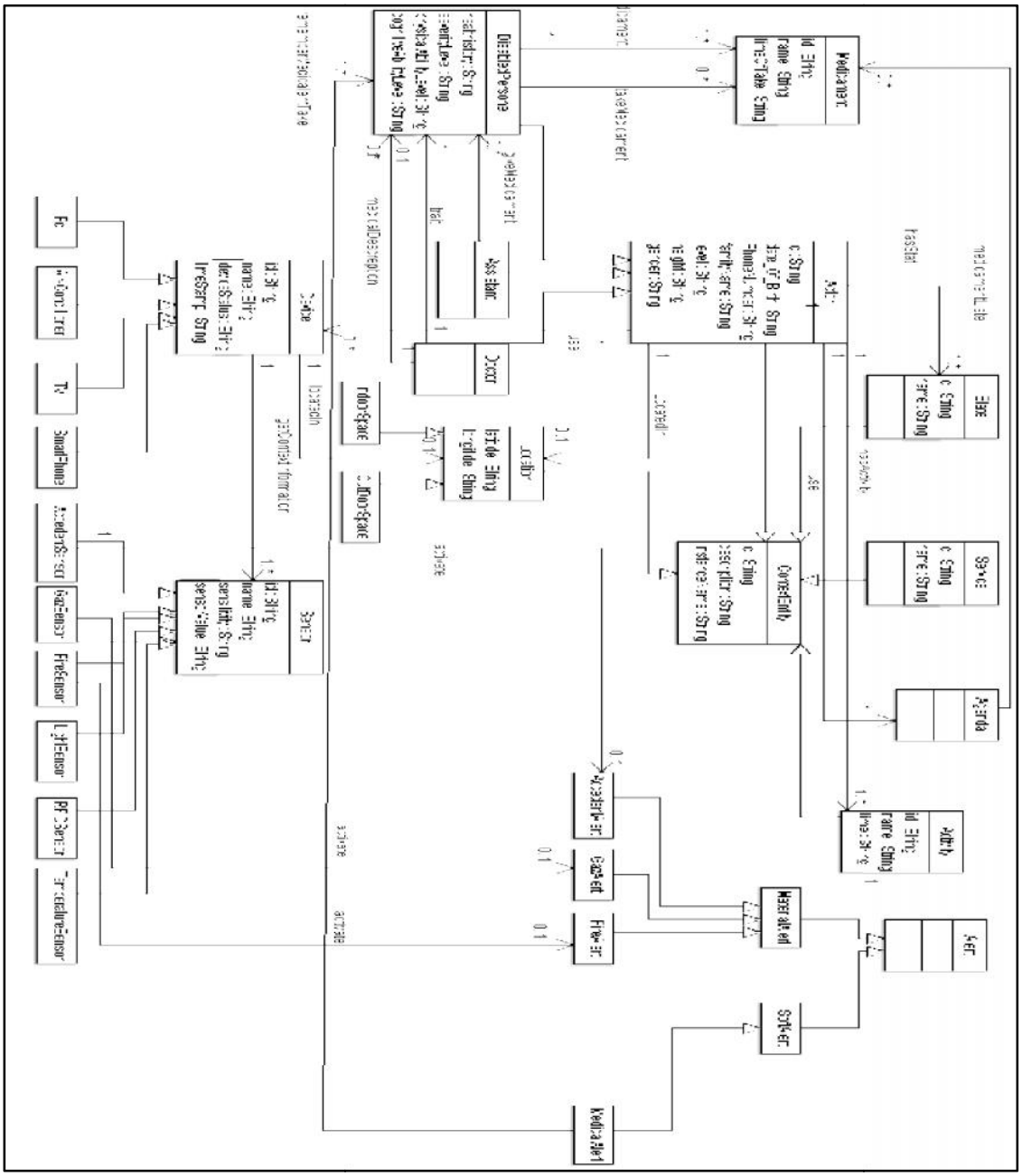


Figure 4.1 Diagramme de Classe de l'environnement domotique.

Les propriétes des differentes classes de notre ontologie sont les suivantes :

- ***LocateIn*** *:* cette relation affecte une entité physique ou une personne à une localisation dans l'environnement.
- ***hasActivity*** *: relation entre la classe Actor et la classe Activity*. *C*ette relation définit l'activité courante des acteurs.
- ***hasUsage*** *:* relation entre la classe *Activity* et la classe *Devise.* Elle définit les dispositifs utilisés dans une activité.
- ***ActivateAccidentAlert*** *: relation* entre la classe *AccidentSensor* et la classe *AccidentAlert. Elle* modélise l'action de déclanchement d'une alarme en cas d'un accident de la personne handicapée.
- ***ActivateMedicalAlert*** *:* relation entre la classe *Agenda* et la classe *MedicalAlert.* Elle modélise l'action de déclanchement d'un rappel de prise de médicament.
- ***alertAccident*** *:* relation entre la classe *AccidentAlert* et la classe *Assistant.* Elle modélise l'information de l'assistant de l'handicapé par une alarme d'accident.
- ***getContexteInformation :*** relation entre la classe *Devise* et la classe *Sensor.* Elle modélise la récupération des informations du contexte par les dispositifs installés dans l'environnement à partir des capteurs.
- ***giveMedicament :*** relation entre la classe *Assistant* et la classe *DiseabledPersone.* Elle modélise la donnée de médicament à la personne handicapée par son assistant.
- ***hasMedicament :*** relation entre la classe *DiseabledPersone* et la classe *Medicament.* Elle définit la liste des médicaments à prendre par la personne handicapée.
- ***medicalDescreption :*** realtion entre la classe *Doctor* et la classe *DiseabledPersone.* Elle modélise la descreption des médicaments à la personne handicapée par le médecin.
- ***remembreMedicalTake:*** relation entre la classe *MedicalAlert* et la classe *DiseabledPersone.* Elle mosélise le rappel de prise de médicament par la persone handicapée en cas d'oubli.
- ***takMedicament :*** relation entre la classe *DiseabledPersone* et la classe *Medicament.* Elle modélise la prise de médicament par la personne handicapée
- ***treat :*** relation entre la classe *Doctor* et la classe *DeseabledPersone.* Elle modélise le contrôle et le diagnostique de la personne handicapée par son médecin .

- La description des attribues des classes et des cardinalités des relations

Classe "ContextEntity"

Nom	Description
id : [1..1] string	L'Identité de la classe
description : [1..1] string	La description de la classe
instanceName : [1..1] string	Le nom de l'instance

Table 4.1 Attributs de la classe "ContextEntity"

Classe "Actor"

Hérite de: ContextEntity

Nom	Description
id : [1..1] string	L'ID de la classe
name : [1..1] string	Son prénom
date_Of_Birth : [1..1] string	Sa date de naissance
PhoneNumber : [1..1] string	Son Numéro de téléphone
familyName : [1..1] string	Son Nom
level : [1..1] string	Son niveau
height : [1..1] string	La mesure de sa taille
gender : [1..1] string	Son sexe (Masculin ou féminin)

Table 4.2 Attributs de la classe "Actor"

Relations de la classe "**Actor**"

Nom	Description
use-> : [0..*] Device	Un acteur peut utiliser un appareil
use-> : [1..1] Agenda	Un acteur peut utiliser l'agenda
LocatedIn-> : [0..1] Location	La localisation de l'acteur dans la chambre
hasActivity-> : [1..*] Activity	L'activité de l'acteur
hasStat-> : [1..*] State	L'état de l'acteur

Table 4.3 Associations de la classe "Actor"

Classe "sensor"

Hérite de: ContextEntity

Nom	Description
id : [1..1] string	L'ID du capteur
name : [1..1] string	Son nom
sensitivity : [1..1] string	Est-ce qu'il y a une information ou non
sensorValue : [1..1] string	La mesure de l'information

Table 4.4 Attributs de la classe "sensor"

Nom	Description
LocatedIn-> : [0..1] Location	La localisation du capteur
getContexteInformation-> : [1..*] Device	pour capturer les formations de contexte à partir des dispositifs

Table 4.5 Associations de la classe "sensor"

Classe "DisabledPersone"

Hérite de: Actor

Nom	Description
healthistory : [1..1] string	L'historique de son état
Severity Level : [1..1] string	Niveau de gravité de l'état
physicalAbilityLevel : [1..1] string	Niveau de capacité physique
cognitiveAbilityLevel : [1..1] string	Niveau de capacité cognitive

Table 4.6 Attributs de la classe "DisabledPersone"

Nom	Description
medicalDescreption-> : [1..*] Doctor	La description du médicament par le médecin
giveMedicament-> : [1..*] Assistant	La donné du médicament par l'assistant
hasMedicament-> : [1..*] Medicament	La liste de ces médicaments
treat-> : [*..1] Doctor	Le traitement et le suivi de la personne handicapée par le médecin
takeMedicament-> : [0..*] Medicament	La prise du médicament
rememberMedicalentTake-> : [undefined..undefined] MedicalAlert	Le rappel de la prise du médicament

Table 4.7 Associations de la classe "DisabledPersone"

Classe "Assistant "

Hérite de: Actor

Nom	Description
Les mêmes attributs de la classe Actor	

Table 4.8 Attributs de la classe "Assistant "

Nom	Description
alert-> : [0..1] AccidentAlert	Pour informer l'assistant qui il y'a un accèdent
giveMedicament-> : [0..1] DisabledPersone	La donné du médicament à la personne handicapée

Table 4.9 Associations de la classe "Assistant "

Classe "Doctor "

Hérite de: Actor

Nom	Description
Les mêmes attributs de la classe Actor	

Table 4. 10 Attributs de la classe "Doctor "

Nom	Description
medicalDescription-> : [1..*] DisabledPersone	La description du médicament pour la personne handicapée
treat-> : [1..1] DisabledPersone	Le traitement et le suivi de la personne handicapée

Table 4.11 Associations de la classe "Doctor "

Classe "Device"

Hérite de: ContextEntity

Nom	Description
id : [1..1] string	L'ID du Dispositif
name : [1..1] string	Son Nom
deviceStatus : [1..1] string	Son statut ('On' ,'Off')
timestamp : [1..1] string	Le temps associé au statut du dispositif

Nom	Description
use-> : [0..*] Actor	L'utilisation du dispositif par un acteur
LocatedIn-> : [0..1] Location	La localisation du dispositif
getContexteInformation-> : [1..*] sensor	La récupération des information de contexte à partir

	d'un capteur

Table 4.12 Associations de la classe "Device"

Classe "Air-conditioner "

Hérite de: Device

Nom	Description
Les mêmes attributs de la classe Devise	

Table 4.13 Attributs de la classe "Air-conditioner "

Classe "TV "

Hérite de: Device

Nom	Description
Les mêmes attributs de la classe Devise	

Table 4.14 Attributs de la classe "TV "

Classe "AccidentSensor "

Hérite de: Sensor

Nom	Description
Les mêmes attributs de la classe Sensor	

Table 4.15 Attributs de la classe "AccidentSensor "

Classe "GazSensor"

Hérite de: sensor

Nom	Description
Les mêmes attributs de la classe Sensor	

Table 4.16 Attributs de la classe "GazSensor"

Nom	Description
activate-> : [0..1] GazAlert	L'activation de l'alarme GazAlert

Table 4.17 Associations de la classe "GazSensor"

Classe "FireSensor"

Hérite de: Sensor

Nom	Description
Les mêmes attributs de la classe Sensor	

Table 4.18 Attributs de la classe "FireSenso

Nom	Description
activate-> : [0..1] FireAlert	L'activation de l'alarme 'FireAlert'

Table 4.19 Associations de la classe "FireSensor"

Classe "LightSensor"

Hérite de: sensor

Nom	Description
Les mêmes attributs de la classe Sensor	

Table 4. 20 Attributs de la classe "LightSensor".

Classe "RFIDSensor"

Hérite de: Sensor

Nom	Description
Les mêmes attributs de la classe Sensor	

Table 4.21 Attributs de la classe "RFIDSensor"

Classe "TemperatureSensor "

Hérite de: sensor

Nom	Description
Les mêmes attributs de la classe Sensor	

Table 4.22 Attributs de la classe "TemperatureSensor "

Classe "Alert"

Nom	Description

id : [1..1] string	L'ID de L'alarme
alertTime : [1..1] string	Le temps de l'activation de l'alarme
alertStatus : [1..1] string	Le statut de l'alarme ('On' ,'Off')

Table 4.23 Attributs de la classe "Alert"

Classe "SmartPhone"

Hérite de: Device

Nom	Description
Les mêmes attributs de la classe Device	

Table 4.24 Attributs de la classe "SmartPhone".

Classe "Service"

Hérite de: ContextEntity

Nom	Description
id : [1..1] string	L'ID de Service
name : [1..1] string	Son Nom

Table 4.25 Attributs de la classe "Service"

Classe "Activity"

Hérite de: ContextEntity

Nom	Description
id : [1..1] string	L'ID de l'activité
name : [1..1] string	Son Nom
time : [1..1] string	Le temps de déroulement de l'activité

Table 4. 26 Attributs de la classe "Activity"

Nom	Description
hasActivity-> : [0..*] Actor	L'activité de l'acteur

Table 4.28 Associations de la classe "Activity"

Classe "Agenda"

Nom	Description
use-> : [1..1] Actor	L'utilisation de l'agenda par l'acteur
activate-> : [0..1] MedicalAlert	Activation de l'alarme MedicalAlert

Table 4.29 Associations de la classe "Agenda"

Classe "MedicalAlert"

Hérite de: SoftAlert

Nom	Description
Les mêmes attributs de la classe SoftAlert	

Table 4.30 Attributs de la classe "MedicalAlert"

Nom	Description
activate-> : [0..1] Agenda	L'activation de l'alarme à partir de l'agenda
rememberMedicalentTake-> : [1..*] DisabledPersone	Rappelle de prise de médicament

Table 4.31 Associations de la classe "MedicalAlert"

Classe "AccidentAlert"

Hérite de: MaterialAlert

Nom	Description
Les mêmes attributs de la classe SoftAlert	

Table 4.22 Attributs de la classe "AccidentAlert"

Nom	Description
Activate-> : [0..1] AccidentSensor	Activation de l'alarme
alert-> : [0..1] Assistant	Informer l'assistant qu'il y a un accident

Table 4.33 Associations de la classe "AccidentAlert"

Classe "Location"

Hérite de: ContextEntity

Nom	Description
latitide : [1..1] string	La latitude de la localisation
longitide : [1..1] string	La longitude de la localisation

Table 4.34 Attributs de la classe "Location"

Classe "IndoorSpace"

Hérite de: Location

Nom	Description
Les mêmes attributs de la classe Device	

Table 4.35 Attributs de la classe "IndoorSpace"

Classe "OutDoorSpace"

Hérite de: Location

Nom	Description
Les mêmes attributs de la classe Device	

Table 4.36 Attributs de la classe "OutDoorSpace"

Classe "State"

Hérite de: ContextEntity

Nom	Description
id : [1..1] string	L'ID de l'état de la Personne handicapée
name : [1..1] string	Le nom de l'état

Table 4.37 Attributs de la classe "State"

Nom	Descraiption
hasStat-> : [0..*] Actor	L'état de l'Actor

Table 4. 38 Associations de la classe "State"

Classe "Medicament"

Nom	Description
id : [1..1] string	L'ID du Médicament
name : [1..1] string	Son Nom
timeOfTake : [1..1] string	Le temps de prise de Médicament

Table 4.39 Attributs de la classe "Medicament"

Nom	Description
hasMedicament-> : [0..*] DisabledPersone	La liste des médicaments de la personne handicapée
takeMedicament-> : [0..*] DisabledPersone	La prise du médicament par la personne handicapée

Table 4.40 Associations de la classe "Medicament".

