HAL
open science

Une infrastructure de gestion de l'information de contexte pour l'intelligence ambiante

Jérôme Pierson

To cite this version:

Jérôme Pierson. Une infrastructure de gestion de l'information de contexte pour l'intelligence ambiante. Informatique [cs]. Université Joseph-Fourier - Grenoble I, 2009. Français. NNT : . tel-00441121

HAL Id: tel-00441121

https://theses.hal.science/tel-00441121

Submitted on 14 Dec 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE

présentée à
l'Université Joseph Fourier - Grenoble 1

pour obtenir le grade de
DOCTEUR

spécialité
Informatique

intitulée
Une infrastructure de gestion de l'information de contexte pour l'intelligence ambiante

présentée et soutenue publiquement le 5 octobre 2009 par
Jérôme Pierson

devant le jury composé de :

Joëlle Coutaz	**Examinateur**
Olivier Boissier	**Rapporteur**
Jérôme Euzenat	**Directeur de thèse**
Sylvain Giroux	**Rapporteur**
Fano Ramparany	**Co-directeur**
Jean-Paul Sansonnet	**Examinateur et président du jury**

Résumé

Les environnements d'intelligence ambiante servent d'interface entre les applications et les utilisateurs. Ces applications doivent prendre en compte le contexte dans lequel les utilisateurs évoluent (le lieu, la position sociale ou hiérarchique ou l'activité par exemple) pour adapter leur comportement. Il doit exister un flux d'informations de l'environnement vers les applications. Ces applications doivent pouvoir prendre en compte dynamiquement l'arrivée de nouveaux éléments dans l'environnement (utilisateurs ou dispositifs), et les informations de contexte en provenance de l'environnement doivent pouvoir parvenir aux applications entrantes ; ces flux d'informations ne peuvent pas être déterminés à l'avance et doivent se construire pendant l'exécution. Les modèles de gestion de l'information de contexte existants ne traitent pas ou peu cet aspect dynamique de l'informatique diffuse.

Nous proposons d'utiliser les technologies du web sémantique pour décrire et rechercher ces informations : l'information de contexte est exprimée en RDF et décrite par des ontologies OWL. Ces technologies, parce qu'elles sont fondées sur l'hypothèse du monde ouvert, assurent l'ouverture du système et la prise en compte de dispositifs hétérogènes. Nous montrons qu'à l'aide d'un protocole simple qui permet à chacun des dispositifs et des applications d'exhiber sur le réseau un modèle des informations de contexte qu'il produit ou qu'il recherche et de s'identifier, toutes les applications de l'environnement satisfont leurs besoins en informations de contexte. De surcroît, l'ouverture des langages de description d'ontologies permet l'extension des descriptions de contexte à tout moment et les technologies d'alignement d'ontologies permettent l'utilisation d'ontologies développées indépendamment.

Nous avons implémenté un composant pour la gestion de l'information de contexte fondé sur ce modèle. Puis nous avons développé une architecture distribuée où les dispositifs et les applications embarquent ce composant et exposent un modèle de l'information de contexte qu'ils recherchent ou produisent. Nous avons montré comment cette architecture permet d'accepter sans interruption de nouveaux composants.

Abstract

In a pervasive computing environment, the environment itself is the interface between services and users. Using context information coming from sensors, location technologies and agregation services, applications adapt their run-time behaviour to the context in which users evolve (e.g., physical location, social or hierarchical position, current tasks as well as related information). These applications have to deal with the dynamic integration in the environment of new elements (users or devices), and the environment has to provide context information to newly designed applications. We study and develop a dynamic context management system for pervasive application. It is flexible enough to be used by heterogeneous applications and to run dynamically with new incoming devices.

We have designed an architecture in which context information is distributed in the environment. Each device or service implements a context manager component in charge of maintaining its local context. It can communicate with other context manager components : some of them are context information producers, some of them are context information consumers and some of them are both. We have defined a simple protocol to allow a consumer to identify and determine the producer for the information it needs. Context manager components express their context information using an OWL ontology, and exchange RDF triples with each other. The openness of ontology description languages makes possible the extension of context descriptions and ontology matching helps dealing with independently developed ontologies. Thus, this architecture allows for the introduction of new components and new applications without interrupting what is working.

Remerciements

Je tiens à exprimer tout d'abord mes remerciements aux membres du jury, qui ont accepté d'évaluer mon travail de thèse : Jean-Paul Sansonnet, Joëlle Coutaz, Sylvain Giroux et Olivier Boissier.
Je tiens à remercier Jérôme Euzenat pour avoir accepté de diriger cette thèse et dont l'aide précieuse m'a été indispensable sur le plan scientifique. Je tiens également à le remercier pour sa confiance et ses grandes qualités humaines qui ont permis de mener à bout cette thèse. Son soutien s'est avéré déterminant pour mener ce travail à terme. Merci à Fano Ramparany pour avoir encadré cette thèse et pour m'avoir accueilli au sein d'Orange Labs. Sa confiance, son soutien et la sympathie qu'il m'a témoignés ont été un facteur déterminant au bon déroulement de ces cinq longues années de thèse.
Merci également à Gilles Privat, qui m'a permis d'intégrer le projet Amicom, ainsi qu'a tout les participants du projet (Mathieu, Anne, André, Thibault ...).
Je tiens enfin à remercier les amis, thésards ou non qui m'ont aidé : Matthieu, Rudy, Ludovic, Benoît, Yoann, Aurelien, Jean-Luc, Malik, Emmanuel, et j'en oublie certainement.
Un remerciement particulier pour Heidi qui a vécu à mes côtés les pires et les meilleurs moments de cette thèse. Finalement j'adresse un grand merci à toute ma famille qui a toujours été présente lorsque j'en ai eu besoin, en particulier à mon frère, à mon père et à ma mère.

Table des matières

Introduction **1**

1 Intelligence ambiante 1

2 Exemples d'utilisation du contexte dans la sphère domestique 2

3 Le contexte dans les environnements d'intelligence ambiante 2

4 Contributions 4

5 Organisation du mémoire 5

I Contexte et informations de contexte : représentation, utilisation et infrastructures dédiées à leur gestion **7**

1 Contexte et informations de contexte **11**

1.1 Définitions et utilisations du contexte en intelligence ambiante 11

1.2 Contexte, intelligence artificielle et représentation de connaissances 19

1.3 Bilan 20

2 Des infrastructures pour la gestion des informations de contexte en intelligence ambiante **23**

2.1 Vocabulaire 24

2.2 Périmètre de l'étude 24

2.3 Des infrastructures sans ontologie 29

2.4 Des infrastructures utilisant des ontologies 37

2.5 Bilan 56

II Gestion dynamique des informations de contexte en intelligence ambiante : architecture distribuée et modèles sémantiques **61**

3 Architecture distribuée dans un monde ouvert **65**

3.1 Architecture de gestion pair à pair 65

3.2 Fonctionnement 69
3.3 Bilan 72

4 Représentation des informations de contexte dans un monde ouvert 75
4.1 Modèle pour l'information de contexte 75
4.2 Modèle de contexte extensible 76
4.3 Exploiter des ressources hétérogènes 79
4.4 Bilan 82

5 Mise en œuvre 87
5.1 Infrastructure multi-agents 87
5.2 Gestion des modèles sémantiques d'informations de contexte 91
5.3 Bilan 93

6 Expérimentation 95
6.1 Présentation 95
6.2 Scénario 96
6.3 Applications et dispositifs du démonstrateur 97
6.4 Une infrastructure pour les environnements d'intelligence ambiante 102
6.5 Correspondances entre ontologies 104
6.6 Démonstrateur 108
6.7 Bilan 114

Conclusion 117
1 Comparaisons avec l'existant 117
2 Discussion 119

Demander si un ordinateur peut penser revient à
demander si un sous-marin peut nager.
(Anonyme)

Introduction

1 Intelligence ambiante

Ces travaux de thèse s'inscrivent dans le domaine de l'intelligence ambiante. À l'origine, l'informatique ubiquitaire [1] à été proposée par Weiser [84]. D'après ses travaux, les ordinateurs deviennent invisibles dans notre environnement et sont intégrés dans les objets de la vie courante. Quand hier, les utilisateurs partageaient le même ordinateur, la même ressource informatique, aujourd'hui déjà, l'utilisateur dispose de son ordinateur personnel. De plus, chacun d'entre nous est bien souvent équipé (au moins) d'un dispositif (souvent portable) ayant des capacités de communication avec un ou plusieurs réseaux et des capacités de calcul. L'informatique se disperse dans l'environnement, elle va devenir invisible jusqu'à n'apparaître qu'en arrière plan de notre conscience [84]. Cette orientation a donné naissance à plusieurs interprétations, qui constituent aujourd'hui autant de courants de l'informatique : l'informatique sensible au contexte [2], l'informatique tangible [3], l'informatique diffuse [4] et l'intelligence ambiante [5].

Le développement de l'informatique sensible au contexte constitue l'un des développements majeurs de la vision de Weiser. Ces nouveaux systèmes interactifs ne sont pas destinés à être utilisés statiquement derrière un bureau, le paradigme d'interaction change [3] et l'environnement devient potentiellement dynamique (situation de mobilité, multi-utilisateurs...). Les concepteurs de ces systèmes interactifs ne pouvant pas prévoir dès la conception leur environnement d'exécution, il s'agit de concevoir des systèmes capables de recueillir et de traiter des informations sur cet environnement (les informations de contexte) afin d'y adapter leur comportement pendant l'exécution. En généralisant en dehors de la problématique de l'interaction, le but étant de rendre les systèmes informatiques plus "intelligents", en leur permettant d'appréhender leur environnement d'utilisation (informatique, physique, social...). L'intelligence ambiante affirme la volonté de rendre l'informatique invisible en la diluant dans l'environnement quotidien. De façon plus précise, elle regroupe plusieurs idées : l'ubiquité qui se réfère à la présence dans l'environnement de multiples appareils distribués et interconnectés (informatique diffuse), la sensibilité au contexte qui correspond au recueil et au traitement d'informations concernant l'utilisateur et son environnement, l'intelligence, qui caractérise la capacité du système à produire des inférences à partir des informations sur l'utilisateur et son environnement et à initier des actions sur la base de ces inférences, et l'interaction naturelle minimale, c'est-à-dire une interaction qui repose sur des modalités et des répertoires gestuels familiers de l'utilisateur, et qui exige le minimum d'attention de sa part.

L'objectif de nos travaux est de développer une infrastructure logicielle à l'attention des développeurs d'applications, de dispositifs et d'objets communicants pour faciliter leur accès aux informations de contexte. Cette infrastructure devra s'intégrer dans des environnements d'intelligence ambiante, ce qui impose qu'elle supporte l'ouverture en termes de dynamisme et d'hétérogénéité des acteurs. On s'inté-

1. Ubiquitous computing
2. Context-aware Computing
3. Tangible Computing
4. Pervasive Computing
5. Ambient Intelligence

ressera à circonscrire les notions de contexte et d'informations de contexte dans le cadre de l'intelligence ambiante et à déterminer le meilleur moyen d'y accéder ainsi que le moyen de la représenter.

2 Exemples d'utilisation du contexte dans la sphère domestique

Nous introduisons un scénario qui illustre simplement la problèmatique de l'intelligence ambiante que nous étudions et l'utilité des informations de contexte dans ces environnements. Décrit ici informellement, il sera reconsidéré à chaque étape pour montrer comment il est effectivement possible de l'implémenter.

Une personne est tranquillement installée dans sa chambre d'hôtel lorsqu'elle reçoit un appel téléphonique. Le dispositif téléphonique, constatant que l'appel n'est nullement urgent va chercher à savoir s'il doit émettre un signal sonore, une vibration ou dérouter l'appelant vers une messagerie. Afin de préserver au mieux la tranquilité de son utilisateur, il cherchera à connaître son activité (sommeil, discussion avec un tiers, etc.). Pour cela, il va tirer parti des informations que l'environnement est capable de lui fournir : états des autres appareils électriques en fonctionnement (ordinateur, sèche cheveux, téléviseur), présence de tiers dans la pièce, état physique de l'environnement (température, degré hygrométrique, luminosité).

On peut se rendre compte, à partir des exemples donnés, que le support à l'application téléphonique peut se trouver dans de nombreuses chambres d'hôtel. Mais elles n'ont pas été prévue (a) pour communiquer entre elles, (b) pour le type d'application envisagé. On pourrait résumer notre objectif comme proposer une implémentation de (a) faisant en sorte que (b) ne soit jamais nécessaire.

3 Le contexte dans les environnements d'intelligence ambiante

Les informations qui caractérisent l'environnement, ses acteurs et leurs (inter-)actions, sont primordiales dans les environnements d'intelligence ambiante. Nous les appellerons informations de contexte. Il faut les puiser dans l'environnement, les analyser et s'y adapter. L'environnement peut varier autant par les acteurs qui y évoluent ou par les sources d'informations dont il dispose que par les applications qui le peuplent.

3.1 Percevoir l'environnement, adapter pour servir l'utilisateur

Les environnements d'intelligence ambiante sont conçus autour d'un réseau sur lequel des applications, des dispositifs et des capteurs communiquent et qui peut être ouvert sur le monde de l'Internet. Que ce soit dans un environnement public, privé ou professionnel, l'intelligence ambiante préconise la construction de systèmes qui transforment les espaces physiques en systèmes interactifs intelligents, c'est-à-dire des infrastructures capables de tirer le meilleur parti de tous les dispositifs accessibles par le réseau afin d'offrir le maximum de richesse d'interaction pour l'utilisateur combiné avec une simplicité d'utilisation.

Les utilisateurs sont mobiles et souvent impliqués dans plusieurs tâches, utilisant plusieurs applications, ce qui implique des modalités d'interface variées entre l'homme et l'environnement. L'utilisation d'une telle richesse d'interface ne doit pas compliquer les tâches de l'utilisateur ni son utilisation du système mais, bien au contraire, le décharger de certaines tâches et lui laisser disponible le maximum de ses ressources cognitives. De plus, les environnements d'intelligence ambiante sont ouverts et dynamiques. Des dispositifs potentiellement hétérogènes et inconnus peuvent entrer et sortir, notamment ceux portés par les utilisateurs. L'adaptation des comportements applicatifs aux situations changeantes des uti-

lisateurs ne peut pas être planifiée pendant la conception des environnements, mais peut être seulement anticipée pour tous les utilisateurs et pour tous les dispositifs présents dans l'environnement.

Pour permettre aux applications de répondre de la meilleure façon quelle que soit la situation, elles doivent pouvoir apréhender ce qui se passe dans l'environnement. Pour cela, les environnements intelligents sont instrumentés de capteurs basiques ou de "capteurs complexes" pour apréhender les aspects variés du contexte. Les premiers permettent d'appréhender des grandeurs physiques. Ils nous renseigneront sur le lieu, la température, la pression atmosphérique, la pression sur un siège ou dans un lit, l'heure et la date... alors que les seconds réalisent un pré-traitement sur ces grandeurs physiques pour en extraire des informations de plus haut niveau. Ils pourront nous dire s'il fait chaud ou pas, quelle est la position de l'utilisateur sur le siège, quelle est son activité, quel est son contexte social... Par exemple, pour déterminer ce que l'utilisateur fait ou tente de faire à un moment donné et lui proposer des services afin de l'aider au mieux dans son activité, un ou plusieurs services dédiés à l'analyse de l'activité des utilisateurs seront probablement présents dans les environnements d'intelligence ambiante, bien souvent en relation avec un service de localisation et d'indentification des utilisateurs. Des services web ou une application distante peuvent être considérés comme des sources de contexte et fournir des informations de contexte à une application. Par exemple, une application locale peut se servir des informations météorologiques trouvées grace à un service web pour organiser au mieux les loisirs de l'utilisateur.

D'autre part, les applications utilisent les informations de contexte pour présenter ou enrichir l'information et enrichir ou modifier les modalités d'interaction entre l'environnement et l'utilisateur. Une application de communication pourra favoriser la modalité sonore par rapport à la modalité vidéo ou modifier son comportement pour transmettre les messages à l'utiliseur en se référant à ses préférences, sa localisation, ses activités, les autres utilisateurs présents dans la pièce (afin de garantir les conditions de respect de la vie privée de l'utilisateur).

Ceci n'est pas un inventaire exhaustif des adaptations que réalisent les applications en fonction des informations de contexte. Pour résumer, c'est à l'application de s'adapter à l'utilisateur, à ses besoins, à ses préférences et à ses activités courantes, ainsi qu'aux autres utilisateurs et autres applications présentes dans l'environnement. Nous étudierons plus en détails dans le premier chapitre de l'état de l'art différentes façons de considérer le contexte et les informations de contexte.

3.2 Gérer les informations de contexte

Dans un environnement d'intelligence ambiante, les applications doivent pouvoir acquérir et raisonner sur les informations qui caractérisent l'environnement sans se soucier de la provenance des informations, sans avoir de connaissance des réseaux et des différentes manières d'interroger les sources d'information de contexte. Pour ce faire, nous proposons et étudions le principe de l'introduction dans les environnements d'intelligence ambiante d'un intergiciel dédié à la gestion des informations de contexte qui capitalisera ces fonctions pour répondre aux besoins des applications de l'environnement. Premièrement, il assurera la mise en relation des producteurs d'informations de contexte avec les consommateurs et l'établissement d'un flux d'information des premiers vers les deuxièmes. Le but étant qu'un producteur ne soit pas dédié à un seul consommateur et que plusieurs consommateurs puissent interroger le même producteur d'information de contexte. Deuxièmement, il facilitera le développement des dispositifs sensibles au contexte et des sources d'informations de contexte. Il aura un rôle central dans la circulation de ces informations sans pour autant être implémenté sous la forme d'un serveur central de gestion de contexte. Au contraire, on cherchera à doter chaque dispositif et chaque application de fonctionnalités de gestion de leurs propres informations de contexte tout en assurant leur interopérabilité. Les dispositifs sensibles au contexte n'auront à manipuler qu'une seule interface pour interroger l'environnement, et les sources d'informations de contexte n'auront aussi qu'à manipuler une seule interface pour diffuser les informations sur l'environnement.

Les environnements d'intelligence ambiante sont ouverts et dynamiques. Ils accueillent des dispositifs, des services et des applications conçus par des fabricants différents et qui n'ont pas forcément vocation à fonctionner les uns avec les autres. L'infrastructure de gestion d'information de contexte ayant pour but de mettre en relation des applications aux besoins hétérogènes avec des capteurs aux capacités hétérogènes, elle doit garantir la compatibilité entre ces différents acteurs. Basiquement, la compatibilité peut être assurée entre les interfaces programmatiques ou entre les modèles d'information de contexte manipulées par les applications et les capteurs de l'environnement. Cependant, il n'y a pas de raison pour qu'ils aient été développés avec les mêmes interfaces, ni qu'ils utilisent les même modèles pour représenter le monde ou qu'ils utilisent une terminologie commune ou un ensemble partagé de concepts pour décrire les informations qu'ils manipulent. Ils peuvent employer des termes différents pour décrire les informations de contexte, et même s'ils emploient les mêmes termes, ils pourraient attacher une sémantique différente à ces derniers. Le système de gestion d'information de contexte devra assurer l'interopérabilité syntaxique et sémantique entre les dispositifs. Enfin, afin de supporter l'ouverture des environnements d'intelligence ambiante, les relations entre les consommateurs et les producteurs d'informations de contexte devront pouvoir s'établir dynamiquement sans arrêter le système de gestion d'information de contexte et sans aucune intervention d'un opérateur.

Pour résumer, nous avons identifié plusieurs exigences pour les infrastructure de gestion d'information de contexte :

1. Permettre à tous les dispositifs producteurs d'information de contexte de la diffuser.
2. Permettre à tous les dispositifs consommateurs d'information de contexte de trouver les sources appropriées.
3. Permettre l'interopérabilité syntaxique et sémantique entre différents dispositifs.
4. Permettre l'ajout ou le retrait d'un nouveau consommateur ou d'un nouveau producteur dans l'environnement de façon transparente pour l'utilisateur.

Différentes approches ont été suggérées pour implémenter la sensibilité au contexte dans les environnements d'intelligence ambiante. Une approche,"boîte à outils", fournit un cadre pour le développement et l'exécution des applications sensibles au contexte à base de capteurs ainsi qu'un certain nombre de composants réutilisables. Une autre approche consiste à développer une infrastructure ou un intergiciel pour permettre aux applications d'être sensibles au contexte. Celui-ci fournissant des abstractions uniformes pour les opérations classiques, il simplifie le développement et le déploiement des applications et dispositifs sensibles au contexte et des capteurs. Il doit être indépendant du matériel et du système d'exploitation. Enfin, il doit définir un modèle commun de représentation des informations de contexte que tous les dispositifs peuvent utiliser et assurer également que les différents dispositifs partagent une sémantique commune pour la représentation des informations contextuelles.

L'infrastructure que nous proposons s'inscrit dans la deuxième approche. Notre étude propose une modélisation du contexte qui pourra notamment assurer l'interopérabilité syntaxique et sémantique et une infrastructure de gestion d'informations de contexte qui respecte les éxigences citées ci-dessus, c'est-à-dire qu'elle devra prendre en compte les particularités d'ouverture et de dynamisme des environnements d'intelligence ambiante. Nous étudierons dans le premier chapitre, le contexte et sa représentation afin d'affiner notre proposition de modélisation du contexte, puis nous étudierons dans le deuxième chapitre de l'état de l'art sept propositions d'infrastructure pour souligner les points forts et les points faibles des différentes approches.

4 Contributions

Les contributions de ce travail sont de deux ordres :

– Une infrastructure qui permet l'échange des informations de contexte, suffisamment générale pour être utilisée par différentes applications d'intelligence ambiante, et suffisamment spécifique pour couvrir les informations fournies par les dispositifs déjà utilisés comme les services de localisation, suffisamment flexible pour accepter et exploiter dynamiquement l'introduction de nouveaux dispositifs.
– Un modèle de représentation des informations de contexte qui supporte l'interopérabilité (mise en correspondance et requêtes) entre des producteurs et des consommateurs d'informations de contexte. On se propose donc d'abstraire cette notion de contexte afin de spécifier l'adaptation au contexte de manière indépendante d'une application particulière. On proposera pour cela une représentation des informations de contexte indépendante des applications. L'ouverture de l'environnement impliquera l'introduction de descriptions hétérogènes qui devront être réconciliées avant de pouvoir être exploitées.

5 Organisation du mémoire

Ce mémoire est organisé en deux parties. La première partie est une revue de l'état de l'art des différents domaines afin de permettre une bonne compréhension de ces travaux et de notre contribution scientifique. Elle se découpe comme suit :

– Le premier chapitre s'attache à préciser les notions de contexte et d'informations de contexte. On étudiera ces notions du point de vue de l'intelligence ambiante et du point de vue de l'intelligence artificielle.
– Le deuxième chapitre adopte un point de vue technique pour effectuer une revue des infrastuctures logicielles dédiées à la gestion du contexte. On distinguera les approches qui utilisent des ontologies de celles qui n'en utilisent pas.

La seconde partie du mémoire décrit en détail nos travaux et notre contribution scientifique en quatre chapitres.