Classe "MaterialAlert"

Hérite de: Alert

Nom	Description
Les mêmes attributs de la classe Alert	

Table 4.41 Attributs de la classe "MaterialAlert"

Classe "SoftAlert"

Hérite de: Alert

Nom	Description
Les mêmes attributs de la classe Alert	

Table 4. 45 Attributs de la classe "SoftAlert"

Classe "GazAlert"

Hérite de: MaterialAlert

Nom	Description
Les mêmes attributs de la classe MaterialAlert	

Table 4.46 Attributs de la classe "GazAlert"

Nom	Description
activate-> : [0..1] GazSensor	L'activation de l'alarme par le capteur GazSensor

Table 4.47 Associations de la classe "GazAlert"

Classe "FireAlert"

Hérite de: MaterialAlert

Nom	Description
Les mêmes attributs de la classe MaterialAlert	

Table 4.48 Attributs de la classe "FireAlert"

Nom	Description
activate-> : [0..1] FireSensor	L'activation de l'alarme par le capteur FireSensor

Table 4.49 Associations de la classe "FireAlert"

Phase 3 : formalisation de l'ontologie développée par le langage OWL

Cette étape représente la troisième étape dans le cycle de vie de développement de l'ontologie. Il s'agit de l'expression explicite et formelle de la conceptualisation obtenue dans l'étape précédente par un langage formel qui offre : un ensemble des composants sémantiques, des règles structurelles et d'une notation formelle destinée à organiser les relations entre les éléments constituant l'ontologie, permettre de réduire les ambiguïtés du langage naturel en offrant une plus grande expressivité et rendre l'ontologie compréhensible par les machines.

Dans notre étude on a utilisé le langage OWL qui représente le standard proposé par le W3C pour la formalisation de notre ontologie. Le langage OWL utilise les primitives de bases modélisées par les schémas RDF (les classes et les propriétés) comme il offre une structure de représentation dérivées des logiques de description qui augmentent l'expressivité.

Le langage OWL permet de décrire les concepts d'une façon plus expressive que le diagramme de classe UML il permet aussi d'ajouter des restrictions formelles et des caractéristiques aux classes et aux propriétés qui modélise le contexte.

Dans la formalisation de notre ontologie par OWL on a utilisé l'éditeur "Protege-2000» [58]. Ce dernier permet de construire une ontologie pour un domaine donné, de définir les formulaires de saisie de données ainsi que les données à atteindre à l'aide de ces formes sous la forme d'instances de cette ontologie.

a) Identification des classes de l'ontologie :

Une classe définit un groupe d'individus qui sont réunis parce qu'ils ont des caractéristiques similaires. L'ensemble des individus d'une classe est désigné par le terme « extension de classe », chacun de ces individus étant alors une « instance » de cette classe.

A partir du modèle de classe UML on peut définir les nouvelles classes de notre ontologie. Ces classes sont représentées par la figure suivante :

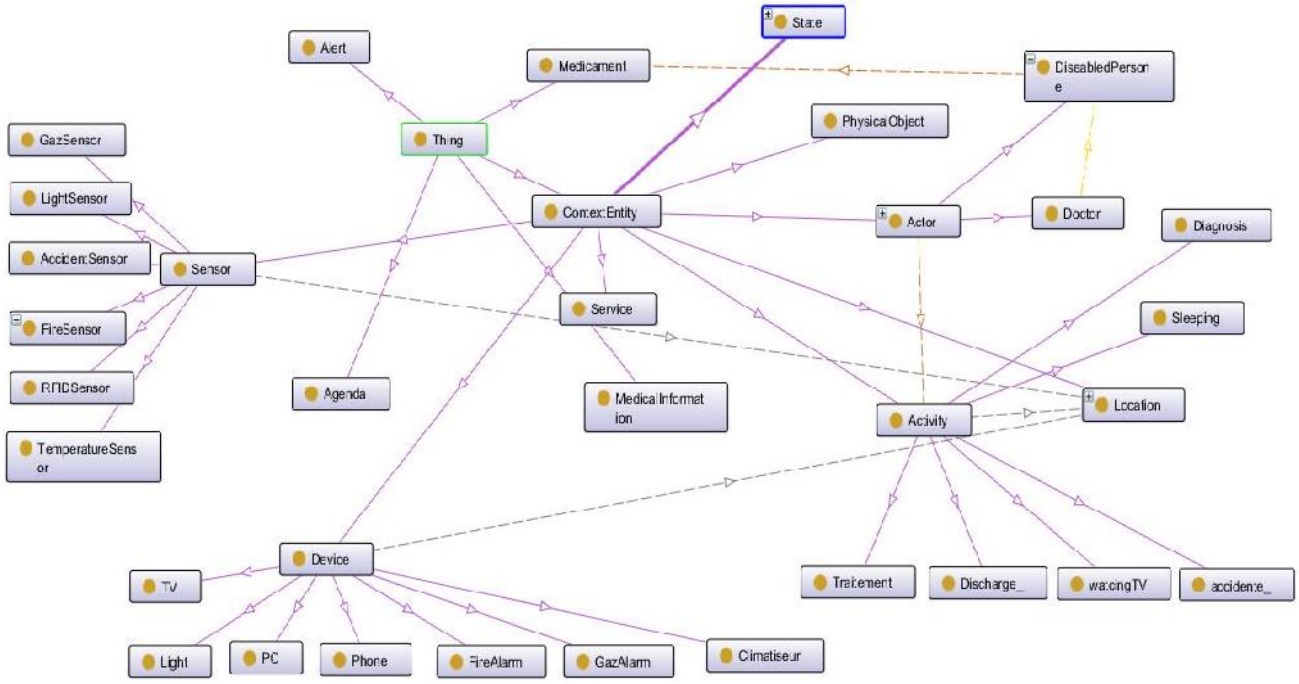


Figure 4.2 formalisation de l'ontologie avec l'outil protégé2000

Toutes ces classes sont héritées de la superclasse ***Thing (***donc toutes les autres classes sont des sous-classes).

b) Déclaration des classes de l'ontologie

La déclaration d'une classe se fait par le biais du mécanisme de « description de classe », qui se présente sous diverses formes.

Par exemple les classes ***Medicament*** et ***DiseabledPersone*** sont déclarée par ce qui suit :

```
106     <Declaration>
107         <Class IRI="#Medicament"/>
108     </Declaration>
```

```
70      <Declaration>
71          <Class IRI="#DiseabledPersone"/>
72      </Declaration>
```

Pour définir la relation d'héritage on utilise la propriété subClassOf. Par exemple la relation d'héritage entre la classe ***Activity*** et la classe ***ContextEntity*** est définie comme suit

```
246     <SubClassOf>
247         <Class IRI="#Activity"/>
248         <Class IRI="#ContextEntity"/>
249     </SubClassOf>
```

c) Définition des Propriétés de l'ontologie développée:

L'objectif des propriétés dans une ontologie est d'exprimer les faits au sujet de ces classes ainsi que de leurs instances. OWL fait la distinction entre deux types de propriétés :

• ***les propriétés d'objet*** permettent de relier des instances à d'autres instances

• ***les propriétés de type*** de donnée permettent de relier des individus à des valeurs de données.

Une propriété d'objet est une instance de la classe *owl*:*ObjectProperty*. Une propriété de type de donnée étant une instance de la classe *owl:DatatypeProperty*. Ces deux classes sont-elles même sous-classes de la classe RDF rdf:Property.

Les propriétés d'objet de notre ontologie sont dérivées directement à partir des relations de diagramme de classe UML, et enrichis par les caractéristiques nécessaires qui permettent de raisonner sur l'ontologie.

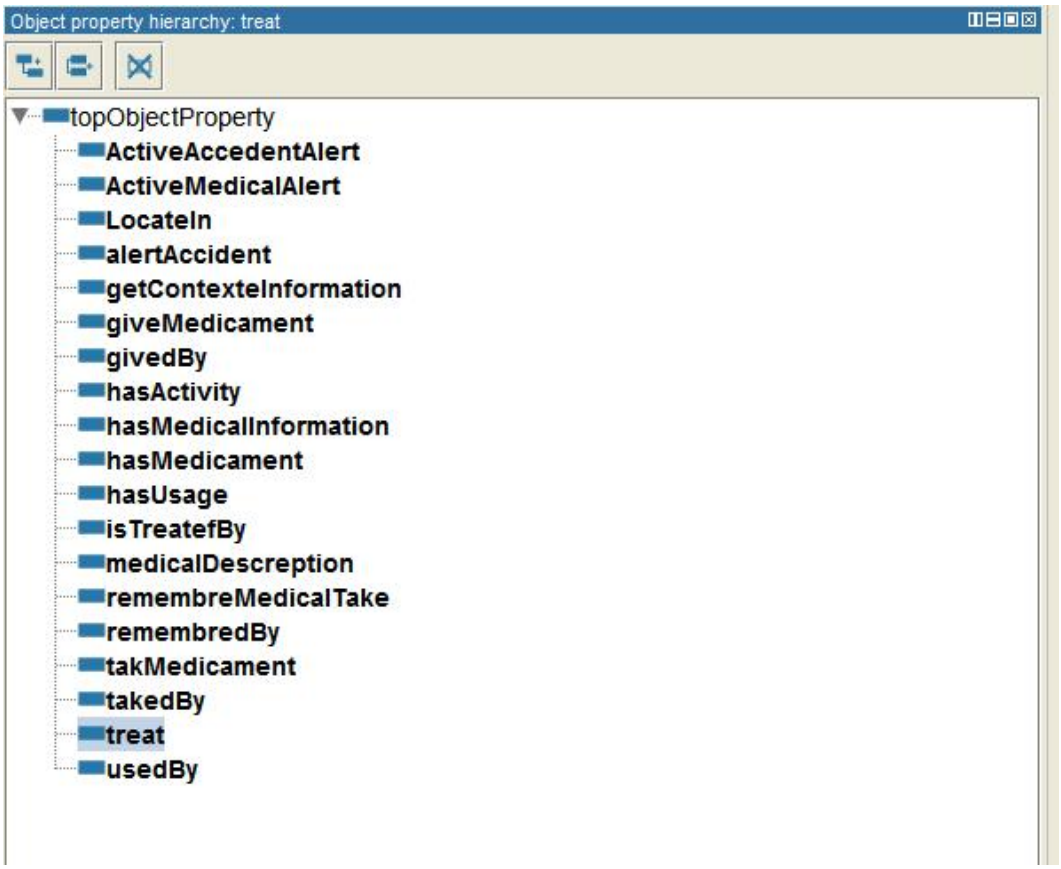


Figure 4.3 Propriétés d'objet de l'ontologie

- ***LocateIn***

```
172     <Declaration>
173         <ObjectProperty IRI="#LocateIn"/>
174     </Declaration>
```

- ***hasActivity***

```
187     <Declaration>
188         <ObjectProperty IRI="#hasActivity"/>
189     </Declaration>
```

- ***hasUsage***

```
196     <Declaration>
197         <ObjectProperty IRI="#hasUsage"/>
198     </Declaration>
```

- ***ActivateAccidentAlert***

```
160     <Declaration>
161         <ObjectProperty IRI="#ActivateAccedentAlert"/>
162     </Declaration>
```

- ***ActivateMedicalAlert***

```
169     <Declaration>
170         <ObjectProperty IRI="#ActivateMedicalAlert"/>
171     </Declaration>
```

- ***ActivateFireAlert***

```
163     <Declaration>
164         <ObjectProperty IRI="#ActivateFireAlert"/>
165     </Declaration>
```

- ***alertAccident***

```
175     <Declaration>
176         <ObjectProperty IRI="#alertAccident"/>
177     </Declaration>
```

- ***getContexteInformation***

```
178     <Declaration>
179         <ObjectProperty IRI="#getContexteInformation"/>
180     </Declaration>
```

- ***giveMedicament***

```
181     <Declaration>
182         <ObjectProperty IRI="#giveMedicament"/>
183     </Declaration>
```

- ***hasMedicament***

```
193     <Declaration>
194         <ObjectProperty IRI="#hasMedicament"/>
195     </Declaration>
```

- ***hasMidicalInformation***

```
190     <Declaration>
191         <ObjectProperty IRI="#hasMedicalInformation"/>
192     </Declaration>
```

- ***medicalDescreption***

```
202     <Declaration>
203         <ObjectProperty IRI="#medicalDescreption"/>
204     </Declaration>
```

- ***remembreMedicalTake***

```
205     <Declaration>
206         <ObjectProperty IRI="#remembreMedicalTake"/>
207     </Declaration>
```

- ***takMedicament***

```
211     <Declaration>
212         <ObjectProperty IRI="#takMedicament"/>
213     </Declaration>
```

- ***treat***

```
217     <Declaration>
218         <ObjectProperty IRI="#treat"/>
219     </Declaration>
```

d) Caractéristiques des Propriétés d'objet

Une propriété d'objet peut avoir plusieurs caractéristiques, ce qui permet d'affiner grandement la qualité des ***raisonnements*** liés à cette propriété.