– Le troisième chapitre détaille notre contribution scientifique au domaine de l'intelligence ambiante. On y décrit l'architecture distribuée pour la gestion des informations que nous proposons, ainsi que le protocole que nous avons établi pour assurer la transmission des flux d'information de contexte des producteurs vers les consommateurs.
– Le quatrième chapitre fournit la modélisation utilisée dans notre infrastructure pour représenter les informations de contexte. Elle est basée sur les technologies du web sémantique. Nous expliquerons comment cette modélisation permet d'exploiter des ressources hétérogènes.
– Le cinquième chapitre présente l'implémentation réalisée de l'infrastructure conceptuelle présentée dans les deux chapitres précedents. On y détaille l'API fournie ainsi que l'implémentation multi-agents que nous avons réalisée. Par la suite, nous présentons les outils utilisés pour manipuler les ontologies et gérer leurs correpondances.
– Avant de conclure le mémoire, nous présentons un cas d'utilisation aussi réel que possible des technologies développées. Nous avons réalisé un démonstrateur comprenant les caractéristiques problématiques d'un environnement intelligent identifiées dans la première partie du mémoire. Nous avons pu vérifier que notre proposition atteignait les objectifs fixés et tester sa robustesse.

Première partie

Contexte et informations de contexte : représentation, utilisation et infrastructures dédiées à leur gestion

Cette partie est une revue de l'état de l'art des différents domaines relatifs à nos travaux. On y détaille ce que l'on entend par contexte et information de contexte, quels sont les moyens utilisés pour les représenter et les infrastructures existantes qui servent à sa gestion. Cette partie doit permettre une bonne compréhension des travaux présentés par la suite et de notre contribution. Elle nous guidera pour établir notre problématique en se questionnant sur la meilleure façon d'acheminer les informations de contexte de leurs producteurs vers leurs consommateurs.

Elle se découpe comme suit :

- Le premier chapitre s'attache à préciser les notions de contexte et d'informations de contexte. On étudiera ces notions du point de vue de l'intelligence ambiante et du point de vue de l'intelligence artificielle.
- Le deuxième chapitre adopte un point de vue technique pour effectuer une revue des infrastuctures logicielles dédiées à la gestion du contexte. Selon une grille d'analyse prédéfinie, nous detaillerons sept infrastructures existantes en analysant leurs apports au domaine de l'intelligence ambiante. Nous distinguerons les approches qui utilisent les technologies du web sémantique pour représenter les informations de contexte de celles qui ne les utilisent pas. Pour cela, nous introduirons et présenterons brièvement le web sémantique. Bien que notre contribution ne se situe pas dans ce domaine, il est utile de le présenter car nous utilisons ces technologies dans nos travaux. Cette distinction dans les approches étudiées, nous permettra d'identifier les apports de cette technologie dans les environnements ouverts et potentiellement hétérogènes. À partir de l'analyse des avantages et des limitations de chacune des infrastructures, nous pourrons proposer nos choix d'architecture.

Chapitre 1

Contexte et informations de contexte

Sommaire

1.1 Définitions et utilisations du contexte en intelligence ambiante 11
1.1.1 Environnements d'intelligence ambiante 12
1.1.2 Définition du contexte en intelligence ambiante 15
1.2 Contexte, intelligence artificielle et représentation de connaissances 19
1.2.1 Validité des informations en contexte 19
1.2.2 Correspondances entre ontologies et contexte 20
1.3 Bilan 20

L'infrastructure que nous proposons, s'appuie sur une modélisation des informations de contexte commune à tous les acteurs des environnements d'intelligence ambiante afin de faciliter l'échange, la transformation et la recherche de ces informations dans le but de répondre aux besoins des applications. Les environnements d'intelligence ambiante doivent être des environnement ouverts, par conséquent, ils accueillent des applications et des dispositifs hétérogènes. Il faut pouvoir gérer des contextes qui sont différents d'une application ou d'un dispositif à un autre. Pour cerner au mieux la notion de contexte et son utilisation, nous allons faire une revue de différents points de vue sur la question.

Les utilisations de la notion de contexte (ou des contextes) dans le domaine informatique sont multiples. Cette notion est utilisée dès les premiers systèmes d'exploitation, où un contexte décrit l'ensemble des informations permettant de passer d'un processus d'exécution à un autre et de gérer les interruptions. Le contexte sauvegardé doit au minimum inclure une portion notable de l'état du processeur (registres généraux, registres d'états, etc.) ainsi que les données nécessaires au système d'exploitation pour gérer ce processus. La notion de contexte est aussi utilisée en linguistique et en traitement automatique des langues où le contexte linguistique désigne tout trait linguistique présent autour d'un mot, d'un morphème ou d'un phonème, et qui a ou non une influence sur lui. Le contexte pragmatique englobe tout ce qui est extérieur au langage et qui, pourtant, fait partie d'une situation d'énonciation et peut s'avérer nécessaire à la compréhension. En intelligence artificielle, la notion de contexte est principalement présente dans deux axes de recherches : la représentation de connaissances et la logique. Enfin, le contexte et les informations de contexte sont utilisés dans le sujet qui nous intéresse ici, c'est-à-dire l'intelligence ambiante, et notamment à travers son utilisation pour adapter l'interaction entre l'homme et la machine. Nous allons concentrer notre effort d'état de l'art sur les deux derniers aspects.

1.1 Définitions et utilisations du contexte en intelligence ambiante

Avant de nous intéresser aux définitions du contexte qui sont employées dans le domaine de l'intelligence ambiante, nous allons faire une revue des différentes utilisations du contexte dans ce domaine.

Cet exercice nous permettra d'analyser quels sont les systèmes clients pour un intergiciel de gestion de l'information de contexte et quels sont ses fournisseurs d'informations, ainsi que les fonctions qu'il doit remplir.

La seconde partie de l'état de l'art consacrée au contexte en intelligence ambiante détaillera les différentes définitions du contexte dans ce domaine selon plusieurs points de vue.

1.1.1 Environnements d'intelligence ambiante

Avant de présenter les utilisations des informations de contexte dans les environnements d'intelligence ambiante, nous allons étudier les problèmes que réserve la mise au point de ces environnements. La présentation des différentes utilisations qui suivra n'aura pas pour but ici de faire un inventaire exhaustif des systèmes sensibles au contexte, ni des infrastructures de gestion de l'information ; mais de rendre compte de la diversité des situations dans les environnements d'intelligence ambiante afin d'affiner les exigences de la solution que nous proposons.

Problématiques des environnements d'intelligence ambiante

Nous avons vu dans l'introduction que l'intelligence ambiante vise à la construction de systèmes interactifs intelligents [57]. L'utilisateur évolue dans un environnement dans lequel il peut accéder à de nombreuses applications locales ou distantes, accessibles à travers un réseau. En effet, que ce soit dans un environnement domestique privé, un environnement de travail ou encore dans un lieu public, on présuppose l'existence d'un réseau sur lequel les dispositifs et les objets communicants se connectent et qui permet la communication et l'adressage. Les dispositifs peuvent ainsi déclarer leurs fonctionnalités, en s'intègrant dans une architecture orientée service [6] [11], [6], [5] et utiliser des technologies d'annuaire et de description de service [81]. Ainsi, ils peuvent se déclarer comme consommateurs d'informations de contexte sans que ces infrastructures ne leur garantissent qu'ils puissent se mettre en relation avec les producteurs d'informations de contexte qui leur sont nécessaires.

Ces systèmes interactifs intelligents sortent du cadre interactif homme-machine classique où l'utilisateur utilise le plus souvent un clavier et une souris comme périphériques d'entrée vers un ordinateur unique muni d'un écran comme périphérique de sortie. On se place dans une interaction appelée post-WIMP [80] où l'utilisateur interragit avec des applications dont il ne connaît pas forcement la ressource sur laquelle elles s'exécutent, et avec lesquelles il interagit en utilisant des objets communicants [52], des écrans, un assistant personnel, un téléphone mobile, un Nabaztag [7], des phycones [50], ou l'interface gestuelle [42]. De plus, on passe de la configuration où les utilisateurs utilisent une application informatique à une configuration avec des utilisateurs mobiles dans l'environnement qui sont impliqués dans plusieurs tâches, ce qui mobilise plusieurs applications qui ne doivent pas s'accaparer toutes les ressources mentales de l'utilisateur [41].

Pour les architectes d'infrastructure, le défi est d'autant plus difficile que ces environnements sont ouverts et dynamiques. Des dispositifs et des applications potentiellement hétérogènes et inconnus peuvent entrer et sortir, notamment ceux portés par les utilisateurs. Il faut que les infrastructures prennent en compte cette hétérogénéité et soient capables d'intégrer ces nouveaux dispositifs pendant l'exécution de façon automatique, sans contraindre l'utilisateur. Quant à elles, les applications adaptent leurs comportements aux situations changeantes des utilisateurs. Les utilisateurs sont inconnus à la conception des infrastructures. Notre travail s'intéresse au cas le plus probable, c'est-à-dire celui où les applications et les dispositifs ne sont pas tous issus d'un même fabricant, ni forcément du fabricant de l'infrastructure de communication. On considère donc que les utilisateurs, les applications et les dispositifs sont inconnus à

6. `http://www.w3.org/TR/ws-arch/`
7. `http://www.nabaztag.com/`

la conception de l'environnement. Cette conception ne peut pas résoudre toutes les adapations possibles pour les applications, mais elle doit offrir un cadre qui facilite leur réalisation.

Ces environnements d'intelligence ambiante offrent la possibilité d'utiliser de nombreuses applications en utilisant des moyens d'interaction variés. Pour garantir aux utilisateurs une grande richesse d'interaction alliée à une simplicité d'utilisation, il faut que l'infrastructure puisse tirer le meilleur parti de tous les dispositifs accessibles par le réseau.

Le contexte, source d'adaptation

Un environnement est intelligent s'il est capable de s'adapter, c'est-à-dire s'il est peuplé d'applications intelligentes capables d'adapter leur comportement en fonction d'un environnement changeant. On appelle applications sensibles au contexte, les applications s'adaptant en fonction des informations de contexte qui sont les informations qui caractérisent l'environnement. Notre problématique est de construire une infrastructure qui donne un accès aux informations de contexte pour des applications intelligentes. Dans un premier temps, nous allons présenter les besoins de ces applications puis les adaptations envisagées.

Instrumentation des environnements sensibles au contexte S'adapter à son environnement et aux besoins (exprimés ou non) des utilisateurs, c'est, pour l'application, savoir ce qui s'y passe et agir en conséquence. Savoir avec quel utilisateur elle interagit, quelle est sa localisation, quelles sont ses préférences d'interactions, dans quelle activité est-il engagé, quelles sont les personnes présentes avec lui dans l'environnement et quelles relations (sociales, hiérarchiques) a-t-il avec ces personnes... Ces informations sont utiles pour adapter l'interaction et plus généralement adapter le comportement de l'application. En plus de ces informations qui concernent l'utilisateur, l'application peut avoir besoin d'autres informations concernant son environnement. Ainsi, une application qui gèrera la température d'un logement aura besoin de connaître la température des différentes pièces, la température extérieure, l'ensoleillement, l'état d'ouverture des différentes fenêtres, la localisation des habitants, leurs préférences et pourquoi pas leurs agendas pour anticiper les besoins en chauffage pour leur plus grand confort.

Les environnements d'intelligence ambiante sont instrumentés de capteurs pour appréhender la variété des informations de contexte. Classiquement, on y trouvera des capteurs basiques. Ce sont des thermomètres, des capteurs de pression installés sur un siège, un lit ou sur le sol. Une horloge ou un agenda nous fournissant l'heure et la date ainsi qu'un capteur de luminosité sont aussi des capteurs qui permettent d'appréhender des grandeurs physiques. Des applications locales peuvent également servir de sources d'informations de contexte. Par exemple, un service de localisation [7] s'appuiera sur des images vidéos pour fournir à d'autres applications la position de l'utilisateur dans l'environnement. Dans le même ordre d'idée, une application chargée d'inférer l'activité [13] d'un utilisateur pourra s'appuyer sur sa localisation, son agenda et ses habitudes. Généralement, ces applications réalisent un pré-traitement sur des grandeurs physiques issues de l'environnement pour en extraire des informations de plus haut niveau. Les applications de l'environnement pourront avoir besoin d'informations ne provenant pas de l'environnement, mais qui seront pourtant utiles à l'adaptation de leurs comportements. Par exemple, une application locale, chargée d'organiser une sortie à la montagne entre un utilisateur et ses amis, et de l'en informer, veut connaître en temps réel les prévisions météorologiques. Pour ce faire, elle interrogera un service web, pour organiser au mieux les loisirs de l'utilisateur, qui lui fournira des informations sur son contexte d'utilisation. Autrement dit, le contexte de cette application contient les informations concernant les prévisions météorologiques.

Les fonctions d'adaptation Pour Dey et al. [31], l'un des objectifs principaux de l'acquisition du contexte est de déterminer par inférence ce que fait ou tente de faire l'utilisateur à un moment donné

pour lui proposer des services et l'aider au mieux dans son activité. Il distingue trois types de fonctions du recueil d'informations contextuelles :

- la présentation de l'information : des applications utilisent les informations contextuelles pour adapter la présentation de l'information ou modifier les possibilités d'interaction en proposant les actions les plus pertinentes à l'utilisateur. Par exemple, présenter un choix d'imprimantes proches à un utilisateur [71] ; montrer à l'utilisateur sa position géographique sur une carte [2].
- l'exécution de services : des applications exécutent automatiquement une ou plusieurs fonctions pour aider l'utilisateur. Par exemple, l'application peut délivrer un message ou une note lorsque l'utilisateur se trouve à un endroit spécifié [8] ; ou enregistrer automatiquement une réunion informelle [15].
- le stockage d'informations : des applications récupèrent ou ajoutent automatiquement de l'information contextuelle aux informations traitées par l'application. Par exemple, le système de Brotherton et al. [15] permet d'accéder, en plus du contenu de la réunion, aux informations relatives au lieu, à la date, à l'heure, aux participants. Le système développé par Pascoe, Ryan et Morse [60] permet aux zoologues de prendre des notes indexées spatialement et temporellement.

Une autre approche, celle de Schmidt, Van de Velde et Kortuem [73], décrit trois autres fonctions pour le contexte :

- l'adaptation au contexte d'usage des modalités d'entrée ou de sortie,
- la réduction pour l'utilisateur du besoin de fournir de manière explicite des informations au système,
- la détermination des moments opportuns et des modes pertinents d'interruption de l'utilisateur.

Les ingrédients d'un environnement intelligent

Nous avons vu précédement que les applications sensibles au contexte qui évoluent dans un environnement d'intelligence ambiante se fournissent en informations de contexte auprès de sources variées afin d'adapter leur comportement. Nous allons développer ici l'analyse de la fonction du contexte et des informations de contexte.

L'objectif des applications est de déterminer le contexte de leur utilisation pour proposer un service adapté à l'environnement et à ses utilisateurs. Pour cette raison, au moins une application dédiée à l'analyse de l'activité des utilisateurs est présente dans les environnements d'intelligence ambiante. Ces applications utilisent souvent un service de localisation et un service d'identification des utilisateurs. [40] propose un service de localisation spécialisé dans la localisation intra-bâtiment et qui permet sa représentation selon différents modèles (un modèle géométrique, un modèle ensembliste, un modèle sémantique et un modèle de graphe). Quant à lui, [7] propose une infrastructure de localisation multi-capteurs qui s'appuie dans sa version basique sur un couplage entre des informations provenant de capteurs RFID [8] et de plusieurs caméras pour alimenter un système de localisation et d'identification. Un système d'analyse de l'activité individuelle et sociale est proposé par [13].

Les adaptations qui viennent le plus rapidement à l'esprit sont celles qui concernent l'interaction entre l'homme et la machine. C'est d'ailleurs un domaine moteur de l'informatique diffuse qui propose de nombreuses solutions pour la réalisation de ces environnements. Les applications utilisent les informations de contexte pour enrichir ou modifier l'interaction proposée. Une application qui doit rendre un message informatif à l'utilisateur choisit la meilleure modalité pour le contacter en fonction de son contexte. Pour délivrer un message personnel, une application utilisera un dispositif personnel si l'utilisation d'un dispositif partagé n'assure pas suffisamment le respect de la confidentialité. Une application préférera délivrer un message sonore si l'utilisateur concentre son attention visuelle sur une tâche

8. `http://fr.wikipedia.org/wiki/RFID`

particulière. Enfin, plus simplement, un dispositif d'affichage modifiera ses réglages en fonction de la luminosité de l'environnement. Pour effectuer une tâche, un utilisateur utilisera plusieurs applications qui utiliseront plusieurs ressources informatiques. Evidemment, dans les environnements d'informatique diffuse plusieurs utilisateurs évolueront en utilisant, chacun, plusieurs applications ; ce qui complexifie l'adaptation des applications.

La présence d'applications variées qui s'exécutent dans un monde multi-utilisateurs amène un problème d'allocation de ressources informatiques. Si deux applications souhaitent utiliser le même dispositif d'entrée pour un utilisateur, ou si plusieurs applications souhaitent utiliser le même dispositif d'affichage ; alors le contexte est une source d'information importante pour répartir l'utilisation de ces différents dispositifs. Il faut dans ce type d'environnement, un service pour orchestrer l'exécution des applications et l'accès aux différentes ressources. Ce service s'appuie également sur les informations de contexte. Par exemple, l'infrastructure proposée par Mathieu Vallée [79] compose automatiquement des services aux fonctionnalités élémentaires pour construire des applications finales pour l'utilisateur. Pour délivrer un message à l'utilisateur, une application de communication cherchera à utiliser le service d'interface de sortie le mieux adapté à la situation de l'utilisateur. Avec un utilisateur mobile, l'application devra sans cesse adapter l'interface de la communication. Des applications utilisent aussi les informations de contexte pour enrichir l'information qu'elles traitent. L'exemple le plus simple est d'utiliser les informations de localisation d'un utilisateur dans une application de guidage pour lui afficher sa position géographique sur un plan et effectuer un zoom sur la partie du plan la plus intéressante. Certaines applications se servent d'un historique d'informations contextuelles pour réaliser un apprentissage, notamment des habitudes de l'utilisateur pour s'adapter au mieux à ses besoins [87].

Dans un environnement changeant où évoluent des applications hétérogènes et qui doit prendre en compte la dimension multi-utilisateurs, les informations de contexte sont utiles à de nombreux égards. Elles fournissent un support pour les applications qui souhaitent s'adapter à l'environnement, mais elles servent aussi de données principales. Ces informations doivent être accessibles pour toutes les applications.

Dans le scénario présenté en introduction, les informations contextuelles sont utiles aux applications afin de déterminer quelles activités occupent l'utilisateur, si l'utilisateur peut être interrompu et selon quelle modalité. Pour cela, la localisation de l'utilisateur sera un bon indice, les objets manipulés, ainsi que son agenda qu'il soit renseigné par l'utilisateur lui même ou qu'il ait été appris par un service de l'infrastructure. Les informations contextuelles, notamment la localisation et l'activité de l'utilisateur, seront utilisées pour choisir le moyen d'interaction avec les utilisateurs locaux ou distants, et pour adapter au mieux cette interaction.

1.1.2 Définition du contexte en intelligence ambiante

Les définitions du contexte nous proviennent de deux types d'études. La première approche est une approche sociologique qui s'intéresse aux environnements intelligents en termes d'usage et du point de vue de l'utilisateur. Nous étudierons en premier lieu les définitions théoriques issues de ce courant. La deuxième approche est une approche technique. Elle concerne les auteurs qui se sont concentrés sur la réalisation de systèmes sensibles au contexte. En plus de la conception de ces systèmes, les auteurs ont, la plupart du temps, apporté une définition du contexte. Ce sont ces définitions que nous détaillerons en distinguant des définitions empiriques des définitions génériques.

Définitions théoriques

L'étude de l'intelligence ambiante par le versant des sciences humaines, notamment la sociologie des usages, montre que le contexte est un élement clef des environnements d'informatique diffuse. Le

contexte est principalement lié à une activité. Il n'y a donc pas un contexte unique mais on peut dénombrer autant de contextes qu'il y a de couples utilisateur(s)-activité [33].

Dans le domaine qui nous intéresse, on trouve des approches qui abordent le contexte du point de vue de ses fonctions. Ainsi, partant de la notion de contexte telle qu'elle est abordée en philosophie du langage, Winograd [85] circonscrit celle-ci à son rôle dans la communication et plus précisément dans l'interprétation des actes communicatifs. Un élément constitue un contexte lorsqu'il joue un rôle dans l'interprétation de quelque chose. Autrement dit, c'est l'usage que l'on fait de l'information qui lui procure ou non le statut de contexte : avec x, un événement ou une caractéristique, x est un élément de contexte si l'interprétation de l'action d'une entité (un individu ou un agent artificiel) dépend de x. De plus, une distinction est faite entre contexte et "setting". Le setting se réfère aux caractéristiques de ce que Dey et al. considèrent comme contexte, c'est-à-dire les personnes, les lieux et les choses qui constituent à un moment donné l'environnement de l'utilisateur. Dans cette approche, le contexte est défini comme l'ensemble des informations caractérisant (partiellement) la situation d'une entité particulière [31]. C'est une notion dont l'acception n'est ni absolue ni universelle, mais relative à une situation [19, 32]. Cette situation peut être une situation physique (comme la localisation spatio-temporelle d'une personne) ou fonctionnelle comme la tâche en cours de réalisation. Dans notre approche, on assimile l'activité d'un l'utilisateur à l'utilisation d'une application par cet utilisateur, par conséquent on considère qu'un utilisateur évoluera dans un contexte différent pour chaque application qu'il utilise. La caractérisation du contexte dans le champ des sciences humaines et sociales montre à quel point ce dernier est d'une complexité telle qu'il est difficile à définir de manière univoque et donc à modéliser. Pour résumer la situation :

- le contexte n'est pas une fin en soi [39]. Il émerge, ou se définit, pour une finalité ou une utilité précise.
- le contexte est un ensemble d'informations. Cet ensemble structuré évolue et peut être partagé entre plusieurs applications. L'utilisation de l'espace d'informations de contexte devrait améliorer la qualité d'utilisation du système et son interaction avec l'utilisateur.

Les articles du domaine de l'intelligence ambiante qui traitent de la communication homme-machine et des usages de l'utilisateur décrivent le plus souvent le contexte comme la tâche que l'utilisateur désire accomplir et l'histoire de l'interaction avec le système dans ce but [32, 85], mais les concepts fondateurs manquent quand il s'agit d'implémenter une infrastructure. Aussi procède-t-on au cas par cas par énumération des éléments contextuels selon une approche extensionnelle. Les propositions qui suivent, visent davantage de généricité.

Définitions empiriques

Dans les premières réalisations techniques, les infrastructures réduisaient généralement le contexte aux informations pouvant être capturées. De plus ces infrastructures proposaient l'adaption d'une ou plusieurs applications pour la réalisation d'une tâche particulière ou d'un ensemble fini de tâches. Elles s'intéressaient à doter une application d'un comportement adaptatif en fonction d'informations contextuelles dont on connait la disponibilité et le moyen pour la recueillir. Il s'agit le plus souvent de la localisation, de l'identité des personnes et du temps. Ainsi, [71] introduit la notion d'informatique sensible au contexte pour désigner un système doté d'un modèle du contexte et capable d'adapter son comportement en fonction du contexte. [71] inclut la localisation et l'identité des personnes et des objets à proximité ainsi que les modifications pouvant intervenir sur ces objets. De ce point de vue, étudier le contexte, c'est répondre aux questions " Où es-tu ? ", " Avec qui es-tu ? ", " De quelles ressources disposes-tu à proximité ? ". Il définit le contexte comme les changements de l'environnement physique, de l'utilisateur et des ressources de calcul.

Un peu plus tard, Brown [16] restreint le contexte aux éléments de l'environnement de l'utilisateur,

puis il introduit l'heure, la saison, la température, l'identité et la localisation de l'utilisateur [17]. Parallèlement aux travaux de Brown, des définitions émergent avec l'introduction explicite du temps et la notion d'état. [70] assimile le contexte à l'environnement, l'identité et la localisation de l'utilisateur ainsi que le temps. Dans ces optiques, le contexte est l'ensemble des informations que l'on peut extraire de l'environnement. Les définitions proposées suggèrent de manipuler une caractérisation du contexte vis-à-vis d'une application, le contexte physique (acquis à partir de données de capteurs en général) y étant une donnée de première importance. Par exemple, dans les définitions du contexte de nombreux auteurs ne font que regrouper une ou plusieurs des informations suivantes : la localisation de l'utilisateur à un instant donné, ses caractéristiques sociales (identité, statut, fonction, groupe-communauté d'appartenance, etc.) et personnelles (intérêts, préferences etc.), son état émotionnel et physiologique à ce même instant, les caractéristiques de son environnement physique (objets, espace géographique, lumière et bruit ambiants, etc.) et social (personnes se trouvant dans son entourage, leur identité, etc.) ou l'heure et la date.

Le contexte est défini en fonction d'un but précis à atteindre, d'applications connues et des adaptations que l'on veut qu'elles réalisent. Ainsi, les données sont élaborées en des caractérisations du contexte plus adaptées à leur utilisation (par exemple, " température élevée " dans le cadre du contrôle de la climatisation). Par rapport à la donnée brute produite par le capteur, celle-ci est appauvrie (c'est-à-dire moins précise) mais plus adaptée à son utilisation. Les différentes définitions du contexte en informatique diffuse sont ainsi très souvent liées à une application ou, au mieux, à un domaine particulier [30, 20]. Cette caractérisation du contexte a un défaut qui est sa dépendance à la tâche que l'on veut accomplir. " Température élevée " n'est pas un terme absolu, il dépend de l'utilisation que l'on veut faire de la pièce : un sauna ou une chambre à coucher ? Il sera donc peu aisé de faire fonctionnner d'autres applications utilisant une caractérisation d'un contexte qui a été développé en fonction d'une application particulière.

Cependant, la préoccupation de modélisation multi-applications du contexte est maintenant reconnue en informatique diffuse. [23] pose le problème de considérer cette notion de contexte comme n'étant pas prédéfinie par les applications. Des auteurs cherchent des définitions servant la réalisation technique d'infrastructures supportant l'ouverture et ils élaborent des définitions visant à plus de généricité.

Définitions génériques

Le premier auteur à proposer une définition générique est Dey, qui caractérise le contexte par rapport à une interaction. Il cherche à adapter l'interaction homme-machine en fonction du contexte. C'est à partir d'une revue de nombreuses définitions proposées dans la littérature du domaine que Dey, Abowd & Salber [31] 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 de l'interaction, l'identité et l'état des (groupes) individus et des objets (informatiques ou non) de l'environnement. Cette définition se veut générique mais restreint sa portée au domaine de l'interaction homme-machine. À partir de cette définition, ces auteurs proposent une modélisation de ce qu'ils entendent par contexte. Le modèle se compose de trois types d'entités et de quatre catégories d'information contextuelle. Les entités sont : les lieux (bureaux, rues, pièces...), les individus ou groupes d'individus et les choses (objets physiques ou numériques). Les catégories regroupent l'identité, la localisation (position dans l'espace, relations spatiales entre les individus ou les objets...), l'état (activité des individus, leur état physiologique, température d'une pièce, niveau de bruit, l'état de fichiers informatiques et de ressources computationnelles...) et le temps (heure, date...). Ce modèle est riche mais il limite les acteurs d'un environnement d'intelligence ambiante dans leurs modélisations du contexte.

Les idées précédentes sont généralisées [83], le contexte est alors vu comme les états des environnements possibles de l'application. En 1998, Pascoe élargi le contexte comme un sous-ensemble d'états

physiques et conceptuels ayant un intérêt pour une entité particulière [59].

Intégrant le travail sur les usages, Rey, pour sa part, affirme dans son travail de thèse [69] qu'un contexte n'existe pas en tant que tel mais se définit pour l'activité d'un utilisateur. Cette activité a lieu dans un réseau de contextes défini par un ensemble de rôles, relations et entités pertinents pour l'activité observée. Chacun des contextes du réseau est composé d'un ensemble de situations définies en fonction des associations rôles-entités et des associations relations-entités.

Cette définition du contexte vaut :

- selon plusieurs points de vue qui appellent la distinction entre le contexte brut, le contexte système, le contexte utilisateur et le contexte net (intersection des contextes système et utilisateur) ;
- selon les deux étapes essentielles du processus de développement d'une telle infrastructure (l'étape de conception-analyse et l'étape d'exécution), qui amènent à distinguer les contextes systèmes captables et captés, les contextes utilisateurs utilisables et utilisés et les contextes nets exploitables et exploités.

Plus généralement, le contexte est défini dans la littérature comme un ensemble d'informations qui portent sur de nombreux éléments ayant potentiellement un intérêt du point de vue de la conception de nouveaux services et bien souvent de l'interaction Homme-Machine. Il s'agit de concevoir des systèmes capables de recueillir et de traiter des informations sur le contexte pour les rendre plus "intelligents". Le contexte est en effet vu comme quelque chose qu'il suffit simplement de capturer via des capteurs matériels ou logiciels (par exemple dans [31]) alors que la caractérisation du contexte est d'une complexité telle qu'il est difficile à définir de manière univoque et à modéliser, le contexte émerge de situations inconnues au stade de la conception des environnements.

Si on veut donner la possibilité aux applications de modifier leurs comportements, et pas seulement l'interaction, en fonction du contexte, on doit élargir cette définition et être capable de lui fournir tout type d'informations. C'est ce que propose Strang [77] qui definit une information de contexte comme toute information qui peut être utilisée pour caractériser l'état d'une entité (une personne, un lieu ou un objet) selon un aspect spécifique. Un aspect est défini comme l'ensemble des états atteignables groupés dans une ou plusieurs dimensions. Un contexte est l'ensemble des informations de contexte pour une tâche précise selon un aspect spécifique.

La majorité des infrastructures de gestion d'informations de contexte reposent sur la définition de Dey : le contexte est l'ensemble des informations caractérisant (partiellement) la situation d'une entité particulière [31]. C'est une notion dont l'acception n'est ni absolue ni universelle, mais relative à une situation [19, 32]. Cette situation peut être une situation physique (comme la localisation spatio-temporelle d'une personne) ou fonctionnelle comme la tâche en cours de réalisation. Comme le contexte n'est pas contraint à contenir toute l'information caractérisant une situation (ce n'est sans doute pas possible), il peut y avoir plusieurs contextes pour une même situation. D'autre part, comme ces contextes peuvent être comparables en fonction des ensembles d'informations qu'ils contiennent, un contexte peut être considéré comme plus ou moins étendu qu'un autre.

Coutaz, Crowley et Rey [25, 69] proposent un modèle de gestion de l'information de contexte structuré en une pyramide à quatre niveaux d'abstraction (capture, transformation, identification et adaptation) tissés de mécanismes pour servir l'historique, la découverte de ressources, la reprise sur panne, la sécurité, la protection de l'espace privé ou la confiance. D'où le nom de ce modèle : pyramide du contexte (voir la figure 1).

Au plus bas niveau d'abstraction, la couche de capture sert d'interface pour accéder aux capteurs physiques : elle en masque la diversité et encapsule les données fournies par les capteurs sous forme d'observables.

La couche transformation s'appuie sur les observables numériques pour produire des observables symboliques. Puis elle s'attache à construire un réseau d'observables en les regroupant pour détecter la présence d'entités, les identifier et les suivre, et ainsi déterminer leurs relations et leurs rôles. Par

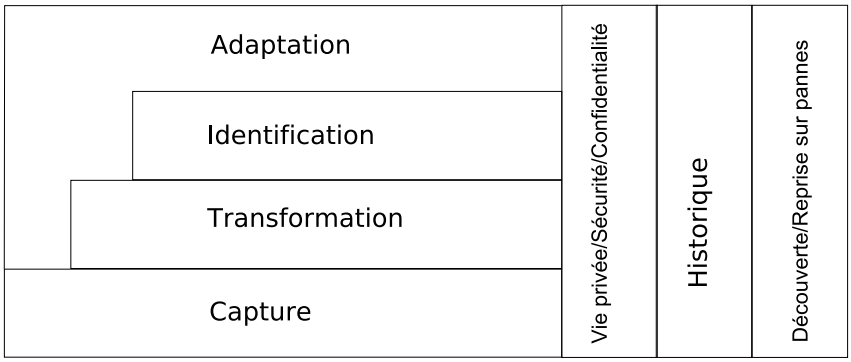


FIGURE 1 – Pyramide de contexte.

exemple, à partir de plusieurs images (observables numériques), le niveau de transformation détermine le nombre d'individus en présence (observable symbolique) et découvre que deux d'entre eux sont à proximité du tableau (relation spatiale) en train de parler (identification d'activité).

La couche identification est chargée de détecter les changements de contexte et de situation et de parcourir les réseaux de contextes et de situations pour identifier le contexte et la situation actuels. Ce niveau est capable de répondre à des requêtes de haut niveau d'abstraction : " quelle est la situation actuelle ? " ou "le bureau B204 de l'INRIA est-il occupé ?".

Au plus haut niveau de la pyramide, la couche adaptation permet de régler toutes les discordances entre l'infrastructure et les applications clientes : adaptation des structures de données, adaptation du protocole de communication, adaptation des formalismes et notamment transformation d'une requête applicative dans les termes attendus par l'infrastructure, au niveau d'abstraction souhaité.

Ces services de base sont complétés de services transversaux : historique, découverte de services et disponibilité, sécurité et protection de la sphère privée, confiance. Ces aspects, qui se retrouvent à chaque niveau d'abstraction, sont répartis et implémentés dans les termes de chaque niveau d'abstraction.

Cette modélisation présente à la fois une porté très générique mais elle reste cependant très proche de l'implémentation.

1.2 Contexte, intelligence artificielle et représentation de connaissances

En intelligence artificielle, la notion de contexte concerne, en général, la représentation de l'information. Elle est exploitée pour rendre compte de deux aspects conjugués : le contexte d'émission de l'information (et plus précisément son contexte de validité [27]) et le fait que plus on développe un raisonnement dans un contexte spécifique, plus on sera efficace [44].

1.2.1 Validité des informations en contexte

John McCarthy [58] a proposé une formalisation logique du contexte fondé sur une "réification" du contexte ainsi que sur un "méta-prédicat" ist, $ist(p, c)$ signifiant que l'assertion p est vraie dans le contexte c.

Les approches du contexte en intelligence artificielle permettent donc de grouper la connaissance en micro-théories [44] et de se placer dans ou hors du contexte de ces théories afin de raisonner. Dans ce cadre (celui de Cyc), le contexte permet de fournir un cadre plus précis pour l'interprétation d'une information. Ce type de fonctionnement peut être utilisé en informatique diffuse afin d'intégrer et d'in-

terpréter les données issues des capteurs. S'adjoindre la théorie associée au capteur permet de tirer parti de l'information qu'il produit sans ambiguïté. Dans cette optique, les données brutes, issues de capteurs, ne sont généralement pas appauvries, mais plus souvent agrégées (et associées à d'autres informations permettant de les interpréter plus précisément).

[45] propose un moyen d'exprimer ce type de contexte pour le web sémantique en associant à chaque triplet une information sur sa provenance ("quad"). Il s'agit ici d'être capable d'interpréter l'information brute provenant de sources diverses (de type fil de nouvelles). D'autres extensions sur le même modèle ont été proposées [53] et sont implémentées dans les gestionnaires RDF modernes.

1.2.2 Correspondances entre ontologies et contexte

Alors que les travaux de McCarthy et Guha consistent à considérer les contextes comme des théories indépendantes portant chacune sur un champ particulier de la connaissance, Fausto Giunchiglia considère plutôt des contextes comme des points de vue concurrents sur une même information. Il explicite les relations entre contextes sous la forme de transformations (" mappings") qui permettent d'importer dans un contexte l'information qui se trouve dans un autre. Une transformation C-OWL est un ensemble de règles de passage ("bridge rules") d'une ontologie source vers une même ontologie cible. Elle a la forme suivante :

1. Un identifiant unique de correspondance (URI)
2. L'URI d'une ontologie source
3. L'URI d'une ontologie cible
4. Un ensemble de règles de passage de l'ontologie source vers l'ontologie cible.

Les règles de passsage s'écrivent avec un élément de l'ontologie source, qui est soit un concept, un rôle ou un individu ; suivi du type de correspondance qui est $\xrightarrow{\sqsupseteq}$, $\xrightarrow{\sqsubseteq}$, $\xrightarrow{\perp}$, $\xrightarrow{\equiv}$ ou $\xrightarrow{*}$ et suivi d'un élément de l'ontologie cible, qui est soit un concept, un rôle ou un individu, avec la restriction que cet élément doit être du même type que l'élément source.

Cette approche peut aussi être utile en informatique diffuse lorsque plusieurs sources d'information produisent des informations comparables. On peut ainsi expliciter la relation qu'il existe entre un modèle de l'environnement qui exprime que la température extérieure est de 25°C et un modèle de l'environnement qui modélise le même environnement en qualifiant la température de "chaude". Ces travaux ont aussi, sous la forme du langage C-OWL une incarnation liée aux langages du web sémantique [78]. Une comparaison des deux approches est donnée dans [75].

Dans [4], les auteurs utilisent une ontologie appelée "background knowledge", et notamment sa structure, pour améliorer le processus de recherche de correspondances entre une ontologie source et une ontologie cible. Cette notion de "background knowledge" peut être rapprochée de la notion d'information de contexte dans le sens où il décrit en détail le domaine de l'ontologie source et de l'ontologie cible. Dans ce sens, cet article nous laisse suposer que plus on dispose d'informations sur le contexte d'émission d'une information, plus on sera capable de l'interpréter précisement. Ceci vient renforcer l'idée qu'une information sur l'environnement prend du sens quand elle est associée aux autres informations sur le même environnement.

1.3 Bilan

Bien que les travaux dans divers domaines se soient penchés sur la notion de contexte, les points de vue sous lesquels cette notion est abordée sont différents. En informatique diffuse, on s'est beaucoup intéressé au contexte d'une application en termes des paramètres physiques qui caractérisent sa

situation [30]. D'un autre côté, en intelligence artificielle, le contexte est plutôt considéré comme les conditions qui font qu'une assertion est valide où ne l'est pas [44]. Si l'on résume les deux approches précédentes on peut considérer que l'informatique diffuse a une approche situationnelle du contexte alors que l'intelligence artificielle a une approche informationnelle. Plus notable, en informatique diffuse, le contexte est très souvent formé autour des besoins d'une application particulière alors qu'en intelligence artificielle, c'est plutôt la source de l'information qui en détermine le contexte. Enfin, en informatique diffuse, l'information provenant des sources a tendance à être plutôt appauvrie pour être directement adaptée à l'application alors que l'intelligence artificielle a plutôt tendance à agréger l'information de contexte pour inférer plus de conséquences (voir figure 2). Bien entendu, rien n'est si tranché et les deux approches sont plutôt complémentaires qu'opposées. Les données brutes dans les deux cas peuvent subir appauvrissement et agrégation.

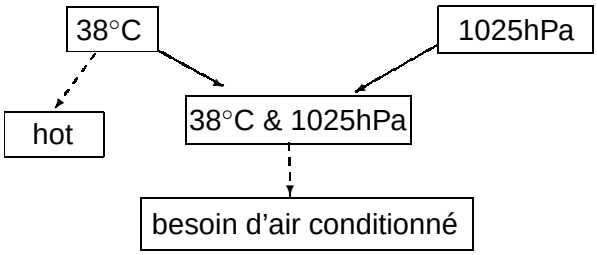


FIGURE 2 – Agrégation (lignes pleines) et appauvrissement du contexte (lignes pointillées). Les données provenant des capteurs doivent être manipulés et transformés pour les besoins de l'application.

D'autre part, cette individualisation du contexte en un ensemble d'éléments de contexte est nécessaire. En effet, une application d'informatique diffuse doit être capable d'entrer et de sortir des contextes : lorsque l'utilisateur entre dans un bâtiment, le contexte associé à la ville dans laquelle il se situe doit passer en arrière plan alors que celui concernant l'intérieur du bâtiment doit devenir principal (pour être oublié lors de la sortie du bâtiment). Ainsi les informations de température, de pression, d'ensoleillement (luminosité), d'humidité et de temps (date et heure) sont des éléments de contexte. Si on souhaite implémenter un service de thermomètre, le contexte se limitera à la température. Si on souhaite implémenter un thermostat ou un service de climatisation (voir la figure 3), on agrégera les informations de température, d'ensoleillement et de temps pour réguler la température. Enfin, le contexte d'un service de station météo regroupera les cinq informations. Il y a aussi d'autres contextes possibles combinant ces informations brutes.

Notre travail vise à fournir à toutes les applications sensibles au contexte les informations qui caractérisent son environnement. La tâche que désire accomplir un utilisateur est liée à l'utilisation d'une application, ainsi nous dirons que le contexte est l'ensemble des informations utiles à l'exécution d'une application. Par extension, les informations captées par un capteur et donc qui caractérisent l'environnement font aussi partie du contexte. Par ailleurs, il est important de noter qu'un contexte n'est défini que par rapport à l'exécution d'un service ou l'utilisation d'une application et qu'il peut exister des liens entre des contextes différents que l'on peut expliciter (voir figure 4). Le contexte d'une application n'est pas un ensemble fini d'informations, on doit pouvoir l'étendre et le modifier. Dans cette perspective, le prochain chapitre sera consacré à l'étude de différentes infrastructures de gestion de contexte proposées dans la littérature.

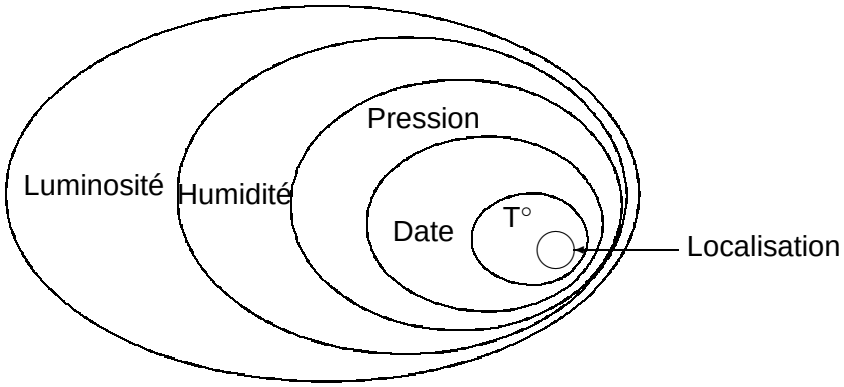


FIGURE 3 – Contexte d'une application de régulation d'air conditionné.

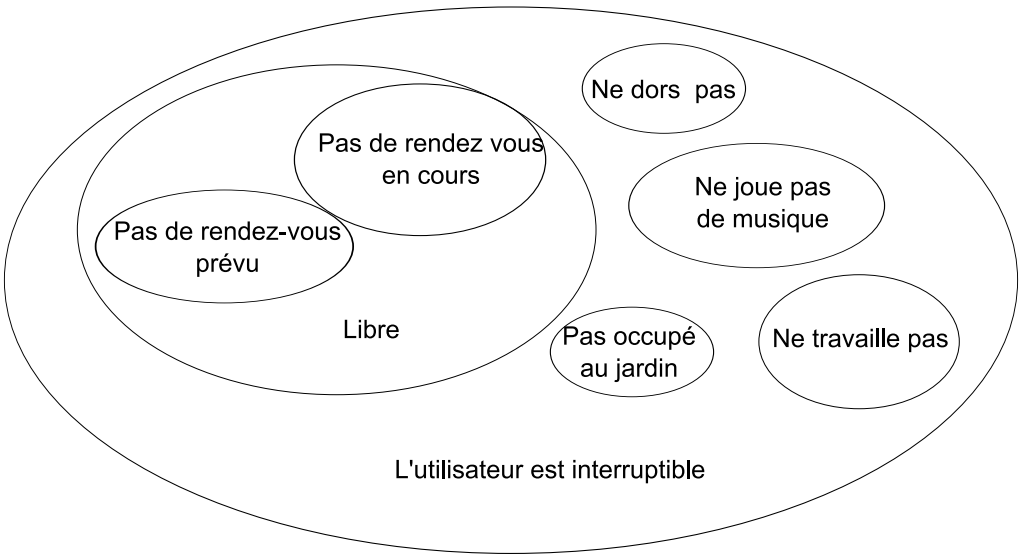


FIGURE 4 – Imbrications de plusieurs contextes

Chapitre 2

Des infrastructures pour la gestion des informations de contexte en intelligence ambiante

Sommaire

2.1 Vocabulaire **24**
2.2 Périmètre de l'étude **24**
2.2.1 Définition 24
2.2.2 Critères d'analyse 26
2.3 Des infrastructures sans ontologie **29**
2.3.1 Schilit 30
2.3.2 Le Context Toolkit 31
2.3.3 Les contexteurs 34
2.4 Des infrastructures utilisant des ontologies **37**
2.4.1 Le web sémantique 37
2.4.2 CoOL 46
2.4.3 Service-Oriented Context-Aware Middleware (SOCAM) 48
2.4.4 Gaia 51
2.4.5 L'architecture CoBrA (Context Broker Architecture) 54
2.5 Bilan **56**

L'importance accordée au contexte en intelligence ambiante et la variété des informations pouvant caractériser une situation ou un environnement ont conduit au développement d'infrastructures supportant la collecte, la diffusion et la gestion des informations de contexte. Ayant déterminé au chapitre précédent notre conception du contexte, il nous faut étudier la meilleur façon de le gérer et pour cela, dans un premier temps, étudier les propositions d'infrastructures de gestions des informations de contexte proposées dans la littérature. Quelle que soit la définition du contexte sous-jacente à ces infrastructures, leur but est de transmettre des informations issues de capteurs ou de dispositifs fournissants des informations sur l'environnement vers des services ou des applications. Dans ce chapitre, nous détaillerons sept infrastructures de gestion d'informations de contexte existantes. Après avoir décrit ce que sont les infrastructures de gestion d'informations de contexte et le vocabulaire associé, on détaillera les différents critères d'évaluation que nous avons établis pour ces infrastructures : prise en compte du dynamisme de l'environnement et de l'hétérogénéité des applications et des dispositifs ainsi que la possibilité d'étendre le modèle de contexte sous-jacent à l'infrastructure. Nous concluerons ce chapitre en étudiant les avantages et les limites de chacun des systèmes présentés pour cerner les enjeux de nos travaux.