Parmi les caractéristiques de propriétés principales, on trouve :

- la transitivité
- la symétrie
- l'antisymétrie
- La réflexivité
- Non réflexivité
- la fonctionnalité
- l'inverse

e) Attribution des fonctionnalités aux propriétés :

L'ajout d'une caractéristique à une propriété de l'ontologie se fait par l'emploi de la balise OWL spécifique à la définition de cette propriété

Exemple :

Avec la propriété ***LocateIn*** :

```
520     <TransitiveObjectProperty>
521         <ObjectProperty IRI="#LocateIn"/>
522     </TransitiveObjectProperty>
```

La même chose avec la propriété ***inverseOf***

```
<owl:ObjectProperty rdf:ID="usedBy">
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="hasUsage"/>
  </owl:inverseOf>
</owl:ObjectProperty>
```

Et les propriétés de type sont dérivées à partir des attributs des classes du diagramme UML et ont les mêmes descriptions que les attribues des classes UML

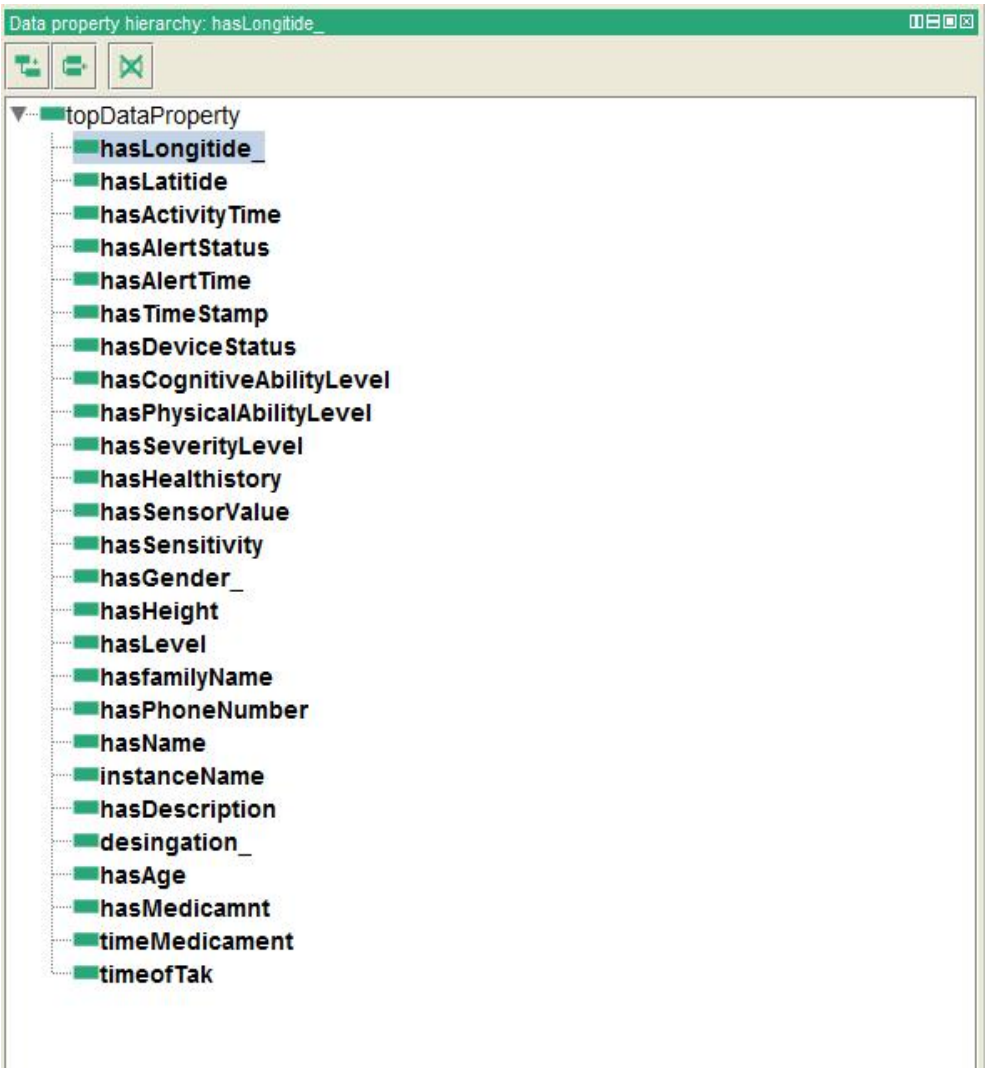


Figure 4.4 Propriétés de type de l'ontologie

4.4 Conclusion

Dans ce chapitre nous avons présenté le processus de développement de notre ontologie spécifique pour la modélisation de contexte dans un environnement domotique et cela en se basant sur les ontologies génériques existantes tel que **SOCAM** et **CoOL.**

Le processus de développement adopté se résume en quatre étapes qui sont : l'évaluation des besoins, la conceptualisation de l'ontologie par le diagramme de classe UML, la formalisation de cette conceptualisation par le langage OWL et enfin l'implémentation qui a été faite par l'outil Protégé2000.

Le prochain chapitre sera consacré au développement du mécanisme de raisonnement sur le contexte basé sur l'ontologie développée.

Chapitre 5

Développement de mécanisme de raisonnement sur le contexte

5.1 Introduction

Le raisonnement sur le contexte c'est la manipulation des connaissances acquises à partir des capteurs installés dans l'environnement. L'objectif d'un raisonnement sur le contexte est de produire des nouvelles connaissances permettant la gestion des différentes situations de l'assistance, telle que la notification sur l'état de la personne handicapée, ou pour lancer une Alarme en cas d'accident ou un incendie dans la maison.

Pour le développement de notre système nous avons adopté la technique du raisonnement sur le contexte a basé de règle. Pour l'implémentation de ces règles nous avons utilisé le langage SWRL (Semantic Web Rule Language) qui permet d'étendre l'expressivité du langage OWL utilisé pour la formalisation de notre ontologie.

5.2 Raisonnement à base de règle

L'élément de base des systèmes à base de règles est la règle de production dont la forme est :

SI <condition> ALORS <action>

La partie condition est exprimée par un prédicat logique correspondant à une affirmation sur la base des connaissances qui doit être vraie au moment de la validation de la règle et cela afin de déclencher l'action.

Dans notre système, pour réaliser un raisonnement à base de règle nous avons utilisé le langage SWRL [53].Dans ce langage la règle prend la forme d'une requête conjonctive, qui consiste en une implication entre un antécédent (représente la *condition*) et un conséquent (représente l'action). , Chaque fois que les conditions spécifiées dans l'antécédent sont vérifiées alors les conditions indiquées dans le conséquent doivent être appliquées.

L'antécédent et le conséquent sont composés d'atomes, dont chacun représente un prédicat-unaire ou prédicat-binaire dans la ***TBox*** de l'ensemble de la logique de description DL. Les atomes dans ces règles sont sous la forme ***C (x), P (x, y)*** où « C » est un nom d'une classe, « P » est un nom d'une propriété et « x » et « y » sont des variables ou des individus dans l'***ABox***.

5.2.1 Syntaxe des Atomes SWRL :

Un atome SWRL est défini comme suit:

Atom ← C (i) | D (v) | R (i, j) | U (i, v) | builtin (p,v1, ..., vn) | i = j | i ≠j

Tel que :

C = Classe,

D = type de donnée,

R= propriété d'objet,

U =propriété de donnée,

i, j = Les noms des variables d'objet ou les noms de individués

v1,. . . vn = Noms des variables des données ou des noms de type de données.

p = nom de Built-in[1].

Exemple de règle :

Règle 2.1: takesPlaceIn (?x, y?)∧ IsBusy (y?) ⇒ hasStarted (?x)

Un exemple d'une règle affirmant que «lorsque la session X qui se déroule en salle Y a déjà commencée cela implique que la chambre Y est occupée" Si, par exemple la ABox contient les faits :

-takesPlace(Middleware_Session, Room_A)

- takesPlaceIn (AmI_Session, Room_B).

- takesPlaceIn(Privacy_Session, Room_C).

- hasStarted (Middleware_Session).

Le résultat de la règle précédente serait l'ensemble {Room_A, Room_C}, indiquant que le prédicat IsBusy est valable pour les deux individus.

L'utilisation d'un langage de règle tel que SWRL est nécessaire pour plusieurs raisons:

- Les ensembles de règles existantes peuvent être réutilisés.
- Il Ajoute plus d'expressivité au langage OWL.
- Il est plus facile à lire et à écrire des règles par utilisation d'un langage de règle.

[1] Bibliothèques réutilisables pour les règles SRWL

5.2.1 Les types des moteurs d'inférence

Il existe plusieurs moteurs d'inférence permettant l'exploitation des règles SWRL, telque [72].

- **SWRLTab** : éditeur de règles SWRL dans Protégé-OWL (open source).
- **Pellet** : Un moteur d'inférence open source en Java
- **RacerPro** : Un moteur d'inférence commercial

Dans notre Système nous avons choisi l'utilisation du moteur d'inférence **SWRLTab** qui offre un environnement de développement dans l'outil Protégé pour travailler avec des règles SWRL. **SWRLTab** a plusieurs caractéristiques parmi lesquelles on trouve :

- Prend en charge l'édition et l'exécution des règles,
- Offre un mécanisme permettant aux utilisateurs de définir et intégrer des bibliothèques réutilisables built-in,
- Prend en charge l'interrogation des ontologies,

En plus Protégé offre l'éditeur SWRL qui permet l'édition et l'exécution des règles SWRL qui est accessible sous forme d'onglet.