2.1 Vocabulaire

Nous utiliserons dans la suite le vocabulaire suivant : L'*environnement* dénotera tout, c'est-à-dire l'environnement physique dans lequel se trouve l'utilisateur, l'ensemble des dispositifs qui y sont présents ainsi que la description complète de la situation. Par *dispositif*, on entendra tout dispositif matériel présent dans l'environnement capable de communiquer. Un *capteur* sera un dispositif sensible au contexte capable de caractériser un aspect de l'environnement physique. Une *application* sera un programme informatique dédié à un utilisateur, alors qu'un *service* sera un programme informatique accessible sur le réseau ayant des interactions uniquement avec d'autres services ou des applications mais il ne sera pas dédié à l'utilisateur. Ce seront par exemple, les logiciels dédiés à l'agrégation d'informations de contexte provenant de plusieurs capteurs, ou un service capable de fournir des informations de contexte de plus haut niveau qu'un capteur, par exemple de déduire l'activité de l'utilisateur à partir de plusieurs sources d'informations de contexte. On s'intéressera ici uniquement à des applications dépendantes du contexte. Les applications que nous considérons plus spécialement sont tout d'abord intéressées à l'appréhension du contexte physique (on suppose que ces applications ont déjà une notion des tâches en cours d'accomplissement), cependant, on verra qu'elles ont toujours la possibilité de recueillir des informations dans un cadre plus global et, en particulier, elles exploiteront le web.

Plus généralement, on parlera de *producteurs d'informations de contexte* pour les dispositifs ou les services capables de fournir une information sur l'environnement et de *consommateurs d'informations de contexte* pour les applications et services qui nécessitent des informations sur l'environnement.

2.2 Périmètre de l'étude

2.2.1 Définition

Notre étude porte sur les systèmes ayant des buts similaires à nos travaux, c'est-à-dire des infrastructures de gestion de l'information de contexte pour les environnements d'intelligence ambiante (voir 2.2.1), d'informatique sensible au contexte ou d'informatique diffuse. Ces environnements sont équipés de capteurs pour appréhender l'environnement et transformer des mesures physiques (température, pression…) en grandeurs du monde numérique. D'autre part, des applications et des services peuplent ces environnements et ont besoin d'informations sur l'environnement soit pour adapter leurs comportements à l'utilisateur et au contexte d'utilisation ; soit pour réaliser des calculs sur ces informations (par exemple un service de localisation) ou pour les enrichir (agrégation d'informations) et les propager dans l'environnement. Ces infrastructures de gestion de l'information de contexte doivent assurer l'échange d'informations entre les capteurs et les services, producteurs d'informations de contexte, et les applications et services, consommateurs d'informations de contexte.

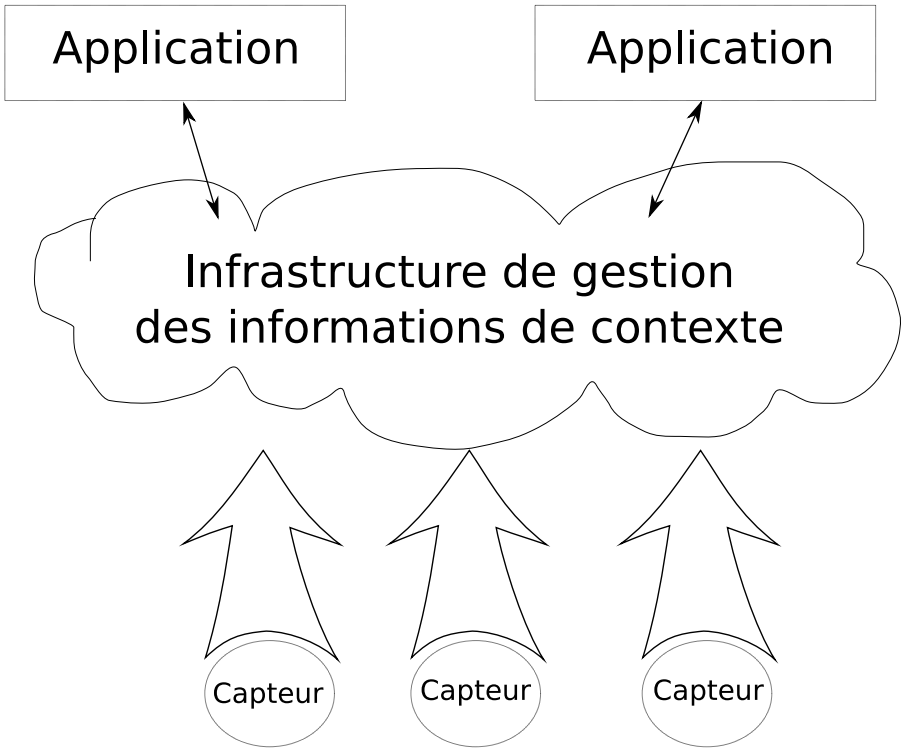


FIGURE 5 – Intergiciel assurant le transfert de données entre les capteurs physiques et les applications utilisant les informations contextuelles.

2.2.2 Critères d'analyse

Pour permettre une meilleure analyse de l'état de l'art et des différentes infrastructures dédiées à la gestion des informations de contexte que nous allons étudier, nous allons dresser une liste de critères qu'il nous semble pertinents d'évaluer pour comparer l'efficacité des systèmes, pour en tirer leurs points forts et leurs points faibles.

Contexte de l'étude

On peut difficilement imaginer un environnement intelligent, qu'il soit privé, semi-public ou public, constitué uniquement de dispositifs fournis par le même fabricant. De façon expérimentale, les dispositifs et les applications sont toujours connus à l'avance, mais sorti de ce cadre, les environnements se peuplent de dispositifs et d'applications hétérogènes qui ne sont utilisés que selon le bon vouloir de l'utilisateur. C'est pourquoi nous chercherons à évaluer la capacité d'une infrastructure à accueillir les dispositifs de tout type et surtout les efforts que doivent réaliser aussi bien l'utilisateur, le concepteur de l'environnement ou celui du dispositif pour l'intégrer dans l'environnement. De la même façon, les environnements d'intelligence ambiante sont ouverts et l'infrastructure doit pouvoir accueillir un nouveau dispositif ou une nouvelle application, qu'il soit producteur ou consommateur d'informations de contexte, pendant l'exécution en exigeant le minimum de perturbations de ce qui est déjà en fonctionnement. Avec les mêmes contraintes, il doit pouvoir profiter de l'environnement ou l'environnement doit pouvoir en profiter. De plus, comme nous avons pu le voir auparavant, il est difficile de définir la notion de "contexte" sorti de tout contexte d'utilisation. Ainsi, les informations de contexte ne sont identifiables qu'en regard d'une situation particulière, c'est-à-dire un utilisateur impliqué dans une tâche nécessitant l'utilisation d'une ou plusieurs applications. C'est pour cette raison que le modèle d'informations de contexte utilisé par une infrastructure ne doit pas être borné. Il doit pouvoir s'étendre pour correspondre à la modélisation de nouvelles situations. Pour finir, le dernier critère concerne l'utilisation des infrastructures par les concepteurs d'environnement intelligents. On cherche à savoir l'effort que concepteurs de dispositifs, capteurs ou applications doivent fournir pour s'intégrer dans l'infrastructure proposée.

Dans un second temps et avec une moindre importance, on évaluera la palette de fonctionnalités offertes par les infrastructures. Nous pensons que les fonctionnalités d'historique, de gestion de la sécurité et des données personnelles, de l'évaluation de la qualité de service et le type d'échange de données supporté (requête, souscription ou autre) sont des fonctionnalités utiles dans les environnements d'intelligence ambiante et nous signalerons si les infrastructures les offrent ; ou si elles permettent leurs implémentations par les développeurs. Le dernier point de l'analyse relève des styles d'architecture de ces infrastructures dans le but d'évaluer leur robustesse.

Prise en compte de l'hétérogénéité des dispositifs

On jugera ici la faculté d'une infrastructure à intégrer des dispositifs, des services et des applications hétérogènes. On évaluera la capacité d'une infrastructure à accueillir des dispositifs, des services ou des applications qui n'ont pas été spécifiquement développés pour cela, et les manipulations à effectuer pour les y intégrer.

Prise en compte du dynamisme de l'environnement

On jugera avec ce critère la faculté d'une infrastructure à accueillir et à exploiter des dispositifs, des services ou des applications pendant l'exécution. La question étant de savoir si cette intégration est automatique, semi-automatique ou manuelle. De plus, on essayera d'évaluer la robustesse de l'infrastructure.

Extensibilité du modèle de contexte

Les infrastructures de gestion d'informations de contexte collectent des informations de l'environnement pour les redistribuer. Elles ont le plus souvent un modèle abstrait pour les représenter afin de faciliter l'échange des informations et assurer une compréhension unifiée de ces informations d'un dispositif à l'autre. Le contexte se définissant par rapport à l'activité d'un utilisateur, il faut que les modèles soutenant les infrastructures acceptent de nouveaux concepts afin d'exprimer de nouveaux contextes et supporter toutes les activités. Nous jugerons avec ce critère si les modèles de contexte sont bornés ou s'ils sont extensibles et de quelle manière cela peut s'opérer.

Facilité d'intégration de l'infrastructure

Les dispositifs et les applications évoluant dans les environnements d'intelligence ambiante sont hétérogènes. Ils n'ont pas été construits pour s'intégrer dans une infrastructure de gestion de l'information de contexte particulière et dans l'idéal ils ne devraient pas être restreints à fonctionner avec une infrastructure de gestion de contexte unique. Nous évaluerons ici la difficulté qui attend un concepteur de dispositif pour intégrer son dispositif dans une infrastructure donnée, et dans quelle mesure ce dispositif calibré pour cette infrastructure pourra s'intégrer dans un autre système de gestion d'information de contexte.

Fonctionnalités offertes

On s'intéressera aux différentes fonctionnalités offertes par les infrastructures : échanges de données, gestion de l'historique des données, gestion de la sécurité des données, qualité de service…

Historique Un service d'historique mémorise les valeurs que prend une information au cours du temps. Cette sauvegarde présente un double intérêt :

- pour l'infrastructure elle-même comme base d'informations pour les reprises sur panne,
- pour les développeurs de systèmes informatiques ambiants, l'historique d'une information permet, par exemple, de calculer des tendances, des moyennes, d'alimenter un système d'apprentissage afin d'affiner l'adaptation d'un comportement.

Un service d'historique offre des fonctionnalités de sauvegarde et de recherche d'information, de suppression et de compilation de l'information de contexte.

Sécurité On s'intéressera ici aux fonctions de sécurité, à la protection de la sphère privée et à la confiance. La sécurité concerne la protection contre les accès non autorisés à un sytème informatique. En informatique conventionnelle, il faut s'assurer que les ressources matérielles et/ou logicielles d'une organisation sont utilisées dans le cadre prévu. En informatique ambiante, elle sert de fondement à la protection de la sphère privée. Pour une infrastructure de contexte, la sécurité consiste à garantir l'intégrité, la confidentialité, la disponibilité et la non répudiation :

- L'intégrité garantit que les informations ne sont pas modifiées ou corrompues.
- La confidentialité assure l'accès à des ressources et informations données aux seules entités autorisées (utilisateurs et/ou services logiciels les représentant). Les techniques par identifiant et mot de passe, de même que le cryptage, relèvent de la confidentialité.
- La disponibilité définit l'accessibilité dans le temps aux ressources et informations.
- La non répudiation consiste à assurer qu'une personne (ou un service logiciel) ne puisse nier avoir participé à un échange.

La protection de l'espace privé est un sujet crucial dans la mise en place d'environnements d'informatique diffuse. Par définition, l'ordinateur personnel disparait de ces environnements au profit d'une multitude de ressources informatiques distribuées dans l'environnement et accessible par un réseau. De ce fait, l'utilisateur est difficilement conscient des échanges de données entre les différents dispositifs, et les infrastructures doivent garantir la protection de ses informations privées.

Échange de données Un autre critère retenu pour l'évaluation des infrastructures de contexte concerne les modes d'échanges de données avec les applications clientes. Les deux modes les plus répandus sont :

- Souscription-Notification : Le client s'abonne à un service particulier auprès de l'infrastructure. Celle-ci envoie les données en fonction du contrat négocié. L'envoi a lieu par exemple, toutes les n millisecondes ou encore chaque fois que la donnée change de valeur. On regroupera ici toutes les situations où c'est l'infrastructure qui décide d'envoyer les informations de contexte aux services et applications concernés. Autrement dit, la souscription n'est pas obligatoire.
- Requêtes-Réponses : L'infrastructure n'envoie des données que sur demande explicite de l'abonné.

À l'évidence, les infrastructures offrant les deux modes d'échanges de données ont l'avantage sur les autres.

Qualité de service Les informations contextuelles sont élaborées à partir d'observables acquis au moyen de capteurs. Or, les capteurs ne sont ni précis, ni stables et tombent en panne. Dans ces conditions, il convient que l'infrastructure auto-évalue le service rendu. Le résultat de ces évaluations est consigné dans des métadonnées qui, attachées aux informations contextuelles, qualifient ces informations. On notera les infrastructures qui proposent une gestion des métadonnées de contexte, ce qui a pour but de fiabiliser son utilisation.

Typologie de l'architecture

Fortement centralisée Un composant central, le serveur, prend en charge l'ensemble des services de l'infrastructure de contexte. Ils sont structurés en deux types de composants : une structure centrale, le serveur, détient les informations contextuelles et une collection de composants indépendants (les clients et les producteurs) opèrent sur la structure centrale. La centralisation signifie que le serveur est le point de passage obligé de toute transaction entre les producteurs d'information (tels les capteurs) et les clients. Cette centralisation n'exclut pas qu'au niveau de la structure physique, le serveur soit réparti sur plusieurs machines.

Faiblement centralisée Un composant central prend en charge la tâche de découverte de services. Il met en relation le client (application) et la source du service (par exemple un composant de capture). Les autres fonctions de l'infrastructure de contexte sont attribuées à d'autres composants spécifiques répertoriés dans le composant central. Contrairement au cas précédent, ici le serveur n'est qu'un annuaire de services. Il n'est plus le point de passage obligatoire de toutes les transactions entre les producteurs et les clients mais seulement des messages d'enregistrements et de découvertes.

Décentralisée Absence de composant central, les services sont alors répartis dans un ensemble de composants. Lors d'une recherche, les messages transitent de composant en composant jusqu'à atteindre leurs destinataires. Une fois le fournisseur du service trouvé, la communication fournisseur-client peut alors se faire directement. Comme pour les infrastructures faiblement centralisées, les systèmes décentralisés utilisent principalement un modèle en flots de données où les entrées des composants clients sont les sorties des composants producteurs.

Synthèse Chacune de ces trois classes d'architecture a ses avantages et inconvénients. Bien que les infrastructures fortement centralisées soient plus facilement administrables, elles nécessitent l'utilisation de machines puissantes, capables d'assurer le rôle du serveur. Elles sont vulnérables et peuvent être entièrement paralysées en cas de panne du serveur. Pour sa part, le modèle faiblement centralisé limite les fonctions du serveur ce qui a pour effet de diminuer la puissance nécessaire de la machine serveur et de faciliter la mise en place de capacité de reconfiguration. Inversement, l'administration de l'infrastructure est plus complexe. Contrairement aux approches fortement et faiblement centralisées, le modèle décentralisé dilue les fonctions de l'infrastructure dans chacun de ses composants. La vulnérabilité due au composant central disparaît en même temps que la nécessité de machines performantes. Une machine peut exécuter autant de composants que sa puissance le lui permet : peu sur un PDA, beaucoup sur une station de travail haute performance. Cependant, l'administration de telles architectures est souvent très complexe.

2.3 Des infrastructures sans ontologie

Nous allons présenter trois infrastructures qui ont pour but d'acheminer les informations caractérisant l'environnement des capteurs vers les applications et qui n'utilisent pas les technologies du web sémantique. La distinction qui est faite entre les infrastructures qui utilisent et celle qui n'utilisent pas ces technologies, nous permet d'introduire ces technologies et leurs spécificités avant de détailler les infrastructures qui les utilisent.

Nous commencerons ici par décrire l'infrastructure de Schilit qui est la première à fournir des mécanismes d'adaptation au contexte. Puis, nous nous intéresserons au Context Toolkit développé par Anin Dey durant son travail de thèse. C'est le premier à baser son infrastructure sur une définition du contexte qui se veut générique, c'est-à-dire qui peut s'appliquer à toutes les situations d'interaction homme-machine. Pour finir cette première série de présentation d'infrastructures de gestion d'informations de contexte, nous présenterons le travail récent de Gaëtan Rey sur les contexteurs. C'est un travail qui s'inspire largement de ce qui a été fait dans le Context Toolkit pour le design de l'infrastructure et dont nous nous sommes inspirés pour élaborer les mécanismes de notre infrastructure.

2.3.1 Schilit

C'est avec [71] que Bill N. Schilit, Norman Adams et Roy Want introduisent la notion d'informatique et d'applications sensibles au contexte [9]. Ce système sensible au contexte doit être capable d'examiner son environnement informatique et de s'adapter à ses changements. Le contexte inclut la localisation et l'identité des personnes et des objets à proximité ainsi que les modifications pouvant intervenir sur ces objets. Étudier le contexte, c'est ici répondre aux questions "Où es-tu ?", " Avec qui es-tu ?", "De quelles ressources disposes-tu à proximité ?". Pour cela, l'architecture s'organise autour du dispositif PARCTAB [72]. C'est un dispositif portable muni d'un écran tactile et trois boutons en interface d'entrée, et d'un écran et d'un haut-parleur en interface de sortie. Il ne possède pas de ressource de calcul, tous les applications s'exécutent sur des serveurs. De plus, il peut communiquer en utilisant un réseau infrarouge, ce qui permet également de le localiser. En plus de la date du jour et de l'heure, l'information de localisation des utilisateurs est la seule information sur l'environnement qui est utilisée.

Cette information de contexte est utilisée de quatre manières différentes :

- Sélection par proximité : Le but est de sélectionner de façon préférentielle les objets les plus proches de l'utilisateur. Ces objets peuvent être des dispositifs d'interfaces, des objets impliqués dans le processus interactif avec le système (incluant les humains) et des lieux d'intérêt.
- Reconfiguration automatique contextuelle : On s'intéresse ici à l'ajout et au retrait de composant du système ou à la modification de la connexion entre deux composants, c'est-à-dire à construire des applications à partir des dispositifs présents dans l'environnement et disponibles (et modifier ces applications en fonction du contexte).
- Informations de contexte et commandes : On s'intéresse ici aux changements que peuvent provoquer les informations de contexte sur une commande. Une même commande envoyée au système dans deux contextes d'utilisation différents peut avoir des effets différents. Par exemple, une application de gestion de fichier affichera les fichiers personnels quand l'utilisateur se trouvera dans son bureau, les fichiers partagés d'un collaborateur quand il sera dans le bureau du collaborateur et le fichier concernant la disponibilité de la pièce dans une salle de réunion.
- Déclenchement d'action en fonction du contexte : À l'aide de fichiers de règles, on veut déclencher des actions en fonction du contexte. Le déclenchement des règles s'effectue à partir des informations concernant l'identité et la localisation des personnes et des objets.

Analyse

Avec ce premier système sensible au contexte, on s'aperçoit qu'on peut réaliser des applications complètes en s'appuyant seulement sur la localisation et l'identification des personnes et des objets. Ces deux informations sont les informations de contexte les plus utilisées dans les environnements de la litérature. Dans ce système, l'existence d'un réseau accessible à tous les dispositifs est indispensable. On peut aussi remarquer que les dispositifs ne sont pas dédiés à une application spécifique. Cependant, cette infrastructure ne propose qu'un nombre limité de types de capteurs et on ne peut pas facilement y intégrer des capteurs d'un genre nouveau. Les informations de contexte qu'elle prend en compte étant limitées, les applications ne sont que partiellement sensibles au contexte.

9. context-aware

2.3.2 Le Context Toolkit

Dey [29] est l'un des premiers chercheurs à avoir généralisé la notion de contexte. Il s'inspire des bibliothèques graphiques d'interacteurs (widgets) pour développer une boîte à outils, le Context Toolkit, capable de fournir aux applications des informations de contexte. Pour Dey, les impératifs d'un système sensible au contexte sont de :

- Séparer l'acquisition du contexte de son utilisation pour que les développeurs d'applications n'aient pas le souci de savoir comment sont obtenues les informations sur le contexte.
- Permettre aux applications d'accéder à l'information de contexte avec le niveau d'abstraction souhaité.
- Fournir un support de communication transparent et distribué entre les applications et les capteurs, ainsi qu'un mécanisme de synchronisation d'horloge entres les différents dispositifs.
- Garantir un accès constant aux informations de contexte.
- Fournir un système pour stocker l'historique des informations de contexte.
- Fournir un mécanisme de découverte des ressources disponibles dans l'environnement.

Le Context Toolkit comprend cinq classes de composants logiciels et deux classes de composants matériels comme on le voit dans la figure 6. Un context widget, ou interacteur de contexte, est un composant autonome qui sert de lien entre un capteur physique et les applications. Son rôle est de communiquer à un (ou plusieurs) agrégateur(s) ou interpréteur(s) les données en provenance de capteurs.

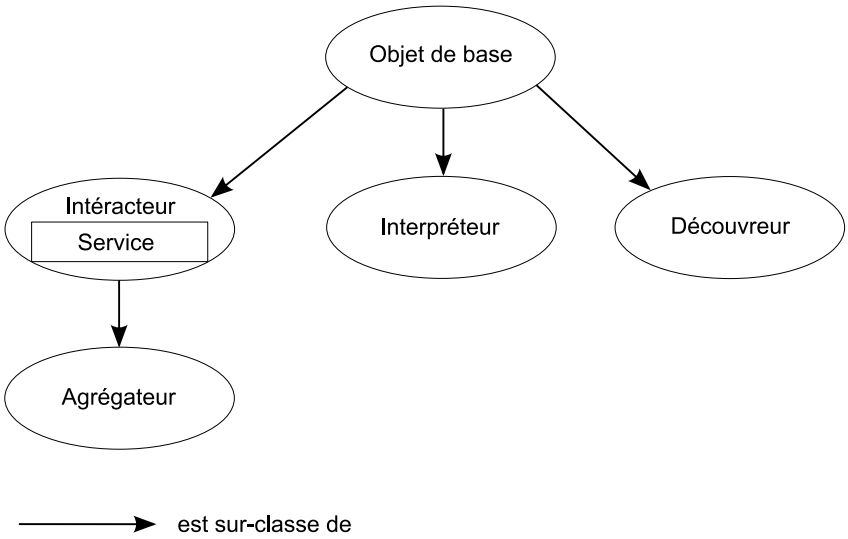


FIGURE 6 – Diagramme d'objet pour les abstractions du Context Toolkit.