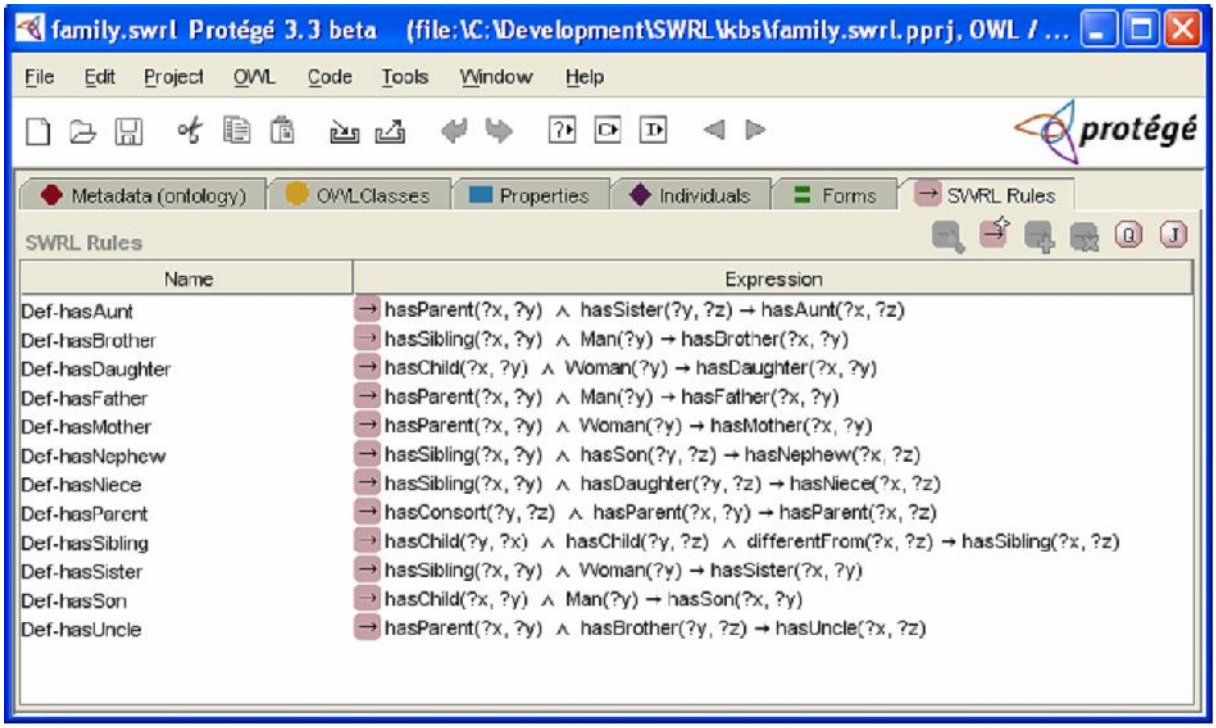


Figure 5.1 SWRLTab : éditeur de règles SWRL

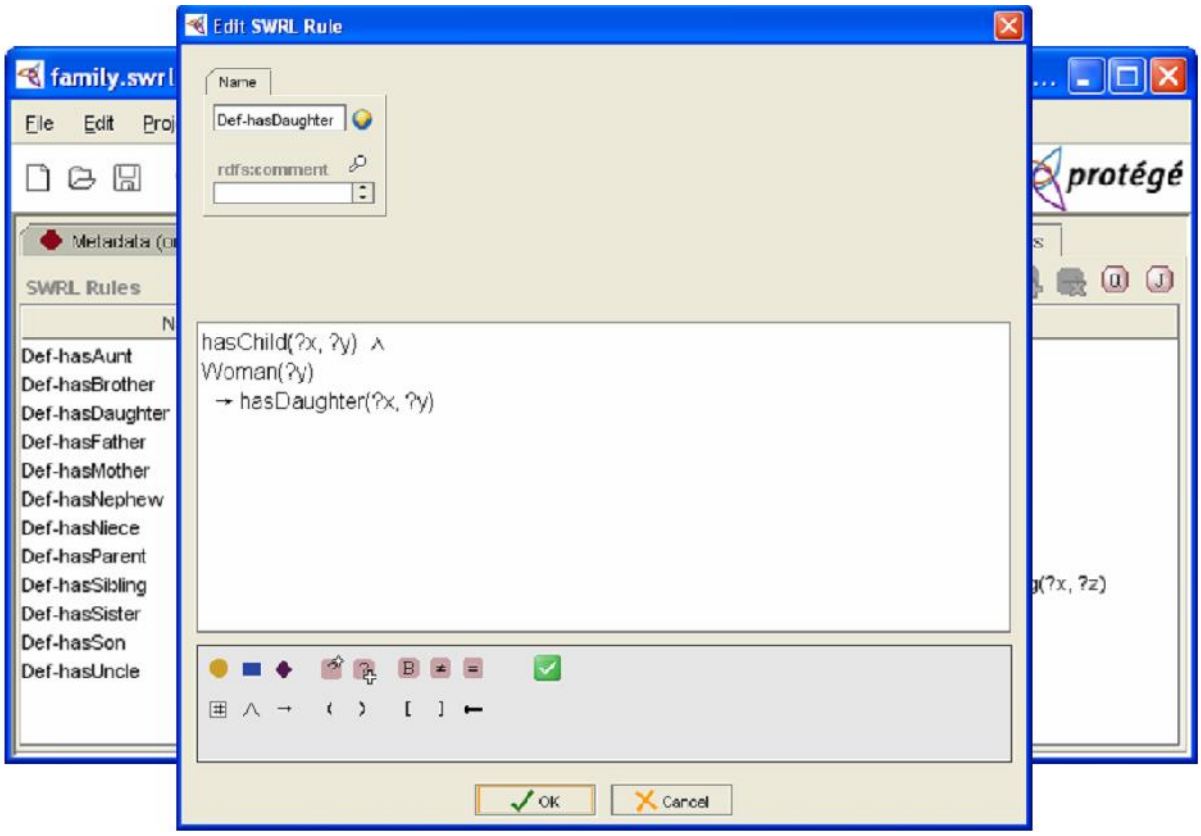


Figure 5.2 Exemple d'édition d'une règle SWRL avec l'éditeur **SWRLTab**

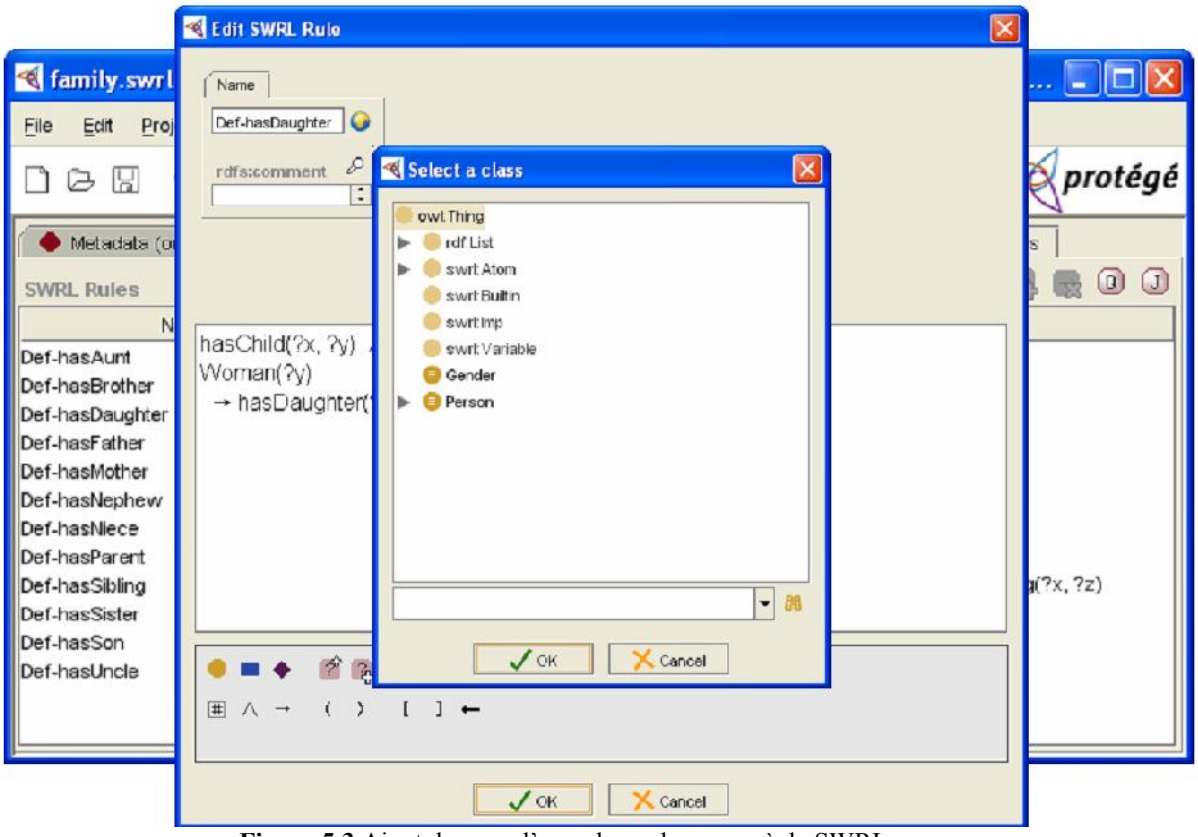


Figure 5.3 Ajout du nom d'une classe dans une règle SWRL

5.2.2 Exécution des règles SWRL

Pour l'exécution des règles SWRL on a besoin d'un module intermédiaire permettant la transformation de ces règles en format approprié pour un moteur d'inférence utilisé. Dans notre système nous avons utilisé le module ***SWRL Rule Engine Bridge.***

Ce module permet aussi d'extraire les règles SWRL, à partir d'une ontologie formalisée en langage OWL, comme il fournit une API permettant l'ajout des nouvelles connaissances déduites à notre l'ontologie.

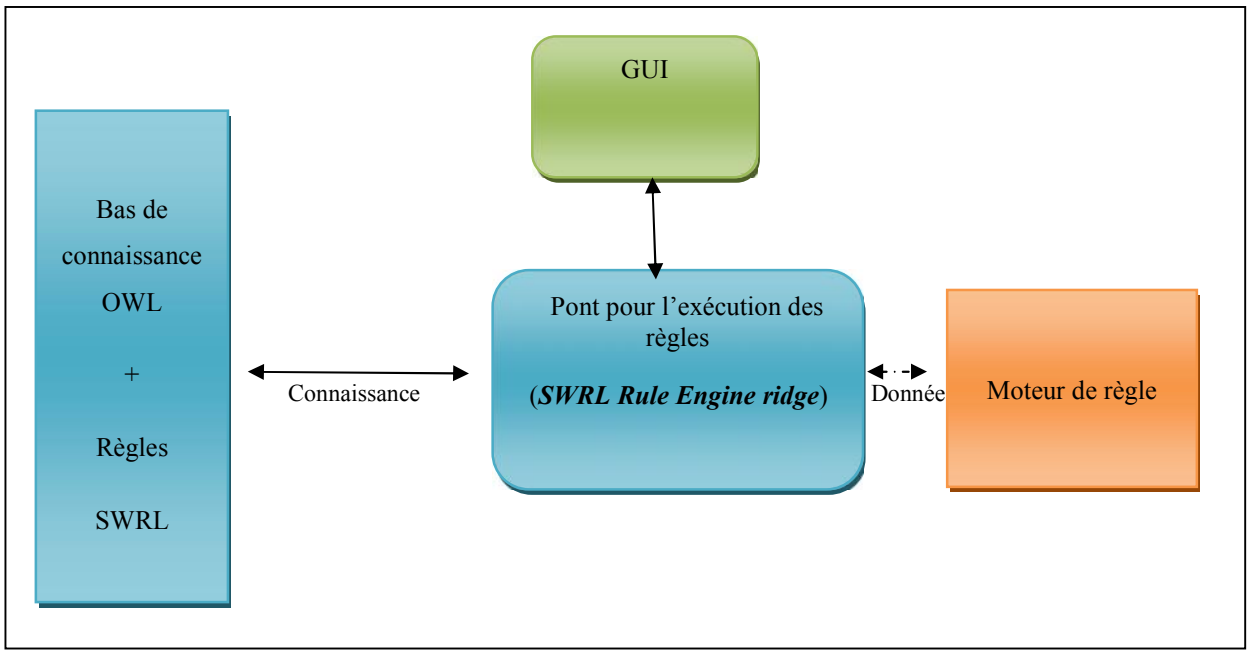


Figure 5.4 Exécution des règles SWRL

Nous avons aussi utilisé le module ***SWRLJessBridge*** pour intégrer le moteur de règles ***Jess*** avec Protégé. Jess est un moteur de règles basé sur Java. Il comprend une base des règles, base des faits, et un moteur d'exécution. Jess est disponible gratuitement pour les utilisateurs académiques et avec une petite taxe pour les utilisateurs non universitaires.