Architecture

Les interacteurs de contexte servent d'abstraction aux informations provenant des capteurs. Ils ont trois objectifs :

- Séparer l'application du ou des capteurs qui lui fournissent l'information de contexte. Quelle que soit la technologie utilisée pour calculer la localisation d'un utilisateur (RFID, capteur infrarouge, capteurs de pression, etc.), son utilisation ne doit pas avoir d'impact sur l'application.
- Abstraire les informations de contexte pour répondre au mieux aux besoins de l'application. Par exemple, si une application désire connaître la localisation d'un utilisateur dans une ville, il n'est

pas nécessaire de l'informer quand l'utilisateur traverse la rue ou à chacun de ses mouvements, mais seulement quand il change de rue.
  - Offrir un modèle d'exécution et des composants de base sensibles au contexte réutilisables. Un composant chargé de calculer la localisation d'un utilisateur peut être utilisé par plusieurs applications ou par d'autres composants. Ainsi, un composant qui cherche à savoir le nombre de personnes présentes dans une pièce utilisera le composant de localisation. On pourra aussi construire un composant chargé de déterminer s'il y a une réunion dans cette pièce en supposant qu'une réunion commence quand au moins deux personnes sont présentes dans cette pièce.

  Les interacteurs de contexte sont également capables de constituer un historique qui leur est propre.

Les interpréteurs fournissent aux applications, aux interacteurs et aux agrégateurs une information de contexte au niveau d'abstraction désiré. Dans la plupart des cas, les applications ont besoin d'information de contexte d'un niveau d'abstraction supérieur à ce que peuvent fournir les capteurs. Ces composants utilisent des informations de contexte provenant d'une ou plusieurs sources d'informations de contexte et produisent une nouvelle information de contexte d'un niveau d'abstraction supérieur. Par exemple, un interpréteur de localisation de personne peut fournir comme informations :
  - Personne A dans la salle E988
  - Personne A dans le Bâtiment E du centre Norbert Segard
  - Personne A à France Telecom

Les agrégateurs centralisent l'ensemble des informations de contexte relatives à une entité. On pourra avoir un agrégateur pour chaque utilisateur présent dans l'environnement, il collectera l'ensemble des signaux émis par les widgets concernant cet utilisateur et sera en lien avec les applications. Même si les applications peuvent obtenir les informations de contexte qui les intéressent directement auprès des widgets, on préférera, pour faciliter l'implémentation de l'application et de sa maintenance, relier les widget relatifs à une même entité (personne, objet, lieux) à un agrégateur et cet agrégateur à l'application. L'agrégateur est un widget de haut-niveau.

Les connexions entre les context widgets, les agrégateurs et les applications ont lieu par l'intermédiaire du discoverer. Le discoverer maintient la liste des composants disponibles et de leurs fonctions. Il peut être interrogé par les applications cherchant des composants et/ou des services spécifiques.

Par le biais des actionneurs ("actuators"), les services exécutent des actions pour le compte des applications offrant ainsi des capacités d'action en symétrie avec la perception par le biais des capteurs.

Sur le plan technique, le système de communication entre les context widgets et l'application se fait principalement par souscription et la réaction aux souscriptions s'effectue sous forme de call-back (réaction). Un mode requêtes-réponses est également disponible.

Analyse

Pour résumer, les interacteurs de contexte offrent un accès simple aux capteurs. Les interpréteurs donnent un sens aux données capturées et les agrégateurs font le lien entre les interacteurs et les applications sensibles au contexte pour une entité donnée. Au bilan, le Context Toolkit est une infrastructure simple à comprendre, construite autour d'un modèle faiblement centralisé et offre en plus un service d'actions basé sur les actionneurs. Le service d'historique est disponible au niveau des interacteurs seulement, la découverte se fait par le biais du discoverer, la sécurité et la protection de la sphère privée ne sont pas traitées. Le modèle n'inclut pas de mesures de la qualité (métadonnées). L'application doit donc faire confiance ou pratiquer ces mesures elle-même. Le Context Toolkit fournit une infrastructure pour construire des environnements sensibles au contexte. On peut construire un flux de données commençant

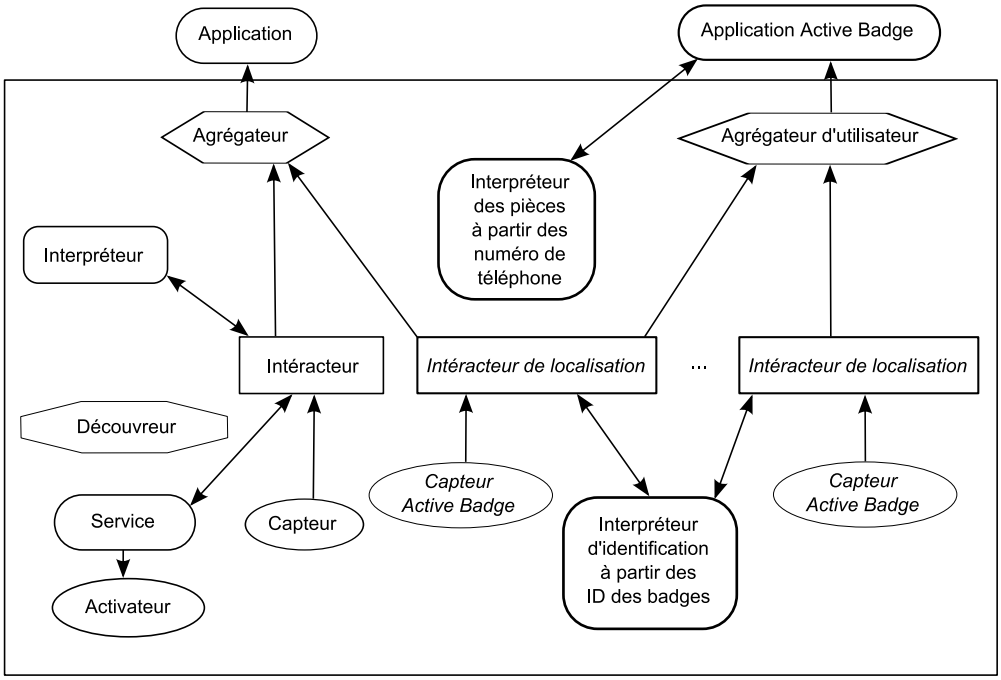


FIGURE 7 – Architecture du Context Toolkit de Dey. Les textes en italique font figures d'exemples.

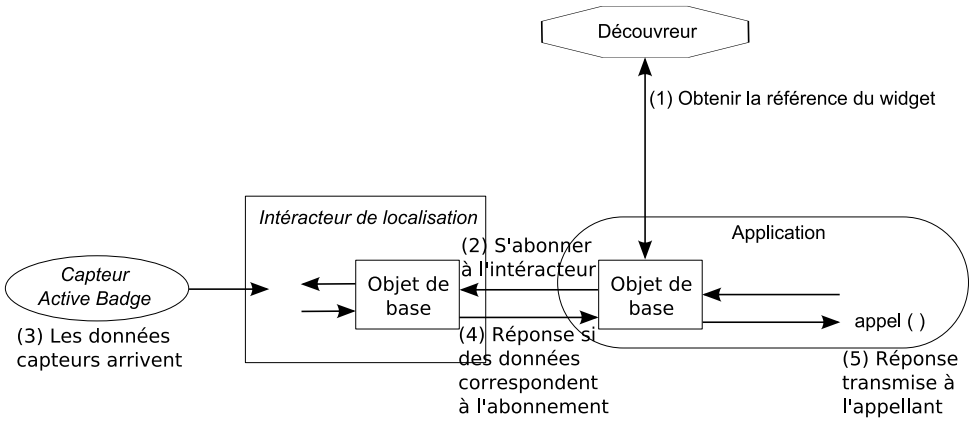


FIGURE 8 – Interaction entre une application et un interacteur de contexte. Les textes en italique font figures d'exemples.

par les données brutes issues des capteurs jusqu'aux données de haut-niveau souhaitées par les applications. Pour réaliser de tels environnements, il faut que les concepteurs sachent dès la conception de l'environnement quelles sont les adaptations souhaitées et quelles sont les sources d'informations de contexte à mettre en place pour cela. En effet, tous les dispositifs que l'on souhaite voir collaborer dans un environnement intelligent doivent dériver du Context Toolkit, ce qui limite l'ouverture d'une telle infrastructure.

2.3.3 Les contexteurs

Rey [69] propose une infrastructure de gestion de contexte décentralisée à base de composants appelés contexteurs. Les contexteurs proposent de structurer les services contextuels en une pyramide à quatre niveaux d'abstraction (voir la figure 1 , 1.1.2, [23]).

Pour résumer, cette architecture abstraite modélise la transformation d'observables de phénomènes physiques (couche capture) provenant directement de l'environnement vers des observables symboliques afin d'identifier les entités, les relations, les rôles et les attributs pertinents (couche transformation), puis d'établir des réseaux de contextes (couche identification), et enfin d'établir l'interface avec les applications au bon niveau d'abstraction (couche adaptation). Ces services de base sont complétés de services transversaux (historique, découverte de services et disponibilité, sécurité et protection de la sphère privée, confiance).

Architecture

Concrètement, un contexteur est une abstraction logicielle qui fournit la valeur d'une variable de contexte de l'environnement. Il comprend trois classes d'élements (voir figure 9) :

- Les entrées de type données ou contrôle. Les premières correspondent aux informations de 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 de contexte, associées à une métadonnée qualifiant la qualité de ces informations, destinées aux autres co-ontexteurs 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 de contexteurs : contexteur élémentaire, contexteur à mémoire, contexteur à seuil, contexteur de traduction, contexteur de fusion et contexteur d'abstraction.

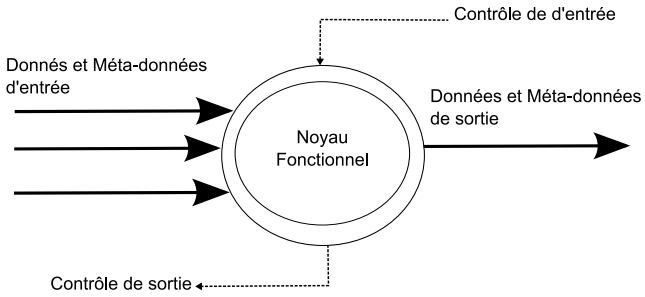


FIGURE 9 – Modèle du contexteur.

Dans son fonctionnement, un contexteur est capable de s'autodécrire et de fournir son nom, sa classe, ses interfaces d'entrée et de sortie ainsi que la ou les fonctions de son corps fonctionnel. Cette propriété assure la possibilité de faire fonctionner les contexteurs en réseau (voir la figure 10). On peut assembler des contexteurs de façon statique afin d'obtenir un service se comportant lui même comme un contexteur. Les contexteurs peuvent aussi être assemblés de façon dynamique en utilisant un protocole de découverte (voir la figure 11). Ce protocole comprend une phase de prospection où le contexteur client recherche les contexteurs dont il a besoin pour fonctionner, et une phase de liaison où les contexteurs s'interconnectent. La communication entre contexteurs tant sur le plan données que sur le plan contrôle se fait par des messages codés en XML. L'échange de données peut se faire selon plusieurs modes : periodiquement,

après chaque calcul de données, à chaque changement de valeur pour une donnée ou par un seul envoi de données.

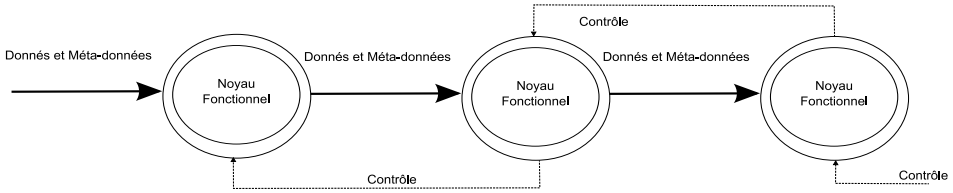


FIGURE 10 – Fonctionnement de trois contexteurs.

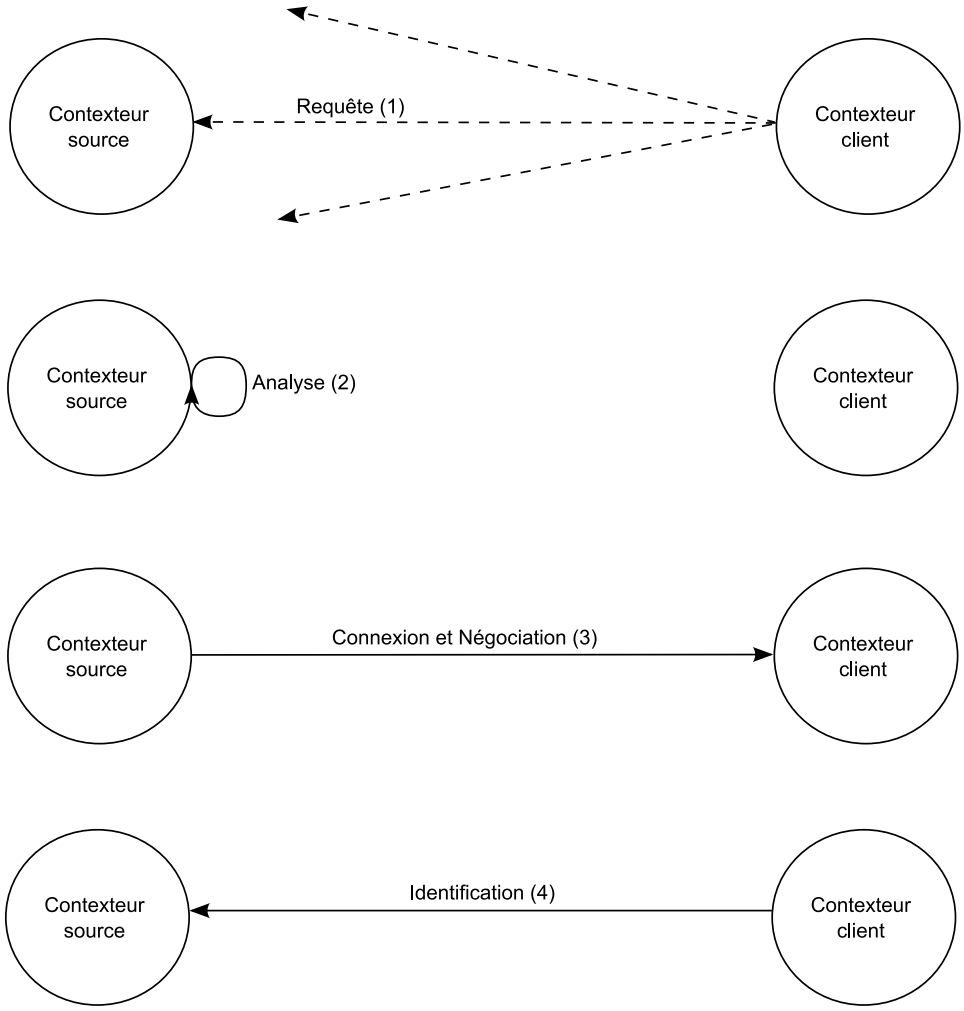


FIGURE 11 – Découverte et connexion à un contexteur source.

Analyse

Les différentes couches du modèle offrent une décomposition du processus qui conduit les informations caractérisant l'environnement, des capteurs vers les applications consommatrices. Cette décompo-

sition permet de construire un environnement intelligent avec un contexteur répondant à la problématique de chaque couche. De ce fait, un contexteur peut être utilisé pour servir plusieurs systèmes adaptatifs, notamment les contexteurs correspondant aux couches basses de la pyramide. Cette infrastructure offre une fonctionnalité de découverte de services mais elle n'est pas soutenue par une modélisation du contexte qui permet à chaque contexteur de décrire ses fonctionnalités dans le cadre d'un environnement ouvert. En effet, aucun mécanisme n'est mis en place pour assurer à un contexteur qu'il pourra s'intégrer dans un système adaptatif qu'il ne connaît pas, même s'il est susceptible d'y servir. L'interopérabilité entre chaque contexteur est possible mais n'est assurée que par l'utilisation du langage XML et donc de schémas de données connus a priori.

2.4 Des infrastructures utilisant des ontologies

Thomas Strang et Claudia Linnhoff-Popien proposent [76] une comparaison des différentes solutions pour modéliser le contexte. Ils comparent les modèles clef-valeur, les modèles à balise de type XML, les modèles graphiques de type UML, les modèles orientés objets, les modèles logiques comme celui de McCarthy (voir 1.2.1) et les modèles basés sur des ontologies en utilisant six critères :

Composition distribuée : les systèmes d'informatique ubiquitaire sont le plus souvent des systèmes décentralisés qui ne disposent pas d'une instance centrale pour la création, le déploiement et la maintenance des données et des services, et en particulier en ce qui concerne les informations de contexte. De ce fait, le modèle de contexte doit supporter la répartition de ses données sur un réseau qui n'est pas figé.

Validation partielle : on souhaite pouvoir valider en partie les connaissances contextuelles, autant du point de vue de leur structure que des instances, auprès d'un modèle de contexte en cours d'utilisation.

Richesse et qualité de l'information : la qualité d'une information délivrée par un capteur varie au fil du temps autant que la richesse des informations fournies par des capteurs différents caractérisant la même entité d'un environnement d'informatique diffuse. Ainsi, l'infrastructure doit fournir au minimum des indicateurs qualifiant la qualité et la richesse des informations sur l'environnement.

Non-complétude et ambiguïté : l'ensemble des informations de contexte disponibles à tout moment est souvent incomplet et/ou ambigu, en particulier si ces informations sont recueillies auprès de réseaux de capteurs. Le modèle d'informations de contexte doit le supporter.

Niveau de formalisme : il est difficile de décrire les informations de contexte et leurs relations de façon précise et traçable. Par exemple, pour accomplir la tâche "Imprimer sur l'imprimante proche de moi", il est nécessaire d'avoir une définition précise des termes utilisés, c'est-à-dire ce que "près de" signifie pour "moi". Il est souhaitable, que toutes les entités de l'environnement engagées dans la tâche partagent la même sémantique pour les données échangées.

Applicabilité aux environnements existants : d'un point de vue de l'implémentation, il est important que le modèle de contexte s'intègre dans les infrastructures d'intelligence ambiante existantes, par exemple dans une infrastructure orientée service.

Les auteurs [76] concluent qu'en regard des six critères ci-dessus, les modèles basés sur les ontologies se révèlent les plus adaptés à la modélisation du contexte dans les environnements d'intelligence ambiante. De plus, d'après eux, les ontologies sont particulièrement adaptées pour décrire les éléments d'information relatifs à notre vie quotidienne dans une structure de données utilisables par des ordinateurs.

Nous allons nous concentrer sur les infrastructures qui gèrent les informations de contexte en utilisant des technologies du web sémantique. Avant de plonger dans la description de plusieurs infrastructures, nous exposerons le web sémantique à travers un court historique, qui reprendra les problématiques abordées par ce domaine, et une description des langages qui en sont issus.

2.4.1 Le web sémantique

Dans ce chapitre, nous présentons les technologies du web sémantique. Bien que notre travail ne se situe pas dans ce domaine, il est utile de le présenter en premier. D'une part, nous utilisons ces technologies dans nos travaux. D'autre part, nous étudions leur utilisation dans l'intelligence ambiante et la représentation des informations de contexte à travers l'état de l'art. Cette section traite de la problématique générale du web sémantique, puis donne plus de détails sur les langages RDF, OWL et SPARQL que nous utilisons. Il s'achève sur une présentation des technologies d'alignement d'ontologies.

Description du web sémantique

L'expression web sémantique, due à Tim Berners-Lee [10] [10] au sein du W3C, fait d'abord référence à la vision du web de demain comme un vaste espace d'échange de ressources entre êtres humains et machines. Il devrait permettre une exploitation qualitativement supérieure, de grands volumes d'informations et de services variés en donnant la capacité aux machines d'appréhender le contenu des ressources qu'elles manipulent et de raisonner sur celles-ci.

Le web actuel est essentiellement syntaxique, dans le sens où la structure des documents (ou ressources au sens large) est bien définie, mais que son contenu reste quasi inaccessible aux traitements machines. Le web sémantique est un enrichissement du web actuel qui a pour ambition de lever cette difficulté. Les ressources du web tout en restant accessibles aux humains seront décrites à l'aide de langages ayant une sémantique définie, ce qui permet aux machines de manipuler les informations et leurs descriptions. Les langages proposés sont à la base de la démarche, ne serait-ce que pour des questions de standardisation, même si l'infrastructure ne se réduit pas à ceux-ci. L'utilisation du web sémantique s'articule aussi autour d'ontologies, représentation formelle d'un domaine. Elles permettent aux logiciels qui communiquent entre eux d'utiliser les mêmes termes avec la même sémantique. Concrètement, le web sémantique est une infrastructure pour permettre l'utilisation de connaissances formalisées en plus du contenu informel actuel du web. Elle doit contribuer à assurer, le plus automatiquement possible, l'interopérabilité et les transformations entre les différents formalismes et les différentes ontologies. Elle doit faciliter la mise en œuvre de calculs et de raisonnements complexes tout en offrant des garanties supérieures sur leur validité.

Pour d'autres points de vue on pourra consulter les premiers livres ou actes de conférence parus [26, 34, 38, 49].

Langages du web sémantique

Dans cette section, nous présentons les deux langages de représentation de connaissances que nous utilisons dans nos travaux : RDF pour représenter des faits et OWL pour représenter des ontologies, ainsi que le langage de requête SPARQL.

RDF Le langage de base du web sémantique est RDF (Resource Description Framework [54]). Il permet d'exprimer des assertions de type sujet-prédicat-objet (appelés triplets et notés $< s, p, o >$). C'est un sous-ensemble du calcul des prédicats. On peut aussi voir un tel ensemble de triplets comme un multi-graphe étiqueté et ainsi faire le lien avec le formalisme des graphes conceptuels. La force de RDF est que les noms des entités (qu'ils soient sujets, prédicats ou objets) sont des URI (les identificateurs du web, que l'on peut voir comme une généralisation des URL : `http://www.w3.org/sw`). Il est ainsi possible dans plusieurs documents RDF de faire référence à une même entité avec certitude : un URI devant dénoter la même chose quel que soit son utilisateur.

OWL Le langage OWL [28], pour sa part, permet de décrire des ontologies ou modèles conceptuels d'un domaine particulier et ainsi de permettre plus facilement l'interprétation des graphes RDF concernant ce domaine. Il permet de définir des classes d'objets et de prédicats ainsi que de déclarer des contraintes pesant sur ces objets. OWL est décliné en trois sous-langages offrant des capacités d'expression croissantes :

- OWL Lite est le sous-langage de OWL le plus simple. Il est destiné aux utilisateurs qui ont besoin d'une hiérarchie de concepts simple.

10. `http://www.w3.org/DesignIssues/Semantic.html`

– OWL DL est plus complexe que OWL Lite et permet une expressivité plus importante. Il garantit la complétude des raisonnements et leur décidabilité.
– OWL Full, la version la plus complexe d'OWL, permet le plus haut niveau d'expressivité. Il offre un haut niveau de description, mais ne garantit pas la décidabilité.