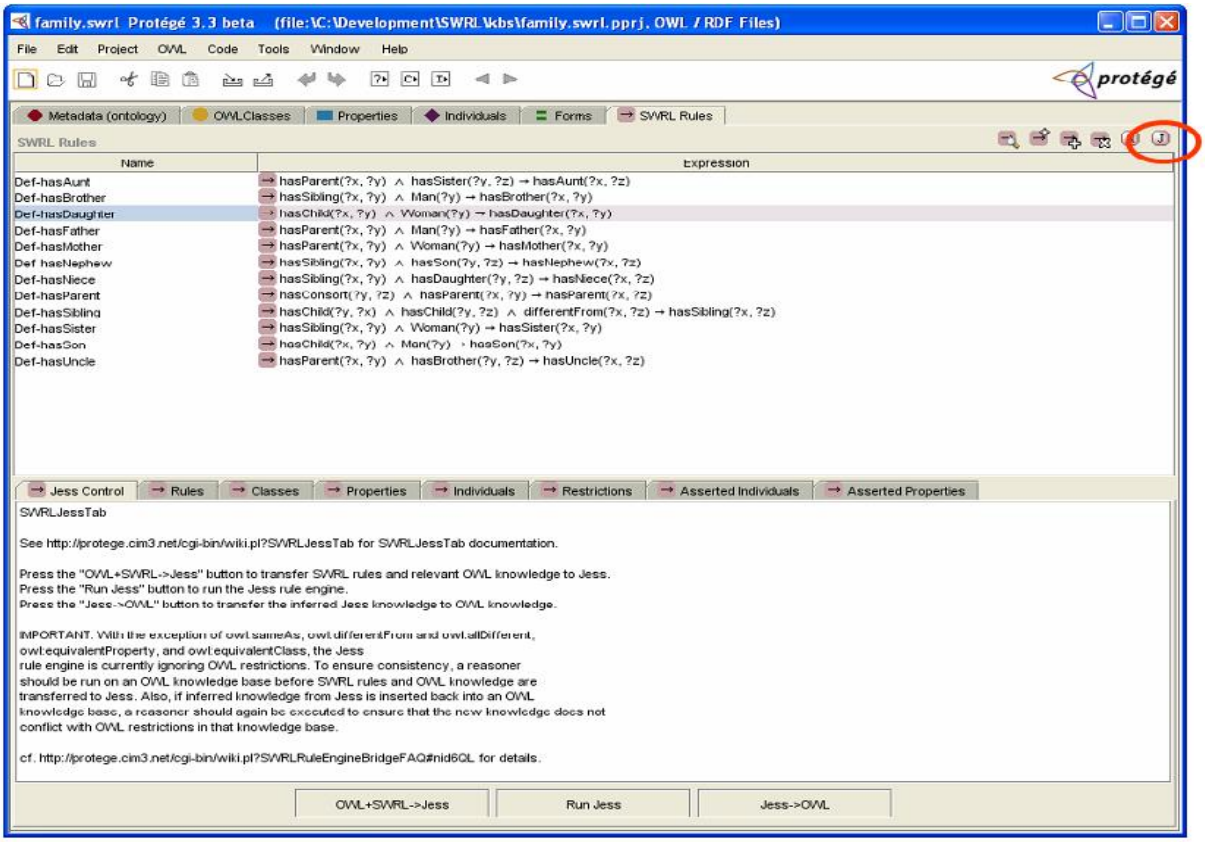


Figure 5.5 Le moteur de règles ***Jess*** intégré dans l'éditeur **SWRLTab**

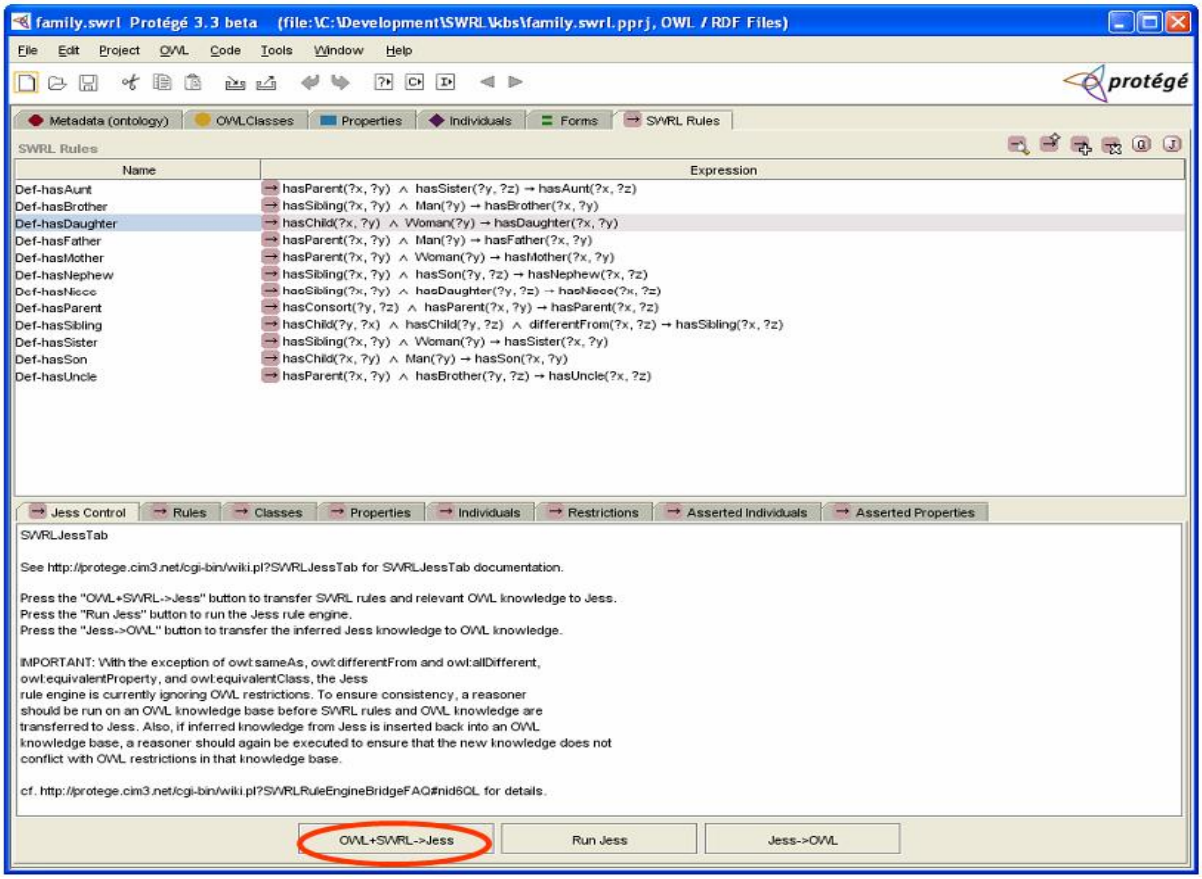


Figure 5.6 Transformation des connaissances OWL en format Jess

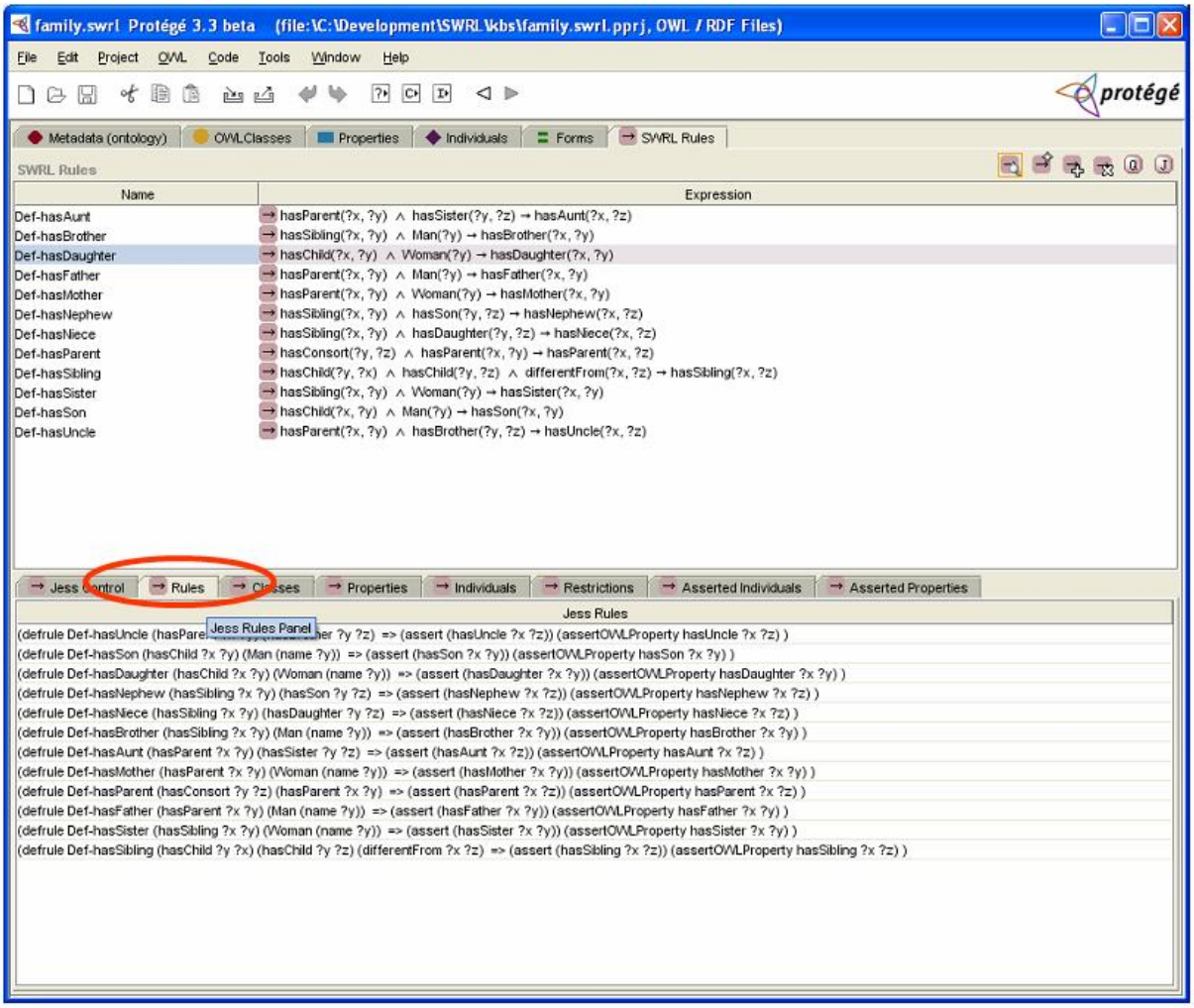


Figure 5.7 Exécution des règles avec Jess

5.3 Architecture du système

Le but de notre travail est :

- de proposer une nouvelle approche pour la modélisation de ***l'information de contexte*** afin de conseiller et d'assister une personne handicapée dans une maison intelligente (***Smart Home)***
- puis de raisonner sur cette information pour la gestion automatique de tout l'environnement intelligent.

Basant sur cet objectif le système que nous avons réalisé comporte deux parties de développement essentielles :

1. Développement de la base de connaissance : d'une ontologie modélisant l'information de contexte dans un environnement domotique et proposition des règles pour le raisonnement,
2. Développement d'un outil qui sera basé sur l'ontologie créée ainsi que sur un mécanisme de raisonnement basé sur le contexte de cette dernière.

L'architecture générale d notre système comporte les modules suivants **figure 5.8**

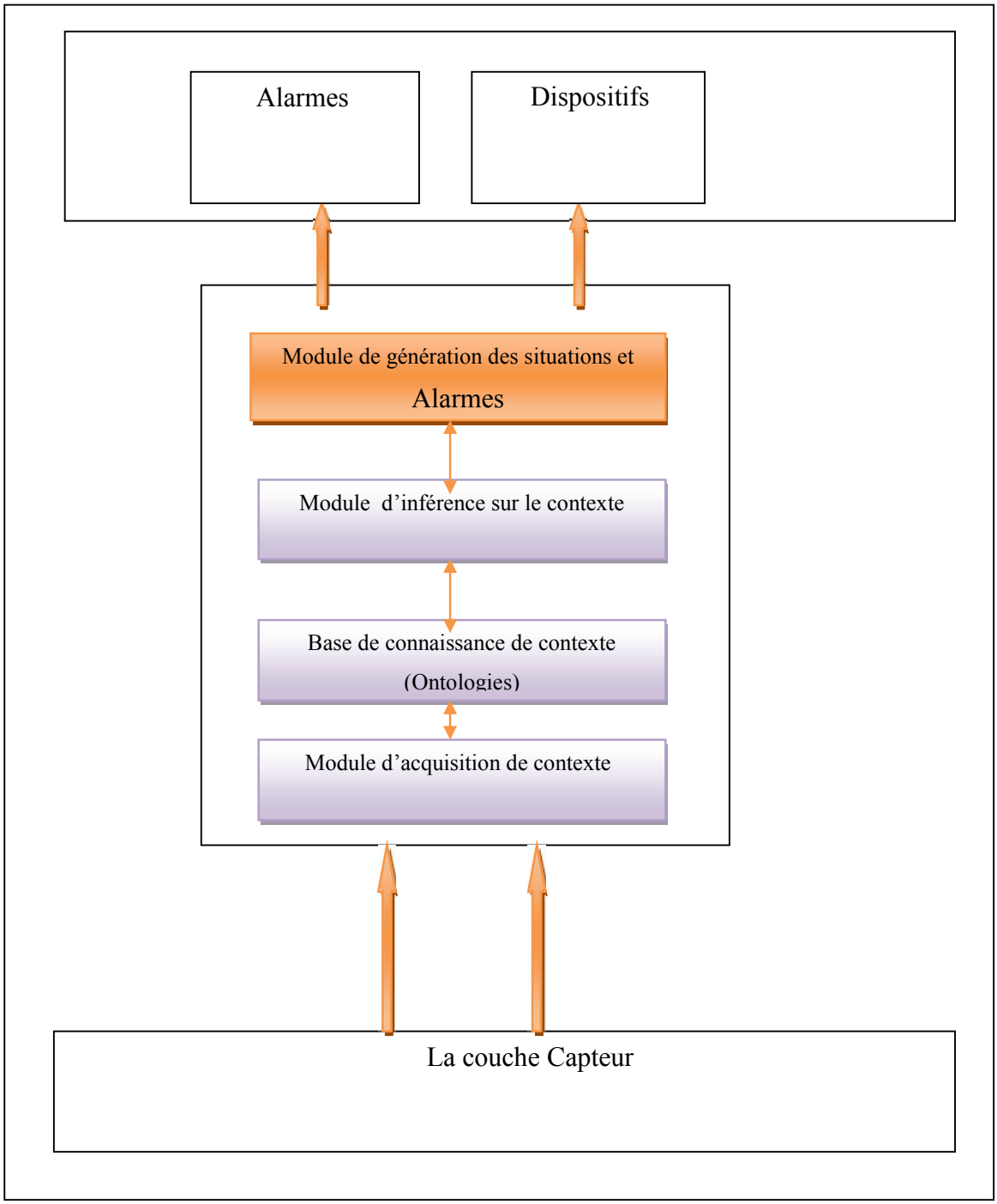


Figure 5.8 Architecture Système

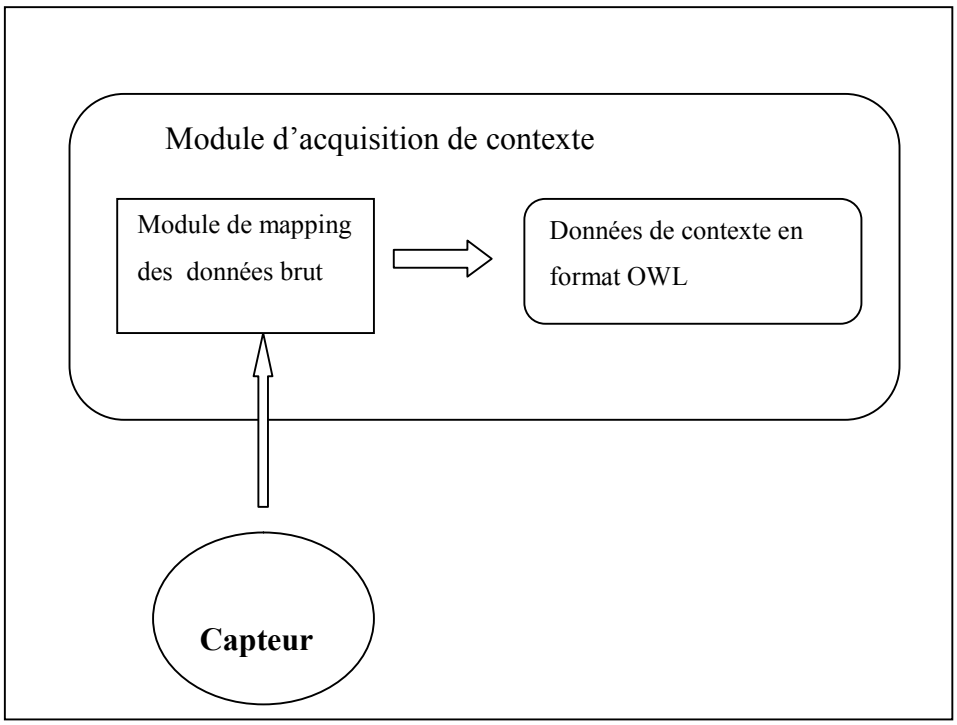


Figure 5.9 Module d'acquisition de contexte

1-La couche capteur

Cette couche contient l'ensemble des capteurs chargés de récupérer les données brutes de l'environnement domotique (exemple : les capteurs de lumière et de température).

2- Module d'acquisition de contexte

Ce module permet d'extraire des données brutes à partir des capteurs, de mapper ces données dans l'ontologie comme Instances de données contextuelles puis les formaliser en lagunage de formalisation (dans notre système nous avons utilisé le langage OWL).

3- Base de connaissance de contexte

Cette base de connaissance contient l'ontologie de l'environnement domotique intelligent décrite dans le langage de formalisation OWL (voire chapitre 3 pour plus de détaille). D'autre part elle contient des règles que nous avons proposées et utilisées dans le processus de raisonnement.

Ces règles de raisonnements utilisent les prédicats-unaire et les prédicats-binaire définis dans la ***TBox*** de notre ontologie:

Dans notre système nous avons proposées neufs règles :

- Pour générer l'alarme en cas d'une fuite de gaz ,
- Pour générer une alarme en cas d'un accident,
- Pour générer une alarme en cas de déclanchement d'un feu,
- Pour le contrôle de la lumière selon l'existence d'une personne,
- Pour le contrôle de la température de la pièce,
- Pour lancer une alerte pour la prise de médicament,
- Pour le don de médicament au personne handicapée,
- Pour lancer le traitement de la personne handicapé par le médcin,
- Pour lancer une alerte de rappelle pour la prise de médicament.

Dans notre système nous avons utilisé Protégé2000 pour le développement de l'ontologie et de ces règles SWRL. Protégé2000 fournit une interface utilisateur convivial pour faciliter le développement et la gestion de l'ontologie. En outre il fournit l'outil ***SWRLTab*** pour développer les règles d'inférence.

Figure 5.10 Ensemble des règles de raisonnement

4- Module d'inférence sur le contexte

Ce module permet l'inférence et le raisonnement à partir des différentes situations de l'assistance et selon les règles de raisonnement sur le contexte que nous avons proposées.

Tout d'abord, ***SWRLJessBridge*** transforme le modèle OWL en format JESS, en suite le moteur d'inférence JESS réalise l'inférence et envoie les résultats au module de génération des situations et alertes.

5-Module de génération des situations et Alarmes

Ce module génère les différentes situations et alarmes à partir des résultats obtenus dans la phase d'inférence.

5.4 Implémentation du système

L'implémentation de notre système se réalise en trois phases dans lesquelles nous avons utilisé plusieurs outils de développement.

1- Implémentation de l'ontologie

Cette partie a été présentée dans le chapitre -4-.

2- Implémentation du mécanisme raisonnement

- ***Ensemble des règles***

Spécification de la règle	***Règle écrite en langage SWRL***
Pour générer l'alarme en cas d'une fuite de gaz	*GazSensor(?s) ∧ GazAlert(?a) ∧ hasSensorValue(?s, "HIGH") → ActivateGazAlert(?s, ?a) ∧ hasAlertStatus(?a, "on")*
pour générer une alarme en cas d'un accident	*AccidentSensor(?s) ∧ AccedentAlert(?a) ∧ hasSensitivity(?s, true) → ActivateAccedentAlert(?s, ?a) ∧ hasAlertStatus(?a, "on")*
Pour générer une alarme en cas de déclanchement d'un feu	*FireSensor(?s) ∧ FireAlert(?a) ∧ hasSensorValue(?s, "HIGH") → ActivateFireAlert(?s, ?a) ∧ hasAlertStatus(?a, "on")*
Pour le contrôle de la lumière selon l'existence d'une personne	*DiseabledPersone(?p) ∧ RFIDSensor(?r) ∧ hasSensitivity(?r, true) ∧ Light(?L) ∧ hasDeviceStatus(?L, "Off") ∧ getContexteInformation(?L, ?r) → hasDeviceStatus(?L, "On")*
Pour le contrôle de la température de la pièce	*DiseabledPersone(?p) ∧ TemperatureSensor(?s) ∧ hasSensorValue(?s, "HIGH") ∧ Air-conditioner(?ac) ∧ hasDeviceStatus(?ac, "Off") ∧ getContexteInformation(?ac, ?s) → hasDeviceStatus(?ac, "on")*
Pour lancer une alerte pour la prise de médicament	*DiseabledPersone(?p) ∧ Medicament(?m) ∧ hasMedicament(?p, ?m) ∧ MedicalAlert(?a) ∧ hasAlertStatus(?a, "Off") ∧ timeofTak(?m, "now") → ActivateMedicalAlert(?a, ?p) ∧ hasAlertStatus(?a, "On")*
Pour le don de médicament au personne handicapée	*DiseabledPersone(?p) ∧ Assistant(?a) ∧ hasMedicament(?p, ?m) ∧ Medicament(?m) ∧ hasActivity(?p, Discharge__8) ∧ timeofTak(?m, "now") → giveMedicament(?a, ?m)*
Pour lancer le traitement de la personne handicapé par le médcin	*DiseabledPersone(?p) ∧ Doctor(?d) ∧ hasStat(?p, "Seek") → treat(?d, ?p)*

Pour lancer une alerte de rappelle pour la prise de médicament	*DiseabledPersone(?p) ∧ Medicament(?m) hasActivity(?p, Discharge__8) ∧ dontTakMedicament(?p, ?m) ∧ MedicalAlert(?a) remembreMedicalTake(?a, ?p) ∧ ActivateMedicalAlert(?a, ?p)*

3- Processus de raisonnement sur le contexte.

Le processus de raisonnement sur le contexte réalisé dans notre système suit les étapes illustrées dans la **figure 5.11** suivantes

.