On peut préciser que toute ontologie OWL Lite valide est une ontologie OWL DL valide, et que toute ontologie OWL DL valide est une ontologie OWL Full valide.

`owl:Thing` et `owl:Nothing` sont deux classes OWL prédéfinies. Toute classe OWL est une sous-classe d'owl :Thing et une super-classe d'owl :Nothing. Les classes sont définies avec un élément `owl:Class`. Une classe "Humain" se déclare : `<owl:Class rdf:ID="Humain"/>`. On peut déclarer la classe "Homme" comme sous classe de la classe "Humain" :

```
<owl:Class rdf:ID="Homme">
  <rdfs:subClassOf rdf:resource="#Humain"/>
</owl:Class>
```

On peut également définir une classe comme une collection d'individus.

```
<owl:Class>
  <owl:oneOf rdf:parseType="Collection">
    <owl:Thing rdf:about="#Yoann" />
    <owl:Thing rdf:about="#Aurelien" />
    <owl:Thing rdf:about="#Mathilde" />
  </owl:oneOf>
</owl:Class>
```

Afin d'exprimer des faits au sujet des classes et de leurs instances, OWL permet de définir des propriétés de deux types différents :

– les propriétés d'objet permettent de relier des instances à d'autres instances. C'est une instance de la classe `owl:ObjectProperty`.
– les propriétés de type de donnée permettent de relier des individus à des valeurs de données. C'est une instance de la classe `owl:DatatypeProperty`.

La définition d'une propriété se fait à l'aide d'un axiome qui ne fait qu'affirmer l'existence de cette propriété. `<owl:ObjectProperty rdf:ID="aPourParent"/>` définit la propriété "aPourParent". On peut enrichir la description d'une propriété et préciser son domaine et son image.

```
<owl:ObjectProperty rdf:ID="habite">
    <rdfs:domain rdf:resource="#Humain"/>
    <rdfs:range rdf:resource="#Pays"/>
</owl:ObjectProperty>
```

Ici, la propriété "habite" a pour domaine la classe "Humain" et pour image la classe "Pays". Elle relie des instances de la classe "Humain" à des instances de la classe "Pays".

Une autre façon de décrire une classe OWL est de définir une classe anonyme composée de toutes les instances de `owl:Thing` qui satisfont une ou plusieurs restriction sur des propriétés. Ces contraintes sont de deux types : des contraintes de valeur ou des contraintes de cardinalité. Par exemple, on décrit en dessous l'ensemble des individus ayant deux frères.

```
<owl:Class rdf:about="a2Frere">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#aPourFrere" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">
          2
      </owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Enfin, on peut définir une classe OWL comme une intersection, une union ou un complément d'autres classes OWL. Ceci n'est pas une présentation exhaustive des possibilités offertes par le langage OWL qui comprend des constructeurs et des possibilités déclaratives non présentés dans cette section.

SPARQL Le langage SPARQL [74] est un langage de requêtes pour les documents RDF basé sur la recherche de correspondance entres les triplets de la requête, incluant des variables, et ceux du graphes RDF. SPARQL permet d'exprimer trois types de requêtes :

- Les requêtes SELECT permettent d'extraire du graphe RDF un sous-graphe correspondant à un ensemble de ressources vérifiant les conditions définies dans une clause WHERE par un ou plusieurs triplet.
- Les requêtes CONSTRUCT engendrent un nouveau graphe RDF.
- Les requêtes ASK sont utilisées pour tester si le graphe de requête trouvera un solution dans le graphe RDF interrogé. Ce type de requête ne retournera pas l'ensemble des solutions.

Exemples Description OWL de l'ontologie de la figure 23

```
<rdf:RDF xmlns="http://www.d-ontologie.fr/myPhD/context/home/fig24.owl#"
     xml:base="http://www.d-ontologie.fr/myPhD/context/home/fig24.owl"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
     xmlns:swrl="http://www.w3.org/2003/11/swrl#"
     xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
     xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:owl="http://www.w3.org/2002/07/owl#">
    <owl:Ontology rdf:about=""/>
    <owl:Class rdf:ID="Brightness">
        <rdfs:subClassOf rdf:resource="#PhysicalProperty"/>
    </owl:Class>
    <owl:ObjectProperty rdf:ID="characterizes">
        <owl:inverseOf rdf:resource="#isCharacterizedBy"/>
    </owl:ObjectProperty>
    <owl:Class rdf:ID="ClimStation">
        <rdfs:subClassOf rdf:resource="#TempServ"/>
    </owl:Class>
    <owl:Class rdf:ID="Device"/>
    <owl:Class rdf:ID="Distraction_device">
        <rdfs:subClassOf rdf:resource="#Device"/>
    </owl:Class>
    <owl:Class rdf:ID="Game">
        <rdfs:subClassOf rdf:resource="#Distraction_device"/>
    </owl:Class>
    <owl:Class rdf:ID="HotelRoom">
        <rdfs:subClassOf rdf:resource="#Room"/>
    </owl:Class>
    <HotelRoom rdf:ID="1345"/>
    <owl:Class rdf:ID="IndoorLocation">
        <rdfs:subClassOf rdf:resource="#Location"/>
    </owl:Class>
    <owl:ObjectProperty rdf:ID="informs">
        <rdfs:domain rdf:resource="#Sensor"/>
        <rdfs:range rdf:resource="#PhysicalProperty"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:ID="isCharacterizedBy">
        <rdfs:domain rdf:resource="#Location"/>
        <owl:inverseOf rdf:resource="#characterizes"/>
        <rdfs:range rdf:resource="#PhysicalProperty"/>
    </owl:ObjectProperty>
    <owl:Class rdf:ID="jsd678-mk2">
        <rdfs:subClassOf rdf:resource="#ClimStation"/>
    </owl:Class>
    <jsd678-mk2 rdf:ID="jsd67898200"/>
    <owl:ObjectProperty rdf:ID="locatedIn">
        <rdfs:domain rdf:resource="#Device"/>
        <rdfs:range rdf:resource="#Location"/>
    </owl:ObjectProperty>
    <owl:Class rdf:ID="Location"/>
    <owl:Class rdf:ID="Moisture">
        <rdfs:subClassOf rdf:resource="#PhysicalProperty"/>
    </owl:Class>
    <owl:Class rdf:ID="PhysicalProperty"/>
```

```
    <owl:Class rdf:ID="Room">
        <rdfs:subClassOf rdf:resource="#IndoorLocation"/>
    </owl:Class>
    <owl:Class rdf:ID="Sensor">
        <rdfs:subClassOf rdf:resource="#Device"/>
    </owl:Class>
    <owl:Class rdf:ID="Temperature">
        <rdfs:subClassOf rdf:resource="#PhysicalProperty"/>
    </owl:Class>
    <owl:Class rdf:ID="TempServ">
        <rdfs:subClassOf rdf:resource="#Sensor"/>
    </owl:Class>
    <owl:Class rdf:ID="TVSet">
        <rdfs:subClassOf rdf:resource="#Distraction_device"/>
    </owl:Class>
</rdf:RDF>
```

Description des besoins du dispositif téléphonique de l’exemple

```
pervact = http://pervasive.semanticweb.org/ont/2004/06/action
 pervloc = http://pervasive.semanticweb.org/ont/2004/06/location
 amidom = http://www.owl-ontologies.com/Amigo/AmigoDomains.owl
 xsd = http://www.w3.org/2001/XMLSchema
<owl:Class rdf:ID="PhoneContextDescription">
  <rdfs:subClassOf rdf:resource="ContextDescription"/>
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="&pervact;#Action"/>
    <owl:Class rdf:about="&Loca;#NumberOfPeople"/>
    <owl:Class rdf:about="&amidom;#ON"
  </owl:unionOf>
</owl:Class>
<owl:Class rdf:ID="phc:PhoneContextDescription">
  <rdfs:subClassOf rdf:resource="ContextDescription"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="phc:action"/>
      <owl:allValuesFrom rdf:about="pervact#Activity"/>
    </owl:Restriction>
  </rdf:subClassOf>
  <rdfs:subClassOf>
   <owl:Restriction>
     <owl:onProperty rdf:resource="Loca;#numberOfPeople"/>
     <owl:cardinality rdf:datatype="xsd#nonNegativeInteger">1</owl:cardinality>
   </owl:Restriction>
 </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="phc:activeDevices"/>
      <owl:allValuesFrom>
        <owl:Class>
          <owl:intersectionOf rdf:parseType="Collection">
             <owl:Class rdf:about="amidom:ActivityMarker"/>
             <owl:Restriction>
               <owl:onProperty rdf:resource="amidon:on"/>
               <owl:hasValue>true</owl:hasValue>
             </owl:Restriction>
          </owl:intersectionOf>
        </owl:Class>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdf:subClassOf>
</owl:Class>
```

Description des capacités d'un capteur de météo [11]