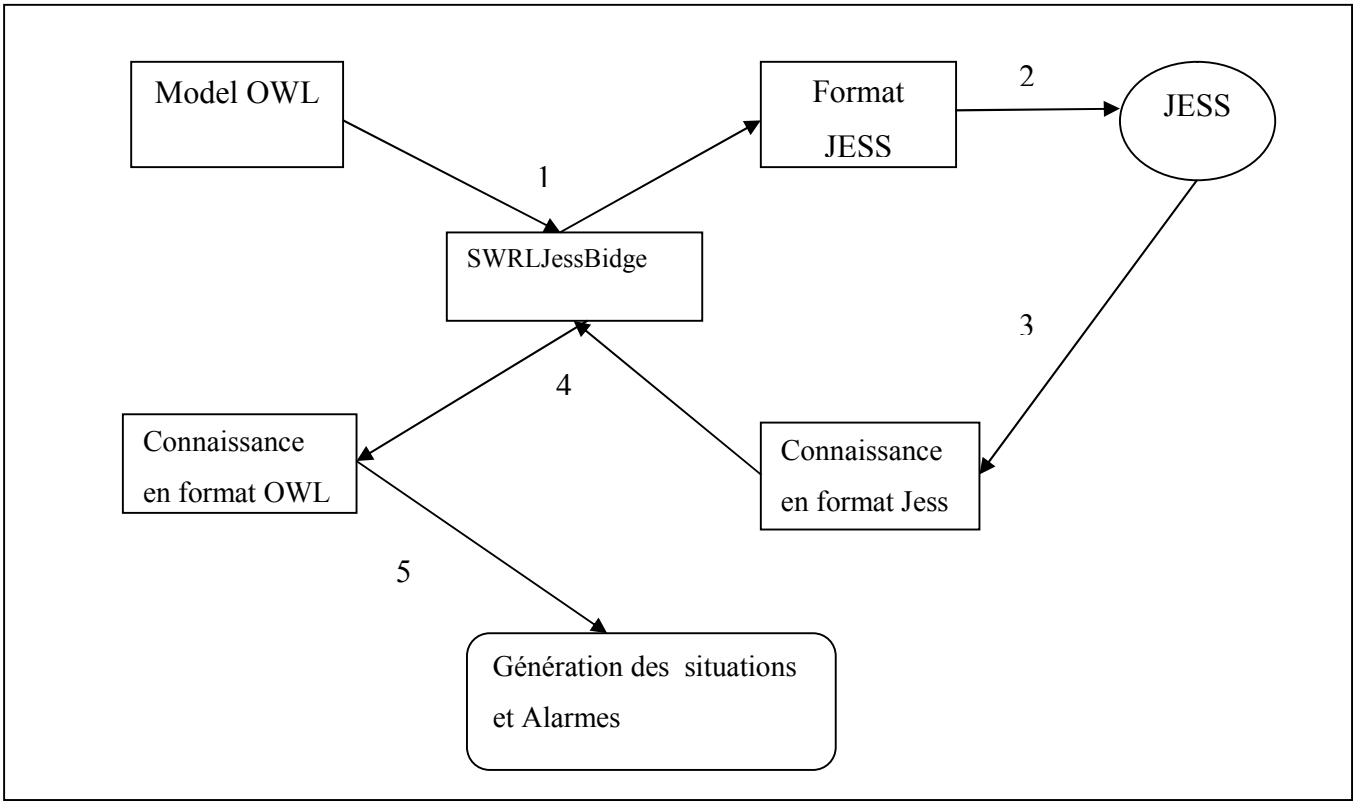


Figure 5.11 Processus de raisonnement sur le contexte

Etape 1 : Le modèle OWL est transformé par le ***SWRLJessBridge*** en format JESS approprié pour le raisonnement.

Etape 2 : JESS réalise l'inférence.

Etape 3 : Après déduction, le modèle OWL sera transformé en format Jess.

Etape 4 : SWRLJessBridge transforme les faits Jess vers le modèle OWL.

Etape 5 (résultats) : génération des situations et alarmes appropriées.

6- Exécution des règles proposées par raisonnement développé dans le système:

1. Description des étapes d'exécution de la règle1 :

La règle suivante génère l'alarme en cas d'une fuite de gaz :

GazSensor(?s) ∧ GazAlert(?a) ∧ hasSensorValue(?s, "HIGH") → ActivateGazAlert(?s, ?a) ∧ hasAlertStatus(?a, "on")

Avant exécution de cette règle nous avons identifiés les états suivants :

1. L'état du capteur ***GazSensor***

```
<GazSensor rdf:ID="GazSensor_1">
  <hasSensitivity rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >true</hasSensitivity>
  <hasDescription rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >gazSensor</hasDescription>
  <instanceName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >gazSensor</instanceName>
  <hasSensorValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >LOW</hasSensorValue>
  <ActivateGazAlert>
    <GazAlert rdf:ID="GazAlert_1">
      <hasAlertStatus rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >on</hasAlertStatus>
      <hasAlertStatus rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Off</hasAlertStatus>
    </GazAlert>
  </ActivateGazAlert>
</GazSensor>
```

2. L'état de l'alerte ***GazAlert_1***

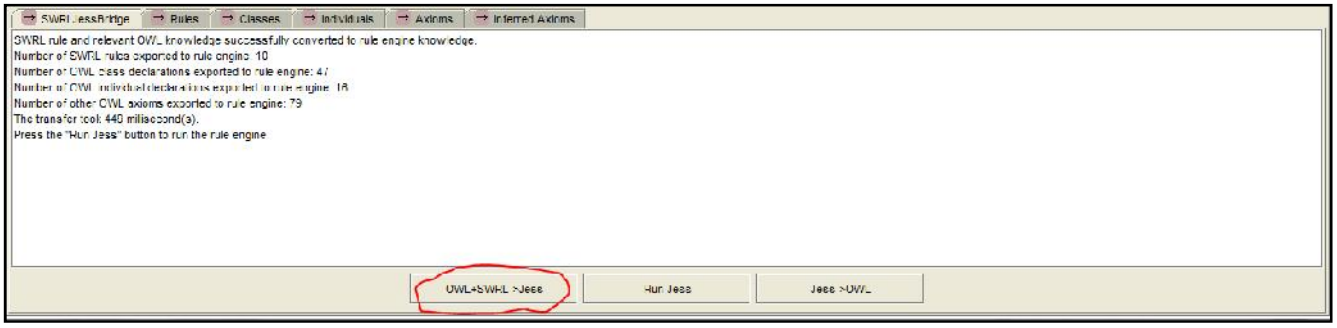
```
<GazAlert rdf:ID="GazAlert_1">
  <hasAlertStatus rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Off</hasAlertStatus>
</GazAlert>
```

3. L'état de la propriété ***ActivateGazAlert***

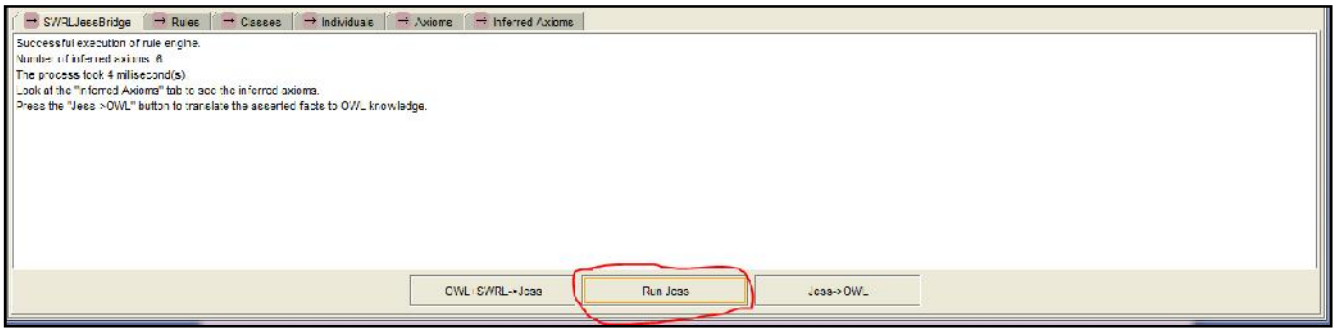
```
<ActivateGazAlert>
  <GazAlert rdf:ID="GazAlert_1">
    <hasAlertStatus rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Off</hasAlertStatus>
  </GazAlert>
</ActivateGazAlert>
```

Pour exécuter cette règle (qui génère l'alarme en cas d'une fuite de gaz) nous passons par les trois phases suivantes :

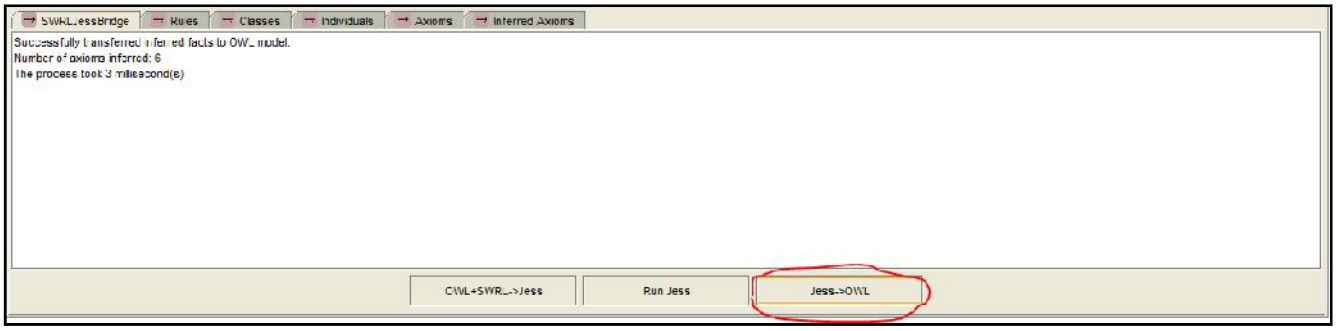
phase1 : transformer les instances OWL vers des faits Jess. En protégé 2000 cette opération est réalisée en appuyant sur le Botton ***OWL+SWRL->Jess***



Phase2 : La deuxième étape consiste à exécuter le moteur d'inférence Jess en appuyant sur le Botton ***Run Jess***



Phase3: La dernière étape consiste à transformer les faits Jess vers des instances OWL pour les préservés dans la base de connaissance en appuyant sur le Botton ***Jess -> OWL.***

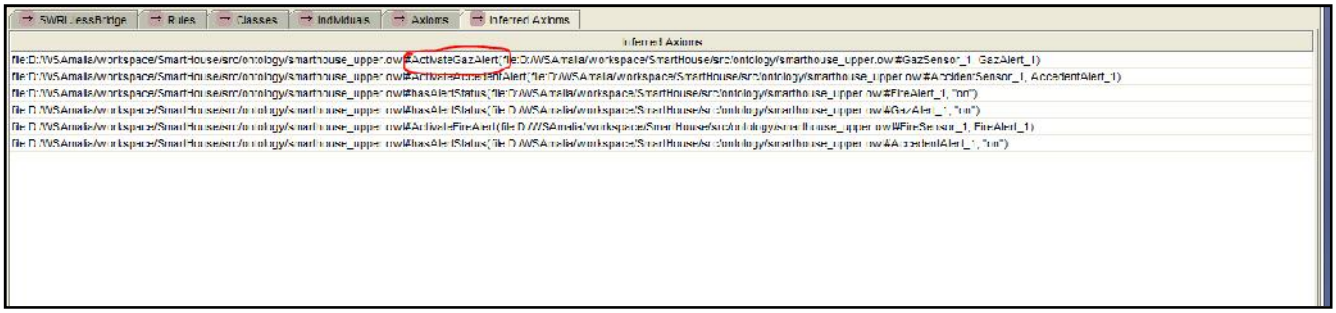


Le résultat de l'exécution de la règle après le changement de la valeur **hasSensorValue** du capteur ***GazSensor_1*** en HIGH est comme suit :

```
<ActivateGazAlert>
  <GazAlert rdf:ID="GazAlert_1">
    <hasAlertStatus rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >on</hasAlertStatus>
```

On voit bien que le statut de l'alerte ***GazAlert_1*** est devenu «ON »

Tous les résultats de l'inférence sont présentés dans l'onglet ***Infered Axiom***



2. Description des étapes d'exécution de la règle2 :

La même chose pour générer une alarmes en cas d'un accèdent on reprend les même étapes .voici la règles :

AccidentSensor(?s) ∧ AccedentAlert(?a) ∧ hasSensitivity(?s, true) → ActivateAccedentAlert(?s, ?a) ∧ hasAlertStatus(?a, "on")

L'état de l'alerte avant l'accédant

```
<ActivateAccedentAlert>
  <AccedentAlert rdf:ID="AccedentAlert_1">
    <hasAlertStatus rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Off</hasAlertStatus>
    <hasAlertTime rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime"
    >2013-03-07T14:10:26</hasAlertTime>
  </AccedentAlert>
</ActivateAccedentAlert>
```

La valeur du capteur ***hasSensitivity*** avant est « «false »

```
<AccidentSensor rdf:ID="AccidentSensor_1">
  <ActivateAccedentAlert>
    <AccedentAlert rdf:ID="AccedentAlert_1">
      <hasAlertStatus rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >on</hasAlertStatus>
      <hasAlertStatus rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Off</hasAlertStatus>
      <hasAlertTime rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime"
      >2013-03-07T14:10:26</hasAlertTime>
    </AccedentAlert>
  </ActivateAccedentAlert>
  <hasSensorValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >HIGH</hasSensorValue>
  <instanceName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >AccidentSensor_1</instanceName>
  <hasSensitivity rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</hasSensitivity>
</AccidentSensor>
```

Après le changement de valeur du capteur ***hasSensitivity*** en true l'état de l'alerte devient « on »

```
<ActivateAccedentAlert>
  <AccedentAlert rdf:ID="AccedentAlert_1">
    <hasAlertStatus rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >on</hasAlertStatus>
    <hasAlertStatus rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Off</hasAlertStatus>
    <hasAlertTime rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime"
    >2013-03-07T14:10:26</hasAlertTime>
  </AccedentAlert>
</ActivateAccedentAlert>
```

3. Description des étapes d'exécution de la règle3

Pour générer une alarme en cas de déclanchement d'un feu, voici la règle :

FireSensor(?s) ∧ FireAlert(?a) ∧ hasSensorValue(?s, "HIGH") → ActivateFireAlert(?s, ?a) ∧ hasAlertStatus(?a, "on")

Avant l'exécution de la règle et le changement de valeur du capteur ***FireSensor l***'état de l'alerte ***FireAlert_1*** est « Off ».