```
<rdf:RDF xml:base="http://metadata.net/WildNET/Climate.owl">
 <owl:imports rdf:resource="http://metadata.net/WildNET/Geography.owl"/>
 </owl:Ontology>
 <owl:Class rdf:ID="ClimateSensor"/>
 <owl:Class rdf:ID="ClimateReading"/>
  <owl:DatatypeProperty rdf:ID="hasDate">
   <rdfs:domain rdf:resource="#ClimateReading"/>
   <rdfs:comment  rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    Date/Time the sensor reading was taken
   </rdfs:comment>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="hasClimateSensor">
   <rdfs:range rdf:resource="#ClimateSensor"/>
   <rdfs:domain rdf:resource="#ClimateReading"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="hasRainfall">
   <rdfs:domain rdf:resource="#ClimateReading"/>
   <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    Rainfall, in millimetres
   </rdfs:comment>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="hasName">
   <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    Name of the climate station/sensor
   </rdfs:comment>
   <rdfs:domain rdf:resource="#ClimateSensor"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="hasMaximumTemp">
   <rdfs:domain rdf:resource="#ClimateReading"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="hasStationID">
   <rdfs:domain rdf:resource="#ClimateSensor"/>
  </owl:DatatypeProperty>
  <owl:ObjectProperty rdf:ID="hasLocation">
   <rdfs:range  rdf:resource="http://metadata.net/WildNET/geography.owl#Location"/>
   <rdfs:domain rdf:resource="#ClimateSensor"/>
  </owl:ObjectProperty>
  <owl:DatatypeProperty rdf:ID="hasMinimumTemp">
   <rdfs:domain rdf:resource="#ClimateReading"/>
  </owl:DatatypeProperty>
  <j.0:ClimateReading>
   <j.0:hasRainfall rdf:datatype="http://www.w3.org/2001/XMLSchema#string"/>
  </j.0:ClimateReading>
</rdf:RDF>
```

11. http://metadata.net/WildNET/Climate.owl

Web sémantique et représentation du contexte

Il n'existe pas à proprement parler d'outil de gestion du contexte au sens où nous l'entendons dans le web sémantique (il existe des propositions comme celles de CDF [53], [45] ou C-OWL [12]). Mais il se trouve que les langages développés pour le web sémantique, en particulier RDF et OWL, sont adaptés à la représentation du contexte en informatique diffuse [76] et plus particulièrement à une représentation du contexte capable de supporter les aspects dynamiques. Ceci pour deux raisons :

- Ces langages sont ouverts : ils implémentent l'hypothèse du monde ouvert où il est toujours possible d'intégrer plus d'information que l'on en dispose à un moment donné. Ceci est adapté à un environnement dynamique où une information momentanément indisponible est de nouveau accessible. C'est aussi très adapté, on le verra plus loin, à un monde où l'on ajoute dynamiquement de nouvelles représentations.
- Ces langages sont destinés à fonctionner en réseau. Pour cela ils disposent d'un espace d'adressage global (les URI) qui permettent de fusionner des représentations sans risque de conflits de noms.

Nous utiliserons donc ces langages dans les représentations de contextes que nous définirons par la suite.

Trouver des correspondances entre deux ontologies

Il peut exister plusieurs ontologies pour décrire un même domaine. Deux agents qui utilisent deux ontologies différentes pour décrire le même objet font face à un problème d'hétérogénéité. Face à cette hétérogénéité d'ontologies, il est nécessaire de trouver des moyens pour assurer leur interopérabilité pour que les agents collaborent. Il existe deux techniques pour réconcilier deux ontologies : la fusion et l'alignement. La fusion consiste à intégrer les ontologies pour en produire une nouvelle regroupant les concepts, les relations et les instances des ontologies originales. L'alignement s'intéresse à trouver les entités similaires des ontologies en question. L'alignement d'ontologies consiste à chercher les correspondances entre les concepts, les relations et les individus de deux ontologies. Ces opérations peuvent s'effectuer manuellement ou de façon automatique. Une infrastructure permettant de trouver un alignement pour deux ontologies données est explicitée dans [37].

2.4.2 CoOL

C'est dans [77] qu'apparaît la notion d'ontologie dans une infrastructure de gestion d'information de contexte. Elles sont utilisées afin de décrire des informations de contexte en utilisant un modèle formel pour faire face aux problèmes d'hétérogénéité des dispositifs et atteindre l'interopérabilité des services dans les échanges d'informations de contexte.

Ce travail se focalise en premier lieu sur la définition d'un langage pour décrire des concepts, des attributs et des relations de manière précise et traçable et d'un langage supportant les requêtes. Pour ce faire, ils définissent un langage appelé CoOL (pour Context Ontology Language), basé sur OWL.

Architecture

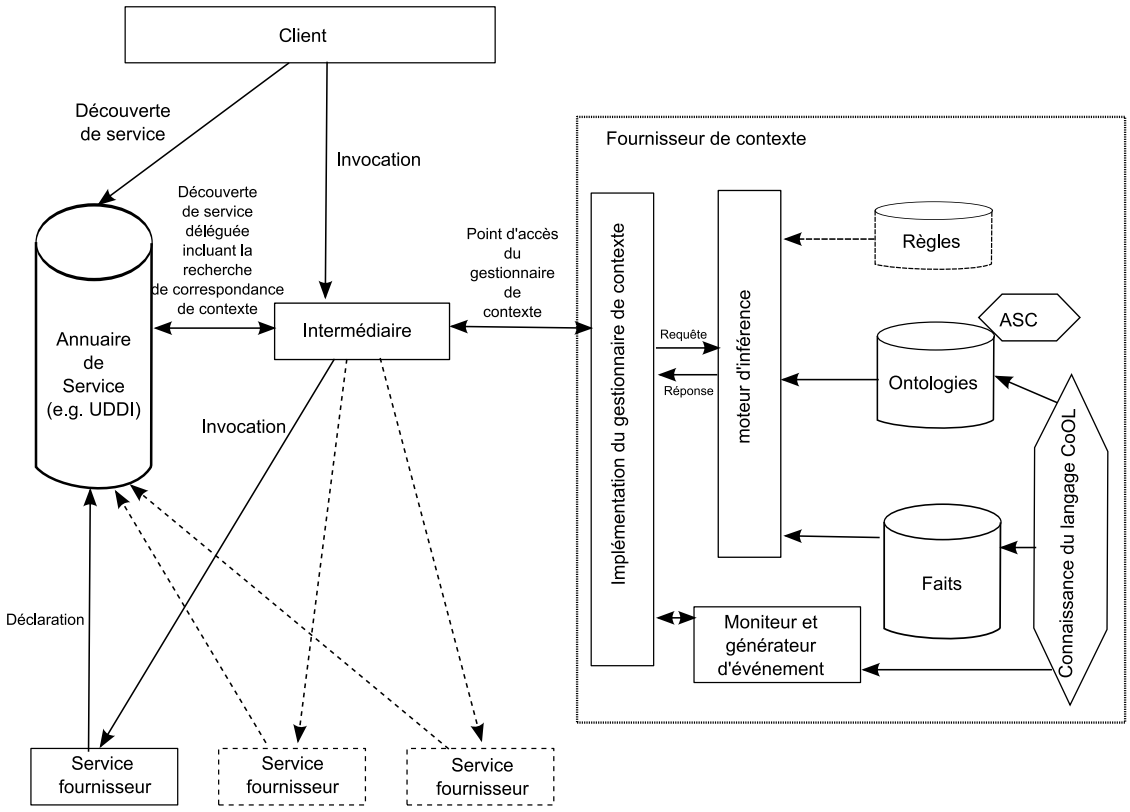


FIGURE 12 – Architecture CoOL.

Une infrastructure (voir la figure 12) est proposée pour utiliser ce langage et permettre aux clients recherchant des informations de contexte d'interroger les services les fournissant. Le composant intermédiaire se comporte comme un client vis-à-vis des producteurs et, parallèlement, se comporte comme un producteur vis-à-vis des clients. Pendant l'exécution, il établit la liaison entre les clients et les producteurs en s'appuyant sur les correspondances entre les paramètres d'appel des services client et les informations des producteurs de contexte.

Ici, une information de contexte est une information utile pour caractériser l'état d'une entité (une personne, un lieu ou un objet) selon un aspect spécifique. Un contexte est l'ensemble des informations de contexte pour une tâche précise selon un aspect spécifique. Cette définition n'expose pas le contexte

comme un ensemble borné, mais comme un ensemble d'informations que l'on peut étendre en fonction des situations que l'on souhaite caractériser.

Ontologies

Le cœur de cette infrastructure est le modèle de contexte ASC ("Aspect Scale Context" que l'on traduit par Aspect, Dimension et Contexte) défini par le langage CoOL. Un aspect agrège une ou plusieurs dimensions qui aggrègent un ou plusieurs contextes. Si l'on s'intéresse à l'aspect température, les dimensions qu'il pourra contenir seront par exemple les degrés celsius et la qualification de très chaud à très froid. Respectivement à ces dimensions, les informations de contexte seront 25°C et chaud. On remarque que la description d'une dimension permet à l'infrastructure de vérifier la validité d'une information dans cette dimension. Pour vérifier la validité d'une dimension dans un aspect, l'infrastructure exige qu'il existe une règle de transformation, codée dans le vocabulaire du modèle, d'une dimension vers une autre dimension. Le vocabulaire propose également des constructeurs qui décrivent les dépendances entre deux aspects disjoints mais relatifs comme l'aspect distance et l'aspect vitesse. D'un point de vue description de service, le vocabulaire propose d'enrichir la description sémantique d'un service par l'attribut *Context Binding* qui permet d'expliciter les liens (typiquement les données d'entrées et de sortie d'un service) entre les services sensibles au contexte et l'attribut *Context Obligation* afin de décrire le contexte d'usage d'un service (par exemple sa portée géographique).

L'architecture utilise un raisonneur qui, en se basant sur des règles préétablies, valide la consistance des ontologies et permet la réconciliation de plusieurs ontologies développées indépendamment. Ce processus vise à déterminer quelles sont les informations jugées utiles à l'adaptation et surtout pour quel aspect.

Analyse

L'architecture CoOL se rapproche d'une architecture orientée service avec un composant de liaison spécialisé pour le contexte. C'est une architecture complexe à mettre en place car elle nécessite la mise en place de deux infrastructures côte à côte (l'infrastructure de service et l'infrastructure de contexte). Par contre, elle permet à des dispositifs hétérogènes de fonctionner ensemble même s'ils ne partagent pas la même vision du monde et de leur environnement. L'utilisation des ontologies permet une interconnexion et un partage des connaissances dans l'environnement. Les correspondances entre les ontologies se faisant sur la base de règles qu'il faut renseigner au préalable, cette architecture ne permet pas l'introduction de dispositifs inconnus pendant l'exécution.

D'autre part, cette approche propose une modélisation du contexte proche de notre analyse (voir la section 1.3). La définition propose un modèle de représentation du contexte ouvert, mais son implémentation oblige tout de même les concepteurs de services à se plier à l'utilisation d'un vocabulaire contraint. L'utilisation de ce vocabulaire pour la description d'informations de contexte ainsi que des besoins et des capacités des dispositifs et services est obligatoire pour les développeurs souhaitant s'intégrer à cette infrastructure car cette dernière se base sur ce vocabulaire pour établir la validité des informations de contexte collectées dans l'environnement.

2.4.3 Service-Oriented Context-Aware Middleware (SOCAM)

SOCAM [43] est une architecture orientée service qui intègre la gestion des informations de contexte. Cette architecture a pour but de faciliter le développement de services sensibles au contexte en partant du constat que cette propriété est essentielle pour les applications informatiques de demain, notamment, les applications mobiles.

Architecture

L'intergiciel sensible au contexte orienté service (voir figure 13) se compose de fournisseurs d'informations de contexte, d'un interpréteur de contexte, de services sensibles au contexte et d'un service de localisation de services.

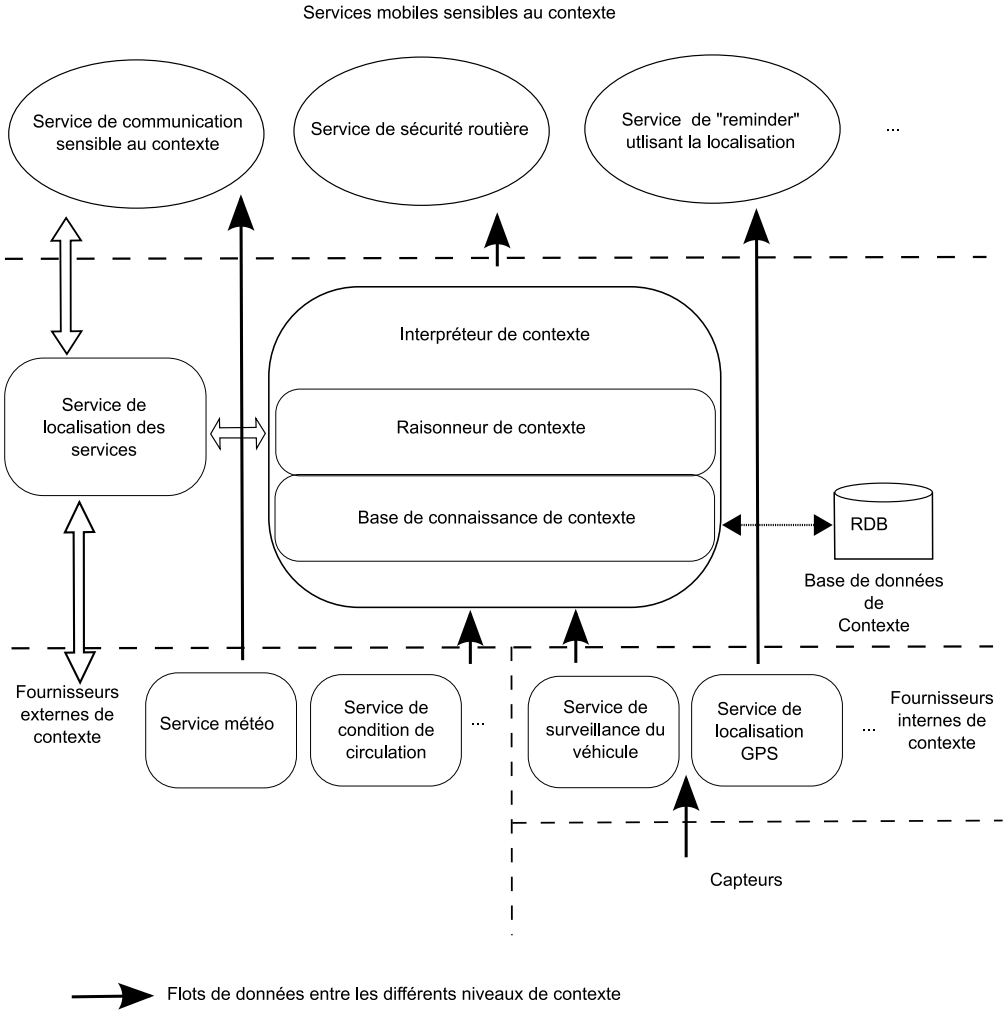


FIGURE 13 – Architecture SOCAM.

Les fournisseurs d'informations de contexte ont pour rôle d'abstraire les informations en provenance de capteurs placés dans l'environnement, d'applications ou de services web. Ils transforment cette information dans une représentation dans le langage OWL afin que tous les composants de l'infrastructure puissent la partager et la réutiliser.

L'interpréteur de contexte est composé d'un moteur de raisonnement sur le contexte et d'une base de connaissance d'informations de contexte. Le moteur d'inférence déduit des contextes, résout les conflits et maintient la consistance de la base de connaissance. Les autres composants de l'infrastructure peuvent interroger la base de connaissance, ajouter, modifier ou enlever des connaissances.

Les services sensibles au contexte s'adaptent au contexte courant en utilisant des informations de différents niveaux.

Le service de localisation permet aux utilisateurs et aux applications de détecter la présence et de localiser les fournisseurs d'informations de contexte et l'interpréteur de contexte.

Ontologies

Cette infrastructure utilise des ontologies OWL qui définissent un vocabulaire de description du contexte autour des concepts agent informatique, localisation, personne et activité. Ce vocabulaire est raffiné dans des ontologies de domaine. Ces ontologies de domaines sont utilisées pour coder les informations de contexte en triplets RDF.

Les auteurs considèrent que l'avantage principal de ce modèle de contexte est de permettre aux utilisateurs, aux dispositifs et aux services de partager une vision commune de la structure des informations de contexte permettant ainsi une interopérabilité sémantique entre les parties.

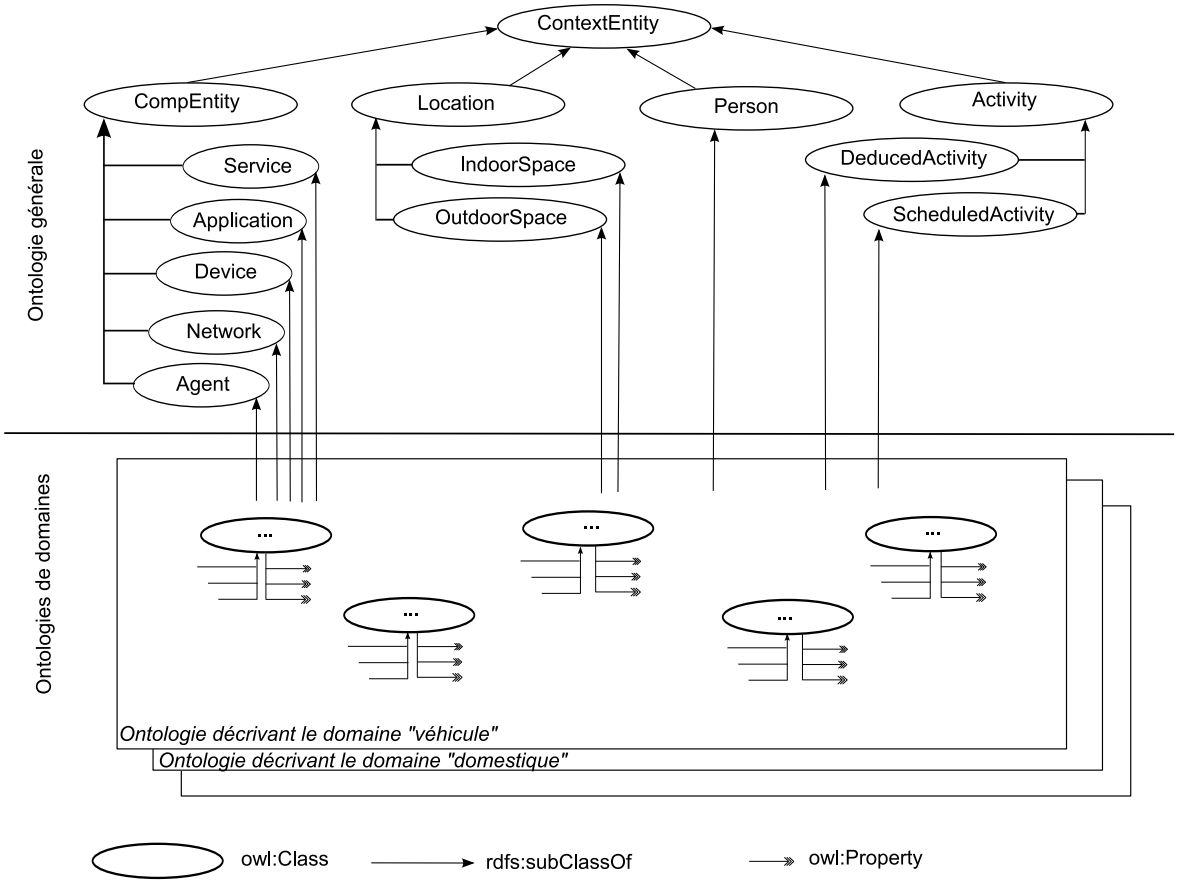


FIGURE 14 – Les ontologies SOCAM.

Les concepteurs ont conçu une ontologie de haut niveau et des ontologies spécifiques comme le décrit la figure 14. L'ontologie de haut niveau code les connaissances générales concernant le contexte, principalement ce qui concerne le monde physique. Les ontologies de bas niveau sont plus spécifiques à un domaine : véhicule, domicile, travail. Elles précisent des concepts et des propriétés définis dans l'ontologie de haut niveau. De plus les ontologies de bas niveau sont interchangeables, c'est-à-dire que le système en utilise une ou l'autre en fonction de la situation.

L'utilisation d'ontologies pour modéliser le contexte offre à l'infrastructure SOCAM un modèle formel, ce qui facilite le raisonnement sur les connaissances de contexte et les rend interprétables par une machine. D'autant plus que les auteurs proposent d'enrichir la description du contexte à l'aide de méta-informations.

Le contexte est catégorisé en contexte direct et contexte indirect. Le contexte direct provient directement des producteurs d'informations de contexte, qu'ils soient internes comme un capteur ou un système de localisation ou externes comme un service web météo. Le contexte direct se décompose en contexte perçu qui s'oppose au contexte défini, par exemple, par l'utilisateur. Le contexte indirect est obtenu à partir du contexte direct et d'un processus de raisonnement (l'agrégation d'information de contexte par exemple). De plus, pour optimiser les capacités de raisonnement du système, une propriété est introduite dans le modèle pour décrire explicitement les dépendances entre les informations de contexte.

L'infrastructure offre deux autres fonctions pour la gestion des informations de contexte. Une ontologie est proposée pour modéliser la qualité d'une information de contexte notamment en termes de précision, de fraîcheur et de certitude avec des métriques associées. Enfin, cette infrastructure est capable de raisonnement sur les informations de contexte soit pour enrichir la description d'une situation soit pour détecter ou résoudre les incohérences dans la description d'un contexte.

Exemple de raisonnement réalisable dans cette infrastructure :

$$Location(Jonh,\ MasterBedRoom) \wedge\ Posture(John,\ LiedDown)$$
$$\wedge\, Status(Door,\ Close)\ \wedge\ (\neg Status(Curtain,\ Open))$$
$$\Rightarrow\ Status(John,\ Sleeping)$$

Analyse

SOCAM [43] est une infrastructure centralisée autour d'un composant appelé interpréteur de contexte. Elle supporte de nombreuses fonctionnalités dédiées à la gestion des informations de contexte. Notamment, elle propose un service qui permet d'introduire de nouveaux services ou dispositifs pendant l'exécution, mais pour s'intégrer ils doivent connaître les ontologies utilisées dans cette infrastructure. Bien que le modèle de contexte et les ontologies proposés soient riches et particulierement bien adapatés à l'informatique pervasive, leur utilisation limite la notion de contexte à ce que le modèle définit car il ne peut pas être étendu.

Au bilan, cette infrastructure peut s'avérer efficace pour la construction d'un environnement ou d'une application dont on connait parfaitement les besoins en termes d'information de contexte et qui ne risque pas d'évoluer dans le temps. Mais son inaptitude à accueillir des dispositifs inconnus à chaud et à étendre son modèle de contexte, rend cette infrastructure inadaptée à la construction d'environnement d'intelligence ambiante telle que nous l'envisageons ici.

2.4.4 Gaia

Gaia [12] est un projet d'envergure pour qui les espaces physiques deviennent des systèmes interactifs appelés "Active Spaces". Ranganathan et Campel [67] proposent des espaces aux fonctionnalités comparables à celles d'un système d'exploitation : gestion des événements, gestion des signaux, un système de fichiers, gestion de la sécurité, gestion des processus, etc. À ces fonctionnalités, Gaia ajoute la prise en compte du contexte, l'adaptabilité en fonction de la localisation, l'informatique mobile et la prise en compte d'actionneurs. Ce système d'exploitation (voir la figure 15) pour applications d'informatique diffuse est conçu pour que la construction des applications soit générique sans avoir à faire d'hypothèses sur la configuration et l'équipement d'un espace. Ainsi le déploiement de ces applications doit pouvoir se faire dans des espaces avec des configurations différentes, en utilisant les ressources disponibles.

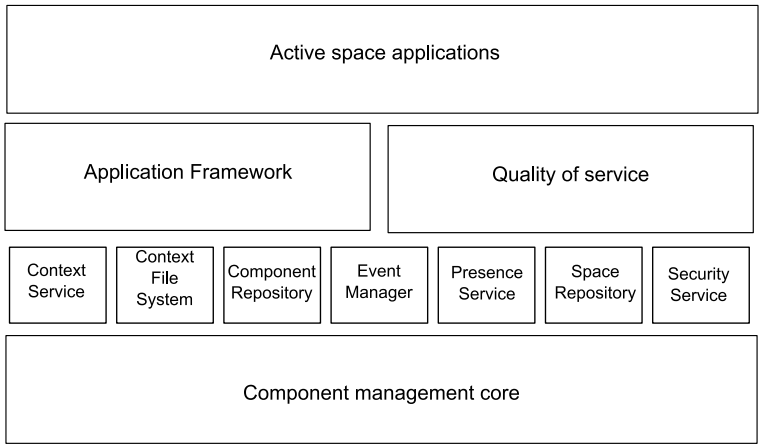


FIGURE 15 – Architecture Gaia.

Modèle de contexte et ontologies

Le modèle de contexte de l'infrastrucutre Gaia décrit les propriétés, la structure des informations de contexte et les opérations que l'on peut réaliser sur le contexte. Le contexte est représenté avec des prédicats de logique du première ordre. Le nom du prédicat représente, par convention, le type de contexte représenté (par exemple : localisation, température, utilisateur). Des opérateurs comme $=$ ou $<$ peuvent être utilisés comme arguments des prédicats. On peut utiliser des opérateurs logiques sur des prédicats tels que la conjonction, la disjonction ou la négation, et des opérations de quantification universelle et existentielle des variables ainsi que l'opérateur de déduction qui peut servir à écrire des règles. Enfin, le modèle permet d'utiliser des fonctions à la place des arguments des prédicats.

Ci-dessous, quelques exemples de prédicat de contexte :

- $Temperature(room3231, " = ", 98F)$
- $Sister(Serena, Venus)$
- $PrinterStatus(srgalw1printerqueue, is, empty)$
- $Time(Grenoble, " < ", 20 : 0015/06/2008)$
- $Location(chris, entering, room3231) \wedge Activity(Room3211, Meeting)$
- NOT $Location(Manuel, In, Romm3231)$
- $\forall_{People} x Location(x, In, currentRoom())$

12. `http://gaia.cs.uiuc.edu`

– $Sound(CurrentRoom(), " > ", 40dB) \wedge Lighting(CurrentRoom(), Stroboscopic) \wedge$
$(CurrentRoom(), " > ", 6)$
$\Rightarrow SocialActivity(CurrentRoom(), Party)$

Les valeurs des arguments d'un prédicat sont contraints en fonction du type de contexte représenté. Les ontologies sont utilisées pour définir et valider l'utilisation des prédicats. Chaque type de contexte est une classe de l'ontologie, détaillée dans [68], qui décrit les arguments de chaque prédicat. La plupart des prédicats de contexte s'écrivent sous la forme $TypedeContexte(< Sujet >, < Verbe >, < Objet >)$. Par exemple, l'ontologie décrit que le prédicat $Temperature$ doit avoit un sujet qui appartient à l'ensemble des localisations, un verbe qui est un opérateur booléen et une valeur qui doit être au format Centigrade ou Fahrenheit. D'autre part, ce modèle de contexte ne précise pas de restriction sur les types de valeurs des différents arguments des prédicats de contexte qui peuvent aussi être des structures complexes. Ainsi, une localisation peut se représenter sous la forme d'une chaîne de caractères codant pour la pièce dans laquelle se trouve l'utilisateur ou un triplet de coordonnées tridimensionnelles. L'utilisation des ontologies assure que la sémantique des arguments soit connue des différents dispositifs. De plus, en supposant qu'il existe des correspondances entre l'ontologie qui définit les prédicats et d'autres ontologies, on peut faire interopérer des environnements utilisant des ontologies différentes.

Architecture

La gestion des informations de contexte est une des fonctionnalités offertes par l'intergiciel du projet Gaia. Cette brique logicielle doit fournir les informations sur l'environnement provenant de capteurs aux applications pour qu'elles adaptent leurs comportements. Pour cela, les auteurs ont conçu un système de gestion de contexte, qui s'inspire lui aussi du context Toolkit de Dey (voir 2.3.2), basé sur plusieurs modules spécialisés (voir la figure 15).

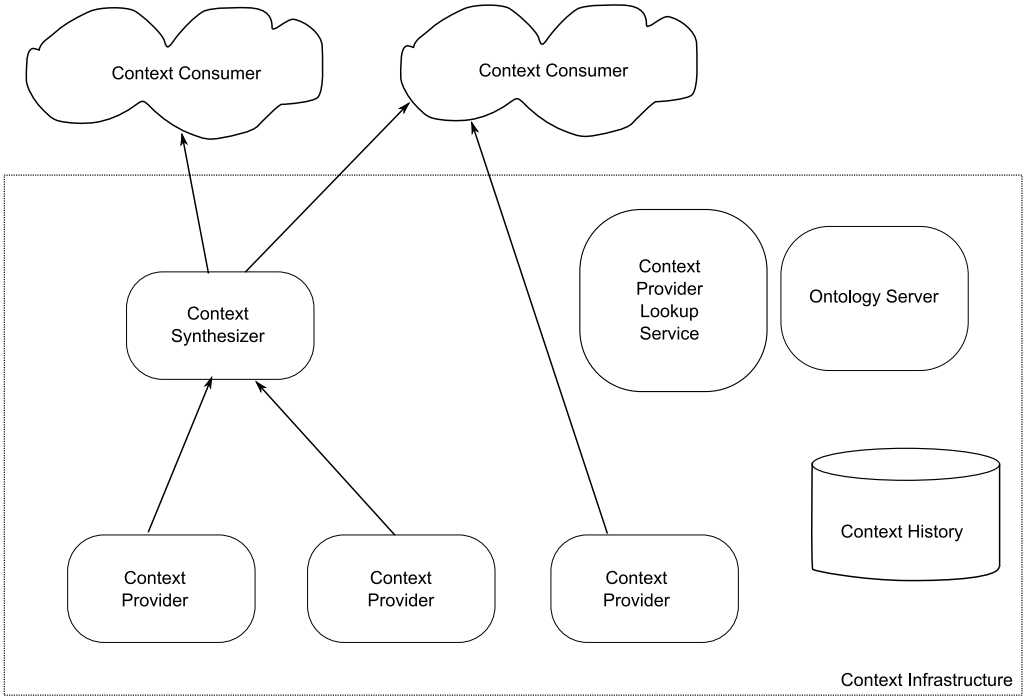


FIGURE 16 – Contexte Gaia.

L'architecture se compose de :

Fournisseurs de contexte : Ils collectent différents types d'informations de contexte et les rendent accessibles aux applications. Ces informations sont représentées par des expressions logiques. Par exemple pour un service de localisation suivant l'utilisateur Bob, on aura :
$\forall_{Localisation}\ y\ Localisation(Bob,\ In,\ y)$

Consommateurs de contexte ou applications sensibles au contexte : Ils obtiennent les informations de contexte en envoyant une requête aux fournisseurs de contexte ou en écoutant les événements produits par les fournisseurs en souscrivant aux canaux déclarés par les fournisseurs d'informations de contexte. Les requêtes sont évaluées de la même façon que dans Prolog et sont de la même forme que les prédicats avec une variable (de la forme $?nomDeLaVariable$) à la place d'un argument. Par exemple, la requête $Location(?X, In, Room3231)$ permet de savoir qui se situe dans la pièce 3231. Les événements sont de la forme des prédicats de contexte, $Location(Bob, In, Room3231)$ est l'événement qui correspond à l'entrée de Bob dans la pièce 3231. Les consommateurs découvrent les fournisseurs disponibles en utilisant un service de recherche de fournisseurs de contexte. Un mécanisme est fourni aux développeurs pour spécifier des règles d'adaptation des consommateurs basées sur Prolog.

Synthétiseurs de contexte : Ils infèrent des informations de contexte de plus haut niveau en se basant sur des informations de contexte simples provenant des capteurs. Ils sont à la fois fournisseurs et consommateurs d'informations de contexte. Il en existe deux types : basés sur un système de règles ou basés sur des techniques d'apprentissage.

Service de recherche de fournisseurs de contexte : Les fournisseurs de contexte envoient au service de recherche une annonce qui correspond aux informations fournies sous la forme d'une expression de la logique du premier ordre. Pour un service de localisation on aura l'annonce suivante : $\forall_{People}x, \forall_{Room}yLocation(x, In, y)$. Le service de recherche de fournisseurs de service fait la correspondance entre cette annonce et la requête, par exemple $Location(?X, In, room3231)$, envoyée par un consommateur d'information de contexte. Ainsi, il pourra lui envoyer les informations pour contacter le fournisseur adéquat.

Historique de contexte : Ce service permet d'enregistrer tous les événements de contexte dans une base de données. Ces informations sont notamment utilisées par les systèmes d'apprentissage.

Analyse

À l'image des systèmes d'exploitation que Gaia souhaite adapter pour les environnements d'intelligence ambiante, cette infrastructure offre un éventail riche de services transversaux. Ainsi, le but exposé par les auteurs est atteint : une grande facilité pour développer des applications intégrées dans cette infrastructure et pour coder leurs comportements. Malheureusement, cette facilité a le désavantage de contraindre fortement les développeurs avec l'obligation d'utiliser les technologies supportées par l'infrastructure.

Le modèle de contexte proposé est interopérable et basé sur la logique du premier ordre. L'utilisation des ontologies permet, encore une fois, une compréhension uniforme des prédicats entre les différents agents. Cependant, il faut que les agent se réfèrent à une ontologie particulière, fondation du modèle de contexte. Si on souhaite faire interopérer plusieurs environnements, il doit exister des correspondances entre les ontologies étrangères et l'ontologie de référence de l'environnement.

2.4.5 L'architecture CoBrA (Context Broker Architecture)

Le système de gestion d'information de contexte proposé par Tim Finin, Harry Chen et Anupam Joshi dans [20] est basé sur un élément logiciel appelé *Context Broker* (voir la figure 17). Ce dernier est un serveur central destiné à partager les informations de contexte entre les différents dispositifs dans un environnement intelligent. Il a cinq fonctionnalités :

- Fournir un modèle de contexte qui sera partagé par tous les dispositifs et applications de l'environnement.
- Acquérir les informations de contexte à partir de capteurs et des autres sources d'information de contexte.
- Déduire les informations de contexte qui ne peuvent pas être directement obtenues grâce aux capteurs (intentions, rôles, relations spatio-temporelles).
- Maintenir une base de connaissances d'informations de contexte cohérente.
- Protéger les informations relatives à l'utilisateur et à son contexte.

Architecture

L'architecture CoBra est organisée autour de quatre modules :

- Une base de connaissances d'informations de contexte qui gère les informations sur l'environnement et fournit aux composants de l'environnement un ensemble d'API pour mettre à jour cette base de connaissance et l'interroger. Cette base de connaissances repose sur une ontologie spécifique à l'environnement intelligent (une salle de réunion par exemple) ainsi qu'un ensemble d'heuristiques qui lui est propre (une personne ne peut pas être dans deux salles de réunion différentes au même moment).