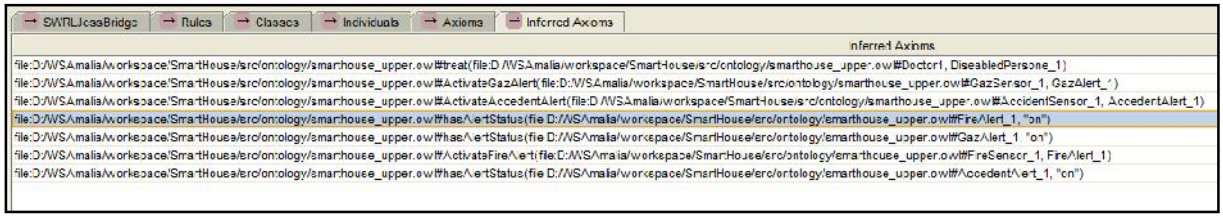
```
1592        <FireAlert rdf:ID="FireAlert_1">
1593          <hasAlertStatus rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
1594          >Off</hasAlertStatus>
1595        </FireAlert>
```

Etat du capteur ***FireSensor*** avant est « «***Low***»

```
    >FireSensor_1</instanceName>
    <hasSensorValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >LOW</hasSensorValue>
    <hasSensitivity rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</hasSensitivity>
  </FireSensor>
```

Après le changement de la valeur ***hasSensorValue*** ver HIGH le Statut du ***FireAlert*** devient « On »

```
<ActivateFireAlert>
  <FireAlert rdf:ID="FireAlert_1">
    <hasAlertStatus rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >On</hasAlertStatus>
  </FireAlert>
</ActivateFireAlert>
```



4. Description des étapes d'exécution de la règle4

Pour le contrôle de la lumière selon l'existence d'une personne voici la règle

DiseabledPersone(?p) ∧ RFIDSensor(?r) ∧ hasSensitivity(?r, true) ∧ Light(?L) ∧ hasDeviceStatus(?L, "Off") ∧ getContexteInformation(?L, ?r) → hasDeviceStatus(?L, "On")

L'état de la lampe avant

```
<Light rdf:ID="Light1">
  <hasDeviceStatus rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Off</hasDeviceStatus>
  <getContexteInformation>
    <RFIDSensor rdf:ID="RFIDSensor_1">
      <hasDescription rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >RfidSensor</hasDescription>
```

La valeur de ***hasSensitivity*** du capteur ***RFIDSensor*** égale « false ».

```
<RFIDSensor rdf:ID="RFIDSensor_1">
  <hasDescription rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >RfidSensor</hasDescription>
  <hasSensitivity rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</hasSensitivity>
  <hasSensorValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >HIGH</hasSensorValue>
  <LocateIn rdf:resource="file:D:/WS/Amalia/workspace/SmartHouse/src/ontology/smarthouse_upper.owl#IndoorSpace_1"/>
</RFIDSensor>
```

Après le changement de la valeur ***hasSensitivity*** en « true » la valeur du ***HasDeviseStatus*** du ***light1*** devient « ***on*** »

```
<Light rdf:ID="Light1">
  <hasDeviceStatus rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >On</hasDeviceStatus>
  <getContexteInformation>
    <RFIDSensor rdf:ID="RFIDSensor_1">
      <hasDescription rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >RfidSensor</hasDescription>
      <hasSensitivity rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
      >true</hasSensitivity>
      <hasSensorValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >HIGH</hasSensorValue>
```

5. Description des étapes d'exécution de la règle5

Pour le contrôle de la température de la pièce voici la règle

DiseabledPersone(?p) ∧ TemperatureSensor(?s) ∧ hasSensorValue(?s, "HIGH") ∧ Air-conditioner(?ac) ∧ hasDeviceStatus(?ac, "Off") ∧ getContexteInformation(?ac, ?s) → hasDeviceStatus(?ac, "on")

L'état du climatiseur ***Air-Conditionair*** avant lorsque la température a une valeur inferieur.

```
<Air-conditioner rdf:ID="Air-conditioner1">
  <instanceName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Air-Conditioner1</instanceName>
  <hasDescription rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Air-conditioner1</hasDescription>
  <hasTimeStamp rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime"
  >2013-03-07T12:49:57</hasTimeStamp>
  <LocateIn>
    <IndoorSpace rdf:ID="IndoorSpace_1">
      <hasLongitide rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
      >12.02</hasLongitide>
      <instanceName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >IndoorSpace_1</instanceName>
      <hasLatitide rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
      >158.23</hasLatitide>
    </IndoorSpace>
  </LocateIn>
  <hasDeviceStatus rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Off</hasDeviceStatus>
</Air-conditioner>
```

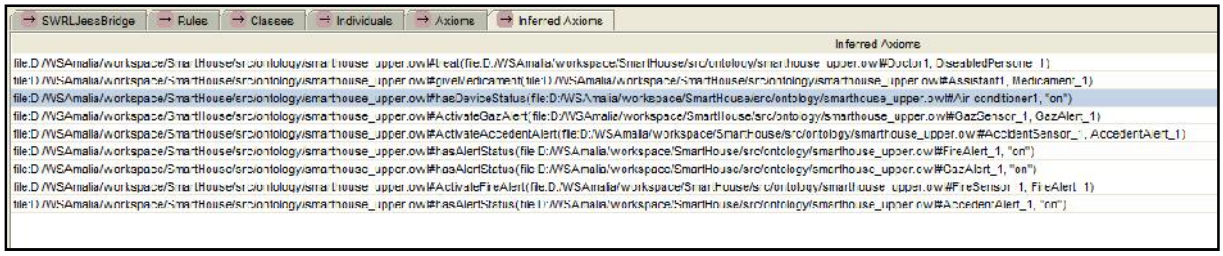
La valeur capteur ***TemperatureSensor_3*** avant

```
<TemperatureSensor rdf:ID="TemperatureSensor_3">
  <LocateIn rdf:resource="file:D:/WSAmalia/workspace/SmartHouse/src/ontology/smarthouse_upper.owl#IndcorSpace_1"/>
  <hasSensitivity rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >true</hasSensitivity>
  <hasSensorValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >LOW</hasSensorValue>
  <hasDescription rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >temperatureSensor</hasDescription>
</TemperatureSensor>
```

L'état du climatiseur ***Air-conditionair1*** après le changement de la valeur du capteur en « HIGH »

```
    <TemperatureSensor rdf:ID="TemperatureSensor_3">
      <LocateIn rdf:resource="file:D:/WSAmalia/workspace/SmartHouse/src/ontology/smarthouse_upper.owl#IndoorSpace_1"/>
      <hasSensitivity rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
      >true</hasSensitivity>
      <hasSensorValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >HIGH</hasSensorValue>
      <hasDescription rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >temperatureSensor</hasDescription>
    </TemperatureSensor>
  </getContexteInformation>
```

```
<Air-conditioner rdf:ID="Air-conditioner1">
  <instanceName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Air-Conditioner1</instanceName>
  <hasDescription rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Air-conditioner1</hasDescription>
  <hasTimeStamp rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime"
  >2013-03-07T12:49:57</hasTimeStamp>
  <LocateIn rdf:resource="file:D:/WSAmalia/workspace/SmartHouse/src/ontology/smarthouse_upper.owl#IndoorSpace_1"/>
  <hasDeviceStatus rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >On</hasDeviceStatus>
```



6. Description des étapes d'exécution de la règle 6

Pour lancer une alerte pour la prise de médicament voici la règle :

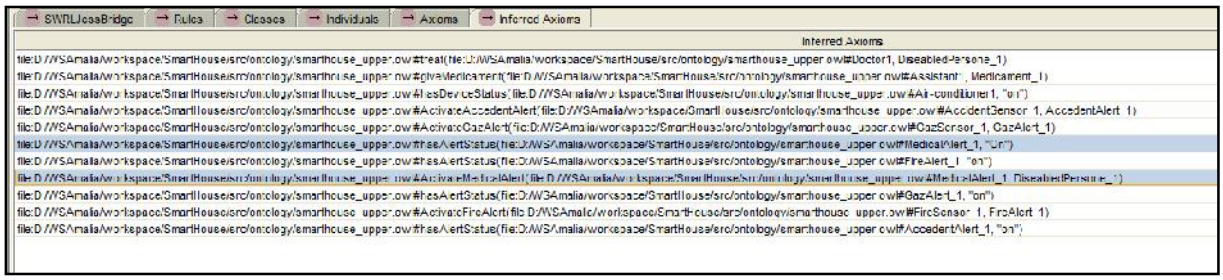
DiseabledPersone(?p) ∧ Medicament(?m) ∧ hasMedicament(?p, ?m) ∧ MedicalAlert(?a) ∧ hasAlertStatus(?a, "Off") ∧ timeofTak(?m, "now") → ActivateMedicalAlert(?a, ?p) ∧ hasAlertStatus(?a, "On")

Etat de l'alerte ***MedicalAlert*** avant :

```
</ActivateMedicalAlert>
<hasAlertStatus rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Off</hasAlertStatus>
```

Une fois le temps du médicament ***timeofTak*** devient « now », Le statut de ***MedicalAlert*** devient « On » et la relation ***ActivateMedicalAlert*** est activée.

```
<hasAlertStatus rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>On</hasAlertStatus>
<hasAlertStatus rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

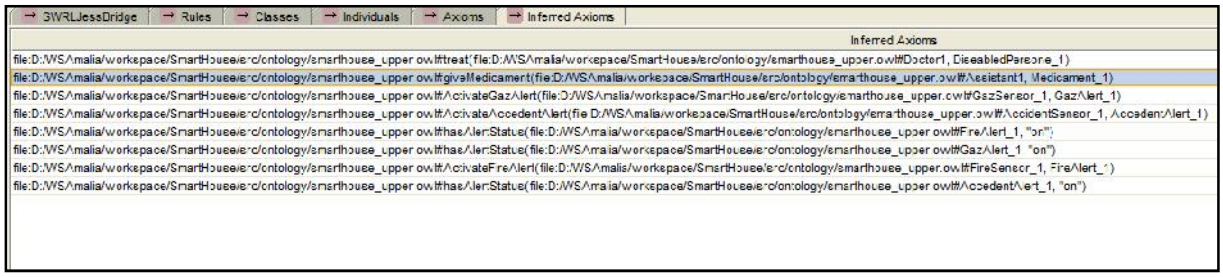


7. Description des étapes d'exécution de la règle7

Pour le don de médicament au personne handicapée voici la régle :

DiseabledPersone(?p) ∧ Assistant(?a) ∧ hasMedicament(?p, ?m) ∧ Medicament(?m) ∧ hasActivity(?p, Discharge__8) ∧ timeofTak(?m, "now") → giveMedicament(?a, ?m)

Une fois le médicament associé à la personne handicapée prend la valeur ***timeofTak*** ->now le système génère la relation ***giveMedicament***



Pour lancer le traitement de la personne handicapé par le médcin voici la régles :

DiseabledPersone(?p) ∧ Doctor(?d) ∧ hasStat(?p, "Seek") → treat(?d, ?p)

Après le changement de l'état de personne en «***Seek*** » le système infère la relation ***treat*** entre ***DiseabledPersone_1*** et ***Doctor1***.

```
<Doctor rdf:ID="Doctor1">
  <treat>
    <DiseabledPersone rdf:ID="DiseabledPersone_1">
      <LocateIn rdf:resource="file:D:/WSAmalia/workspace/SmartHouse/src/ontology/smarthouse_upper.owl#IndoorSpace_1"/>
      <hasName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >amar</hasName>
      <hasCognitiveAbilityLevel rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Low</hasCognitiveAbilityLevel>
      <hasStat rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Seek</hasStat>
      <hasHeight rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
      >150.0</hasHeight>
      <instanceName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >DP1</instanceName>
      <hasGender rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Male</hasGender>
      <hasPhysicalAbilityLevel rdf:datatype='http://www.w3.org/2001/XMLSchema#string'
      >Low</hasPhysicalAbilityLevel>
      <hasfamilyName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Amari</hasfamilyName>
      <hasActivity>
        <Discharge_ rdf:ID="Discharge__8"/>
      </hasActivity>
      <hasPhoneNumber rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >055889878</hasPhoneNumber>
      <hasSeverityLevel rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Low</hasSeverityLevel>
      <hasLevel rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >1</hasLevel>
      <hasAge rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
      >60.0</hasAge>
      <hasMedicament>
        <Medicament rdf:ID="Medicament 1">
          <desingation>Medicaqment1</desingation>
          <timeofTak rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime'
          >2013-05-28T13:28:27</timeofTak>
        </Medicament>
      </hasMedicament>
    </DiseabledPersone>
  </treat>
```

8. Description des étapes d'exécution de la règle8

Pour lancer une alerte de rappelle pour la prise de médicament voici la règles :

DiseabledPersone(?p) ∧ Medicament(?m) ∧ hasActivity(?p, Discharge__8) ∧ dontTakMedicament(?p, ?m) ∧ MedicalAlert(?a) → remembreMedicalTake(?a, ?p) ∧ ActivateMedicalAlert(?a, ?p)

Etat de l'alerte ***MedicalAlert*** avant :

```
</ActivateMedicalAlert>
<hasAlertStatus rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Off</hasAlertStatus>
```

Une fois le système vérifie que la personne n'a pas pris son médicament avec la relation ***dontTakMedicament*** , il déclenche l'alerte ***MedicalAlert*** puis rend son statut

```
<hasAlertStatus rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>On</hasAlertStatus>
<hasAlertStatus rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

5.5 Conlusion

Dans ce chapitre nous avons présenté le mécanisme de raisonnement sur le contexte pour gérer les différents situations de l'assistance de la personne handicapée tel que la notification sur son état ou pour lancer une Alarme en cas d'accèdent ou un incendie dans la maison.

Pour l'implémentation de ce mécanisme nous avons utilisé le langage SWRL (Semantic Web Rule Language) qui permet la formalisation des règles de raisonnement dans le même format des connaissances de l'ontologie.

On plus nous avons utilisé l'éditeur **SWRLTab** qui offre un environnement de développement dans l'outil Protégé pour travailler avec des règles SWRL et qui permet l'intégration du moteur d'inférence **Jess** pour l'exécution des règles**.**

Conclusion et perspective

L'objectif de notre travail est de proposer une modélisation de l'information de contexte dans un environnement domotique en utilisant les ontologies comme modèle de représentation et le développent d'un mécanisme de raisonnement sur le contexte basé sur l'ontologie pour assurer l'assistance des personnes handicapées dans leur maison.

Nous avons commencé par un état de l'art sur les nouvelles technologies lies à l'intelligence ambiante et les différentes approches de modélisation de l'information de contexte dans telles technologies.

Nous avons présenté la notion d'ontologie, ses composants, le cycle de son développement et comment elle est utilisée dans la modélisation de contexte dans un environnement sensible au contexte avec une démonstration de quelques ontologies génériques de modélisation de contexte, comme **SOCAM** ,**CoOL** et **SOUPA**, bien que ces modèles saisissent les concepts généraux de contexte et fournissent un raisonnement sur le contexte, ils ne sont pas spécifiques pour le domaine de l'assistance des personnes handicapées a domicile et les maisons intelligentes *SmartHose* ce qui nous nécessite l'adaptation de ces dernières ou le développement des nouvelles ontologies pour ce domaine d'application.

En plus on a développé un mécanisme de raisonnement sur le contexte pour notre système qui permet de prendre en charge le contexte de l'utilisateur et agir sur l'environnement pour le conseil et l'assistance de ce dernier par la suggestion des actions qu'il juge les mieux appropriées ce mécanisme et la démonstration des résultats a été réalisées avec l'environnement SWRLTab de protégé2000.

Le travail que nous avons réalisé dans ce mémoire de magistère représente un début de recherche dans un environnement domotique qui peut être enrichi par plusieurs autre travaux future comme :

- Le renforcement et l'enrichissement de l'ontologie réalisée afin de capturer le maximum d'information de contexte.

-l'implémentation de ce système dans un environnement domotique réel avec les capteurs et les diapositives nécessaires pour l'assistance des personnes handicapées.

-l'intégration de paradigme agent a notre système pour bénéficier de ces multiples avantages et sa puissance dans la mise en œuvre des systèmes complexes

Bibliographies

[1] N.Sahli : Agent-based Middlewares for Context in Proceedings of the First International Conference on Context-aware Adaptation Mechanisms for Pervasive and Ubiquitous Services CAMPUS. Oslo Norway, Volume 11, 2008.

[2] J. Pierson : Une infrastructure de gestion de l'information de contexte pour l'intelligence ambiante. Thèse de doctorat. Université Joseph Fourier - Grenoble 1 2009.

[3] T. Dujardin, J. Rouillard, J.C. Routier, J.C. Tarby, et al : Gestion intelligente d'un contexte domotique par un système multi-agents. Acte des Journées Francophones sur les Systèmes Multi-Agents. France, 2011.

[4] K. Haigh, J. Phelps and C. Geib : An Open Agent Architecture for Assisting Elder Independence. In Proceedings of the first international joint conference on Autonomous agents and multiagent systems "AAMAS" 2002.

[5] B. Bertin, E. Jeanne& N. Crespi. Context-aware systems : a case study. In the Proceedings of international Conference on Digital Information and Communication Technology and Its Applications, Page 1–15. Dijon, France, 2011

[6] A. Roy : Agent-Based AmI System Case Study: The Easy Line + Project. In the Proceedings of Trends in Practical Applications of Agents and Multiagent Systems Conference. Page 157-164. Salamanca, Spain, 26-28 April 2010.

[7] C. Lim, P. Anthony : Applying multi-agent system in a context aware smart home. In Proceedings of Borneo Science24 Conference, Page 53-64. Curtin Sarawak Malaysia 2009

[8] W. Xiaohang. The Context Gateway: A Pervasive Computing Infrastructure for Context Aware Service. Research Report, School of Computing, National University of Singapore & Context -Aware Dept., Institute for Infocomm Research, November, 2003.

[9] T. V. Nguyen, H. A. Nguyen and D. C. Nguyen : "Development of a Context Aware Virtual Smart Home Simulator", In Proceedings of The 3rd International Conference on Ubiquitous Information Technologies and Applications (ICUT). Suwon, Republic of Korea,2008.

[10] G. Kannan and S. Vijayakumar: “Smart home tested for disabled people,” TIFAC CORE, Velammal Engineering College, Chennai, Mobile and Pervasive Computing Conference -2008.

[11]B. Kaluža ,M.Luštrek Gams, Matjaž : Context-Aware MAS to Support Elderly People, in Proceeding of 11th International Conference on Autonomous Agents and Multiagent Systems, Pages 1485-1486. , Richland, 2012.

[12] W. Yao, C. Chu, A. Kumar and Z. Li : Using ontology to support context awareness in healthcare, in Proceedings of the 19th Workshop on Information Technologies and Systems: Page 14-15 , Phoenix, AZ, USA, 2009.

[13] F.Paganelli and D.Giuli : An Ontology-based Context Model for Home Health Monitoring and Alerting in Chronic Patient Care Networks. In the Proceedings of 21st International Conference on Advanced Information Networking and application Workshope, Niagara Falls, Canada 2007.

[14]F. Paganelli and D.Giuli : A context-aware service platform to support continuous care networks for home-based assistance. In the Proceedings of 21st International Conference on Advanced Information Networking and application Workshop, Niagara Falls, Canada, 2007.

[15] K. Skillen, L.Chen and C.Nugent : Ontological User Profile Modeling for Context-Aware Application Personalization Journal of Springer, Volume 7656, pp 261-268 2012.

[16] A.Wang, X. Hang Gu, T. Zhang, D. Qing Pung and H. Keng : Ontology Based Context Modeling and Reasoning using OWL. In the Proceedings of PERCOMW ’04 Second IEEE Annual Conference on Pervasive Computing and Communication Workshops, DC, USA , 2004.

[17]D. Zhang,T.Gu,X.Wang : Enabling Context-aware Smart Home with Semantic Web Technologies. International Journal of Human-friendly Welfare Robotic Systems,Vol 6 No 4,2005.

[18] M.Ziefle,C.Röcker and A.Holzinger: Medical Technology in Smart Homes. In Proceedings of 35th IEEE Annual Computer Software and Applications Conference Workshops, Munich Germany, 2011.

[19] A. Hoszu Context sensitive multiagent system. Thesis technical university of cluj-napoca, 2008.

[20] A. Florea, F. Seghrouchni :A Context-Aware Multi-Agent System for AmI Environment.s, In the Proceedings of Computer Science and Information Systems Conference. Warsaw, Poland, 2011.

[21] K. Haigh, J. Phelps and C. Geib: An Open Agent Architecture for Assisting Elder Independence. In the Proceedings of first international joint conference on Autonomous agents and multiagent systems. Bologna, Italy, 2002.

[22] G G. Kannan and S. Vijayakumar, "Smart home tested for disabled people," TIFAC CORE. In Proceedings of Velammal Engineering College, Chennai, Mobile and Pervasive Computing-2008.

.

[23] A.ANDRIATRIMOSON : Assistance robotisée à la personne en environnement coopérant. Thèse de doctorat d'université d'Evry Val D'Essonne, France. 2012.

[24] P. Carrera : Étude de comportements coopératifs pour l'intelligence ambiante : Application à la gestion de crises. Mémoire de Master Université Carlos III de Madrid, Espagne, 2009.

[25] h. Amalia :Diploma project smart house: context sensitive multi-agent system, technical university of clu-jnapoca , 2008.

[26] R. Bergamann : Ambient Intelligence for Decision Making in Fire Service Organizations, Journal of Springer, Volume 4794. Berlin, Germany , 2007.

[27] B. Nieto, A. Carretero de los Ángeles, Noelia. 2004 : Intelligence Ambiante , Rapports de recherche , Centre des Technologies, CEDITEC-FUNDETEL, Madrid Espagne, 2004.

[28] Marreiros, Goreti, Ramos, Carlos et Neves, José. 2005. Dealing with Emotional Factors in Agent Based Ubiquitous Group Decision. Lecture Notes in Computer Science: Embedded and Ubiquitous Computing - EUC 2005, Volume 3823. Berlin, Germany : Springer Berlin Heidelberg, 2005.

[29] A.Dey: Providing Architectural Support for Building Context-Aware Applications. Ph.D. thesis, Georgia Institute of Technology, Nov. 2000.

[30] G. Kouadri, M. Efaoui, J. Pasquier-Rocha, P. Brézillon : Context-Aware Computing: A Guide for the Pervasive Computing Community, Pervasive Services. In Proceedings of ICPS 2004 IEEE/ACS International Conference, Beirut, Lebanon. July 2004.

[32]O. Gassmann, H. Meixner: Sensors in Intelligent Buildings. In the Proceedings of Wiley-VCH Verlag GmbH, Weinheim, Germany, 2001.

[33] G. Rey : Contexte en Interaction Homme-Machine, le contexteur, Ph.D. thesis, Fdration IMAG-Universit Joseph Fourier - Grenoble I, France. 2005.

[34] T. Strang ,C. Linnhoff-popien :A Context Modeling Survey. In the Proceedings of UbiComp, Tokyo, Japan, 2004

.

[35] J. Bauer : Identification and Modeling of Contexts for Different Information Scenarios in Air Traffic. In the Proceedings of Diplomarbeit, Berlin, German. 2003.

[36] T.Strang1 C.Linnho-Popien, K.Frank, CoOL: A Context Ontology Language to enable Contextual Interoperability. In the Proceedings of Distributed Applications and Interoperable Systems, DAIS'03 ,Paris, France , 2003.

[37] J. mccarthy : Notes on formalizing contexts. In the Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence, pp. 555–560, San Mateo, California, 1993.

[38] H.Chen: An Intelligent Broker Architecture for Pervasive Context-Aware Systems. PhD Thesis, University of Maryland, Baltimore, 2004.

[39] B.Chihani, E. Bertin, F.Jeanne and N.Crespi, Noel Context-aware systems: a case study, in Proceedings of The International Conference on Digital Information and Communication Technology and its Applications, Dijon , France. 2011 .

[40] V.Sinderen, M.J., van Halteren, A.T., Wegdam, M., Meeuwissen, H.B., Eertink, E.H.: Supporting context-aware mobile applications: an infrastructure approach, Journal of IEEE Communications Magazine, Volume 44 Num 9, 2006 .

[41] Jena2 Semantic Web Toolkit: http://www.hpl.hp.com/semweb/jena2.htm.

[42] T.Gua, H.Keng , D.Qing Zhangb: A service-oriented middleware for building context-aware services, Journal of Network and Computer Applications archive Volume 28 , Issue 1, January 2005.

[43] S.Thomas, C.Linnho-Popien, K. Frank, CoOL: A Context Ontology Language to enable Contextual Interoperability, In Proceedings of Distributed Applications and Interoperable Systems, DAIS'03, Paris ,France. 2003.

[44] Hal : Home automated living, http:// automatedliving.com.

[45] Y. Bellik, I.Rebaï, E.Machrouh,Y.Barzaj, C. Jacquet, G. Pruvost, and J-P. Sansonnet: Multimodal interaction within ambient environments: An exploratory study. In Proceedings of INTERACT'09: 12th IFIP TC13 Conference on Human-Computer Interaction, pages 89–92. South Africa, 2009.

[46] J. CookDiane, Y. blood Michael, O. HeiermanEdwin, G. Karthik, R. Sira, L. Andrey and K. Farhan.Mavhome: An agent-based smart home, Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, DC, USA , 2003.

[47] Gaia Project http://gaia.cs.uiuc.edu/ last visited Sept. 2007.

[48] D. Salber, A. K. Dey, G. Abowd. The context toolkit: aiding the development of context-enabled applications. In Proceedings of the SIGCHI conference on Human factors in computing systems. pp. 434–441. ACM, 1999.

[49] P. Maes: Agents that reduce work and information overload, Journal of ACM 37(7):30–40, 1994

[50] J. Wohltorf, R. Cissee, A. Rieger, H. Scheunemann. BerlinTainment: An Agent-Based Serviceware Framework for Context-Aware Services. In Proceedings of Symposium on Wireless Communication Systems Conference, Paris France , 2004

[51] W. Qin, Y. Suo, Y. Shi. CAMPS: A Middleware for Providing Context-Aware Ser- vices for Smart Space, Springer LNCS 3947, pp. 644–653, , 2006

[52] R. Plesa, L. Logrippo: An Agent-Based Architecture for Context-Aware Communication. In the Proceedings of the 21st Int. Conference, on Advanced Information Networking and Applications Workshops. pp. 133–138. IEEE, 2007

[53]H. Takahashi, T. Suganuma, N. Shiratori. AMUSE: An Agent-based Middleware for Context-aware Ubiquitous Services, In Proceedings of the 11th Conference, on Parallel and Distributed Systems. Pp. 743–749. Fuduoka, Japan, 2 Volumes. IEEE Computer Society 2005 .

[54]E. Kovacs, K. R¨ ohrle, B. Schiemann: Adaptive Mobile Access to Context-Aware Services, In Proceedings of the 1st International Symposium ASAMA. P. 190. IEEE, 1999.

[55]H. Harroud, A. Karmouch: A Policy Based Context-Aware Agent Framework to Support Users Mobility, In Proceedings of AICT-SAPIR-ELETE, Page 177–182, Lisbon, Portugal ,2005.

[56] JADE Tutorial for beginners, organized by the JADE Board, http://jade.tilab.com

[57] http://www.linternaute.com/dictionnaire/fr/definition/handicape/

[58] Protege Project. http://protege.stanford.edu.

[59] R.Anas & M.Wissam & I. Anis : Utilisation d'une ontologie distribuée sur des Pockets PC ". Mémoire de Master, université Télécom ParisSud, 2004.

[60] T. Dujardin : De l'apport des ontologies pour la conception de systèmes multi-agents ouverts. Mémoire de Master Université des Sciences et Technologies de Lille, France, 2012 .

[61] A. Miron, C. Capponi, J.Gensel : Rapprocher AROM de OWL, dans le Proceedings de la conférence Langages et Modèles à Objets, France. mars 27-29, 2007.

[62] Barthès, J.-P : La gestion des concepts et du vocabulaire dans l'entreprise, terminologies et ontologies : état de l'art. Rapport de veille technologique IIIA-98-VT9, 1998.

[63] H. Mihoubi, A.Simonet, M.Simonet: An Ontology Driven Approach to Ontology Translation. In the Proceedings of International Conference on Database and Expert Systems Applications DEXA'2000, Page 573-582, London,-2000

[64] M.King : The enterprise ontology, Journal of The Knowledge Engineering Review 13 Issue 1, pp. 31–89, 1998.

[65] A. Maedche:Ontology Learning for the Semantic Web Journal, Springer 2002.

[66] B. Biébow, S.Szulman, B. Clément TERMINAE : A Linguistics-Based Tool for the Building of a Domain Ontology Journal, Springer 1999.

[67] M. FERNÀNDEZ, A .GÓMEZ-PÉREZ: Bulding a chemical ontology using methontology and the ontology design environment. In the Proceedings of IEEE Intelligent System and their Applications Conference. 14(1), 37–45 1999.

[68] C. Cormier, C. Kassel and J.Nobécourt : Evaluation de langages opérationnels de représentation d'ontologies, dans les Actes des journées Ingénierie des Connaissances : IC'2001, Grenoble, France.

[69] J.F Sowa :Conceptual Structures :Information Processing in Mind and Machines Addison, Book Adison-Wesley Longman Publishing Co., Inc. Boston, MA, USA 1984.

[70] H. Jean-Paul et al : Le raisonnement en Intelligence Artificielle. Dans le Proceedings d'InterEditions, 1991

.[71] R. Gruber : Ontolingua: A mechanism to support portable ontologies. Technical Report KSL, Stanford University, Knowledge Systems Laboratory. Pp 91-66, 1992.

[72] V.Karimi ' Semantic Web Rule Language ', Tutorial du langage SWRL 2008.