- Un moteur d'inférence sur les informations de contexte qui maintient la consistance de la base d'informations de contexte à partir des heuristiques définies et qui déduit des nouvelles connaissances à partir des informations recueillies et de l'ontologie.
- Un module d'acquisition de contexte qui est un ensemble de procédures pour accéder simplement aux informations fournies par les capteurs hétérogènes. Il a un rôle comparable aux context-widget du Context Toolkit (2.3.2).
- Un module pour la gestion de la vie privée. C'est un ensemble de règles et de protocoles qui definissent quels dispositifs ou quelles applications ont le droit d'accéder à quelles informations de contexte ou d'être notifiés lors d'une de leurs mises-à-jour.

Ontologies

Pour assurer l'interopérabilité des différents dispositifs et le partage de l'information de contexte, les dispositifs qui utilisent l'architecture CoBrA utilisent la même ontologie, *CoBrA ONT*, pour décrire leurs informations de contexte. Cette ontologie étend l'ontologie SOUPA et introduit un vocabulaire spécifique au domaine. SOUPA se décompose en deux ensembles distincts mais interopérants d'ontologies : SOUPA Core et SOUPA Extension. L'ensemble des ontologies de SOUPA Core définit les vocabulaires génériques pour bâtir des applications d'informatique diffuse. L'ensemble des ontologies de SOUPA Extension définit de nouveaux vocabulaires pour construire des applications spécifiques et fournit des exemples pour définir de nouvelles extensions. Afin de permettre une meilleure interopérabilité entre les applications utilisant l'ontologie SOUPA et les applications utilisant d'autres ontologies, de nombreux termes définis dans SOUPA ont des correspondances explicites, exprimés à l'aide des constructeurs OWL `owl:equivalentClass` et `owl:equivalentProperty` par exemple, avec des termes utilisés dans des ontologies standards. Les ontologies qui sont référencées dans SOUPA sont les suivantes :

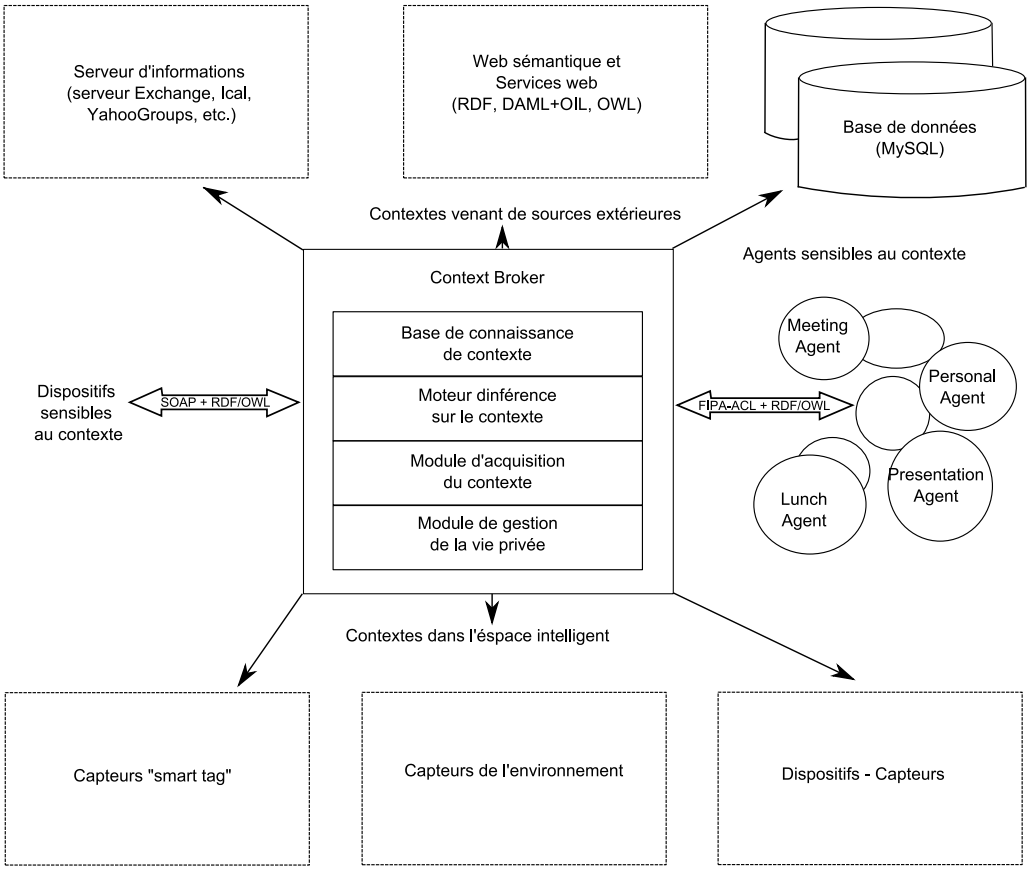


FIGURE 17 – Architecture CoBra.

- Friend-Of-A-Friend (FOAF)[14, 63] : l'ontologie FOAF permet l'expression d'informations personnelles et des liens sociaux.
- DAML-Time et the Entry Sub-ontology of Time [48] : Les vocabulaires de ces ontologies sont conçus pour exprimer les concepts temporels et les propriétés communes à toute formalisation du temps.
- OpenCyc Spatial Ontologies [56] et Regional Connection Calculus (RCC) [66] : Elles définissent un ensemble complet de vocabulaires pour la représentation symbolique de l'espace. L'ontologie RCC se compose de vocabulaires pour exprimer les relations spatiales pour le raisonnement qualitatif spatial.
- COBRA-ONT [20] et l'ontologie de MoGATU BDI [61] : Elles ont pour but d'appuyer la représentation de la connaissance et le raisonnement dans les environnements d'intelligence ambiante. Alors que la conception de COBRA - ONT se concentre sur la modélisation des contextes dans les salles de réunion intelligentes [20], la conception de l'ontologie MoGATU BDI se concentre sur la modélisation de la croyance, du désir et de l'intention des utilisateurs et des agents logiciels.
- Rei Policy Ontology [51] : Elle définit un ensemble de notions déontiques (c'est-à-dire sur les droits, les interdictions, obligations et dérogations) pour la spécification et le raisonnement sur la sécurité des règles de contrôle d'accès.

Ontologie SOUPA L'ontologie SOUPA Core se décompose en neuf ontologies. Elles définissent des vocabulaires pour la description des personnes, de croyances, de désirs, les intentions d'un agent, d'actions, des politiques de sécurité, du temps, et de l'espace et des événements.

SOUPA Extension Les ontologies SOUPA Extension ont été définies pour répondre à deux objectifs :
- Définir un vocabulaire pour permettre l'introduction de nombreux types d'applications d'informatique diffuse,
- Montrer comment définir de nouvelles ontologies en étendant l'ontologie de base SOUPA.

Actuellement, l'ontologie SOUPA Extension est composée expérimentalement d'ontologies servant à des applications d'espaces intelligents et d'échange de données pair-à-pair. Elle offre un vocabulaire pour décrire la priorité, c'est-à-dire des valeurs de préférence pour les désirs et les actions prévues d'un agent, les croyances conditionnelles et inconditionnelles (temporalité, précision...) d'un agent, les préférences personnelles d'un utilisateur notamment en ce qui concerne ses modes de communication et enfin des rendez-vous et un calendrier.

Analyse

D'après les auteurs, l'architecture CoBrA vise à permettre le partage de connaissances et la fusion des données de contexte entre les dispositifs d'un environnement d'intelligence ambiante, notamment en évitant que la connaissance du contexte ne soit trop fortement intégrée dans les objets programatiques (classes Java ou C++ par exemple) qui construisent les dispositifs de l'environnement. Les auteurs veulent, en plus, pouvoir caractériser précisément les informations de contexte, prendre en compte les exactitudes et pouvoir qualifier la précision des informations provenant des capteurs et faire face aux inconsistences et incompatibilités fréquentes des informations de contexte.

Afin de remédier à ces problèmes, l'infrastructure CoBra utilise des ontologies. L'apport de l'utilisation des ontologies pour la construction de ce système sensible au contexte sont les suivantes :
- une ontologie commune permet le partage des connaissances dans un système distribué ouvert et dynamique [62],
- les ontologies offrent une représentation des informations de contexte avec une sémantique bien définie permettant aux agent intelligents de raisonner dessus,
- les ontologies autorisent les appareils et les agents qui n'ont pas été conçus pour fonctionner ensemble d'interopérer.

Cependant, pour atteindre l'objectif d'interopérabilité entre les dispositifs, il faut qu'ils utilisent tous la même ontologie ou un ensemble d'ontologie défini à la conception du système. Même s'ils ont été conçus de manière indépendante, ces dispositifs doivent avoir une connaissance préétablie de l'ontologie utilisée dans l'environnement, ce qui ne facilite pas leur intégration dans ce type d'environnement ouvert. Il est intéressant de remarquer que les ontologies SOUPA couvrent correctement l'ensemble des concepts utiles pour le développement d'environnements intelligents. Ce travail de formalisation est utile pour modéliser ce type d'environnement en utilisant directement les ontologies SOUPA ou les ontologies sur lesquelles elles s'appuient.

2.5 Bilan

La littérature compte de nombreux travaux qui proposent une solution pour la gestion des informations de contexte dans les environnements d'intelligence ambiante. Nous en avons sélectionné sept pour notre revue de l'état de l'art soit qu'ils sont fondateurs [71, 29], soit que leur mise en œuvre utilise des technologies semblables à celle que nous utilisons, soit que nous partagions, au moins en partie, leurs

modélisations du contexte [77]. Mais cette problématique est prolixe en solutions et nous allons dresser la liste des objectifs à remplir pour qu'une solution soit utilisable.

Les environnements d'intelligence ambiante doivent être ouverts, c'est-à-dire qu'un utilisateur, une application ou un dispositif peut entrer dans cet environnement et doit pouvoir y participer en tant qu'acteur. Les infrastructures de gestion d'informations de contexte visent à fournir les informations de contexte nécessaires à l'éxecution d'une application ou au bon fonctionnement d'un dispositif si un service ou un capteur qui fournit cette information est accessible à travers un réseau. On peut espérer l'efficacité maximum pour une infrastructure de gestion d'information de contexte si elle fonctionne avec tous les dispositifs, toutes les applications et tout les capteurs présents dans son environnement.

Supporter l'ouverture de l'environnement impose aussi aux infrastructures d'être capables de prendre en compte pendant l'exécution et sans pertubation l'arrivée de nouveaux acteurs. La plupart des infrastructures sont équipées de fonctions ou de services de découverte afin de pouvoir intégrer de nouveaux acteurs à tout moment. Dans le meilleur des cas, pour intégrer une infrastructure il faut respecter le protocole qu'elle définit, mais parfois ce protocole de découverte est complexe. Seule l'infrastructure des contexteurs présente un protocole simple pour y ajouter un nouveau dispositif.

L'environnement doit supporter l'hétérogénéité de ses acteurs. Il doit intégrer une fonctionnalité permettant aux acteurs de profiter des fonctionnalités des uns et des autres alors même qu'ils ne se connaissent pas au préalable et qu'ils n'ont pas forcément été construits pour collaborer. Les infrastructures CoBra, CoOl et Gaia offrent cette fonctionnalité en proposant aux concepteurs de décrire grâce à un vocabulaire défini dans un nombre fini d'ontologies, les fonctionnalités offertes ou recherchées par leurs applications. En utilisant ce vocabulaire pour décrire les informations de contexte produites ou recherchées, les applications, les dispositifs et les capteurs peuvent profiter de l'infrastructure de gestion. Pour sa part l'infrastructure basée sur les contexteurs utilise le même mécanisme avec des descriptions de contexte en XML. D'autres infrastructures (Dey et les contexteurs) présentées dans ce chapitre s'appuient sur des compatibilités programmatiques. C'est-à-dire que pour pouvoir fonctionner avec ces infrastructures les dispositifs, applications et capteurs doivent implémenter des interfaces particulières. Supporter l'hétérogénéité pour une infrastructure de gestion d'information de contexte, c'est être capable de récolter et de diffuser des informations de contexte connues auprès de dispositifs, d'applications et de capteurs inconnus.

Une infrastructure offrant le support de l'hétérogénéité de ses acteurs et de l'ouverture de l'environnement permet le développement d'applications sensibles au contexte pour les contextes que l'on aura prévus à la conception de l'infrastructure. Nous avons vu dans 1.3 que le contexte ne peut être défini a priori, qu'il est défini par rapport à une application et qu'il doit être extensible. Un capteur doit pouvoir enrichir la connaissance de l'environnement même si cette connaissance n'a pas été prévue dans l'infrastructure. De la même façon, une application qui recherche une information de contexte doit pouvoir caractériser le monde avec sa propre vision qui ne doit pas être contrainte par l'infrastructure. Pour ouvrir leur fonctionnement, certaines infrastructures utilisent des ontologies. Pour décrire ou classer les informations de contexte que les dispositifs fournissent ou qui sont disponibles dans l'environnement, ces ontologies sont utilisées de façon rigide dans toutes les solutions étudiées. Même si elle sont parfois découpées en ontologies modulaires, comme dans l'ontologie SOUPA [21], chacune d'entre elles representant une partie du monde, elles ne permettent qu'une seule représentation du monde à laquelle les acteurs de l'environnement sont obligés d'adhérer pour pouvoir profiter de l'infrastructure. Par le jeu de la subsomption, elles peuvent dans certains cas [77] être étendues avec de nouveaux concepts, mais dans tous les cas, leurs contenus et leur organisation sont fixés par les concepteurs de l'environnement à sa conception.

Il nous semble indispensable de remplir ces trois objectifs (ouverture, dynamisme et extensibilté du modèle de contexte) pour le bon fonctionnement d'une infrastructure de gestion d'information de contexte. Mais, s'il existe des infrastructures qui traitent l'un ou l'autre de ses aspects, aucune ne satisfait

complètement les trois.

Les solutions qui existent sont soit peu génériques, soit ne supportent pas les environnements ouverts. Les infrastructures Context ToolKit et Contexteurs ne supportent l'introduction que de dispositifs connus à l'avance. Certes, elles ne nécessitent pas d'arrêt dans leur exécution pour prendre en compte ces dispositifs mais ils doivent présenter des fonctionnalités uniquement prévues dans leur infrastructure respective. L'autre principale faiblesse des infrastructures présentées dans cet état de l'art, et plus généralement dans la littérature, est qu'elles fonctionnent uniquement avec des contextes préétablis à la conception de l'environnement. Même les infrastructures qui utilisent des ontologies comme vocabulaire de description pour les connaissances de contexte, ce qui peut permettre de couvrir l'ensemble des domaines de l'intelligence ambiante, fonctionnent avec des ontologies imposées dès la conception de l'environnement qui ne peuvent pas être ni changées, ni enrichies. Les fabricants sont contraints d'utiliser uniquement le vocabulaire de l'infrastructure pour modéliser les informations de contexte prises en compte par leurs applications, dispositifs ou capteurs.

Prendre en compte l'hétérogénéité, c'est aussi s'assurer, pour une infrastructure, de pouvoir s'interfacer avec un dispositif quelles que soient ses capacités de calculs ou quel que soit l'environnement d'exécution d'une application. Ce sont les développeurs d'infrastructure de gestion d'informations de contexte qui devront fournir un effort de conception remarquable pour faciliter cette compatibilité et pour minimiser le travail des fabricants pour rattacher leurs dispositifs à cette infrastructure. C'est pourquoi nous intégrons un critère de minimalité à notre analyse. Il évaluera l'effort à fournir pour intégrer un dispositif ou une application à une infrastructure.

Nom	Supporte			Facilité d'intégration
	Hétérogénéité	Dynamisme	Extensibilité du modèle	
Schilit	+	+	??	-
Context Toolkit	++	+	??	+
Les contexteurs	+	++	??	+
CoOL	++	+	-	-
SOCAM	++	+	+	-
Gaia	+	??	+	+
CoBra	++	+	+	-
But à atteindre	++	++	++	++

TABLE 2.1 – Tableau récapitulatif des critères d'évalutation : ?? non mentionné, - non supporté , + supporté, ++ clairement explicité.

À l'image des infrastructures que nous avons détaillées dans cet état de l'art, les infrastructures existantes offrent, pour la plupart, des fonctionnalités qui facilitent et enrichissent le service de gestion d'informations de contexte, mais aucune d'entre elles ne peut être réellement déployée dans un environnement domestique intelligent. De par leur ouverture ces environnements sont très peu contraints et les infrastructures détaillées dans ce chapitre ne supportent généralement que partiellement les spécificités de ce type d'environnement : hétérogénéité, dynamisme et extensibilité des modèles de contexte.

Dans le but d'offrir une infrastructure de gestion d'informations de contexte pour les environnements d'intelligence ambiante, il faut la concevoir pour qu'elle fonctionne dans un monde ouvert [55]. Il est nécessaire de construire cette infrastructure pour qu'elle soit dynamique, qu'elle supporte l'hétérogénéité des dispositifs et des applications, l'extensibilité des modèles de contexte et ne demande qu'un effort minimum aux concepteurs pour s'y rattacher.

Dans un environnement ouvert, notre approche est de donner la liberté aux concepteurs d'utiliser des modèles extensibles pour représenter les informations de contexte qu'ils pourront exhiber sur le réseau, et

Nom	Fonctionnalités				Typologie
	Historique	Sécurité	Échange de données	Qualité de service	
Schilit	-	-	SN	-	Flots de données
Context Toolkit	+	-	SN	-	Faiblement centralisée
Les contexteurs	+	+	RR et SN	+	Décentralisée
CoOL	+	-	RR	-	Centralisée
SOCAM	+	-	RR et SN	+	Centralisée
Gaia	+	+	RR et SN	-	Faiblement centralisée
CoBra	+	+	RR et SN	-	Centralisée

TABLE 2.2 – Tableau récapitulatif des fonctionnalités offertes : - non supporté, + implementé ; SN pour Souscription-Notification et RR pour Requête-Réponse

de leur offrir les outils nécessaires à la découverte des modèles d'informations de contexte compatibles. Le premier volet de notre étude s'attachera à la conception d'une architecture, puis nous étudierons la modélisation des informations de contexte, le tout devant correpondre aux spécifications dégagées (1.3) et aux objectifs que nous nous somme fixés (2.2.2). Pour les atteindre, nous nous attacherons à utiliser des technologies standard, à rechercher le protocole d'interaction minimum entre les acteurs de l'environnement et à utiliser des technologies du web sémantique particulièrement recommandées pour les environnement d'informatique diffuse [76]. Nous exposerons dans les chapitres suivants : le modèle d'infrastructure de gestion d'information de contexte que nous proposons, l'implémentation que nous en avons réalisée et l'utilisation que nous en avons faite dans la construction d'un environnement d'intelligence ambiante.

Deuxième partie

Gestion dynamique des informations de contexte en intelligence ambiante : architecture distribuée et modèles sémantiques

Cette seconde partie du manuscrit décrit en détail nos travaux et notre contribution en quatre chapitres. Nous détaillerons le modèle conceptuel et argumenterons les choix architecturaux qui découlent de l'analyse de l'état de l'art. Puis, nous présenterons les réalisations techniques de la thèse, c'est-à-dire une implémentation du modèle conceptuel. Enfin, nous présenterons un cas d'utilisation réel de l'infrastructure.

Afin de disposer des informations de contexte, les applications d'intelligence ambiante doivent les puiser directement ou indirectement des capteurs en relation avec l'environnement. Nous souhaitons proposer une architecture permettant de réaliser ceci de la manière la plus flexible possible, c'est-à-dire une infrastructure dans laquelle tous les dispositifs peuvent s'intégrer et participer à l'intelligence de l'environnement, ou en profiter, sans altérer ce dernier. Nous définissons le contexte d'une application comme l'ensemble des informations utiles à son fonctionnement dans un environnement intelligent, et par extension le contexte d'un capteur ou d'un service producteur d'informations de contexte contient les informations qu'il produit caractérisant l'environnement. Par ailleurs, nous souhaitons donner la possibilité aux concepteurs de ces applications et de ces dispositifs de modéliser le contexte selon leur propre vision du monde. Nous avons vu que de modéliser le contexte en utilisant des technologies du web sémantique peut offrir cette possibilité.

Envisager une architecture où les services sont attentifs au contexte, conduit à proposer une infrastructure de gestion du contexte capable de servir aux autres dispositifs les informations de contexte. Cette infrastructure doit répondre aux critères exposés dans la partie 2.2, c'est-à-dire une infrastructure qui supporte l'ouverture des environnements d'intelligence ambiante. Les nouveaux dispositifs et applications pourront être introduits dans l'environnement et y participer "à chaud" sans altération de ce dernier. L'infrastructure proposée devra fournir un protocole de découverte qui aboutit à la caractérisation des fonctionnalités de chacun des acteurs. Ces derniers sont potentiellement hétérogènes produisant ou recherchant des informations de contexte variées et fournies par des fabricants indépendants ; l'infrastructure devra pouvoir accepter ces divers aspects du contexte et ne pas les limiter dans l'expression de leur contexte particulier. Ainsi, l'infrastructure doit être capable de représenter les informations de contexte introduites par des dispositifs (potentiellement inconnus), et de trouver des correspondances entre ces dernières et les représentations des informations de contexte déja présentes dans l'environnement pour assurer leur interopérabilité. Enfin, si l'on veut qu'un maximum de dispositifs et d'applications adhèrent à notre infrastructure et y participent, il faut veiller à ce que l'effort à fournir par leurs développeurs pour qu'ils s'intègrent dans cette infrastructure soit minimum. Il faut pour cela veiller à ce que l'infrastructure consomme le minimum de ressources et que l'interface nécessaire à l'utilisation de l'infrastructure soit elle aussi minimale.

Cette seconde partie se découpe comme suit :

- Dans le premier chapitre nous discuterons de la conception de l'architecture à adopter pour notre infrastructure. De plus, on s'intéressera aux flux d'informations de contexte des producteurs vers les consommateurs et au protocole à mettre en place pour permettre la découverte et les échanges.
- Le deuxième chapitre précisera l'utilisation des technologies du web sémantique dans la construction des modèles d'information de contexte ainsi que le mécanisme servant à leur réconciliation.
- Le troisième chapitre présente l'implémentation réalisée de l'infrastructure conceptuelle présentée dans le chapitre précédent ainsi que les technologies utilisées pour ce faire.
- Avant de conclure le manuscrit, nous présentons un cas d'utilisation des technologies développées dans un démonstrateur réalisé à France Telecom et présenté au salon de la recherche France Telecom et à Ubicomp [13]. Il met en scène un utilisateur et l'organisation de ses loisirs dans le cadre domestique.

13. `http://www.ubicomp2007.org/program/demo/`

Chapitre 3

Architecture distribuée dans un monde ouvert

Sommaire

3.1 Architecture de gestion pair à pair 65
3.1.1 Connexion directe 66
3.1.2 Service de gestion de contexte centralisé 66
3.1.3 Service de gestion de contexte distribué 68
3.2 Fonctionnement 69
3.2.1 Protocole d'interaction 69
3.2.2 Exemple 70
3.3 Bilan 72

Nous présenterons les architectures possibles pour la gestion des informations de contexte et nous discuterons leurs avantages et inconvénients vis-à-vis de leur utilisation dans un environnement d'intelligence ambiante spécifiquement ouvert. Puis, nous spécifierons un protocole d'interaction entre les producteurs et les consommateurs d'informations de contexte. Nous expliquerons ensuite comment une telle architecture peut tirer parti de ce modèle et convenir parfaitement aux besoins identifiés pour les environnements d'intelligence ambiante. Notre solution consistera à prendre en compte l'aspect dynamique de ces envionnements ainsi que d'offrir un support à l'hétérogénéité et l'extensibilité des modèles de contexte de l'intelligence ambiante à deux niveaux :

- Une architecture permettant à de nouveaux dispositifs, capteurs ou applications de s'introduire dans le réseau à tout moment,
- Une représentation des éléments de contexte dont on montrera qu'elle permet d'intégrer de nouveaux élements sans rupture ni modification d'exécution des processus en cours dans l'environnement.

Ces deux niveaux satisfont bien aux conditions d'ouverture et de dynamisme. La minimalité, quant à elle, est assurée en utilisant des technologies légères et standards. L'architecture est présentée dans ce qui suit et la représentation au chapitre suivant.

3.1 Architecture de gestion pair à pair

On l'a vu, le recueil d'informations sur le contexte physique se fait par le biais de capteurs (de température, de luminosité, etc.) ou de dispositifs d'acquisition tel qu'une caméra ou un micro, couplés à une application chargée d'extraire des informations sur l'environnement en réalisant des traitements

sur le signal fourni. Si ces capteurs n'ont généralement pas la capacité à gérer explicitement un contexte, on considèrera qu'ils sont vus par les systèmes d'informatique diffuse au travers de proxys capables de fournir les informations capturées par les capteurs.

3.1.1 Connexion directe

La première possibilité de connexion entre les producteurs d'informations de contexte et les consommateurs consiste à considérer que les applications communiquent directement avec les capteurs dont elles ont besoin (voir la figure 18). Cette architecture directe a l'inconvénient d'obliger les applications à savoir à qui s'adresser pour obtenir l'information nécessaire. Elle complexifie l'agrégation des informations qui doit être réalisée à chaque fois par l'application. Finalement, elle ne permet pas de tirer parti de nouveaux capteurs dans l'environnement et pose donc d'importants problèmes pour la dynamique des applications et de l'environnement. L'infrastructure de Schilit (voir 2.3.1) est un exemple d'architecture à connexion directe entre les capteurs et les applications.

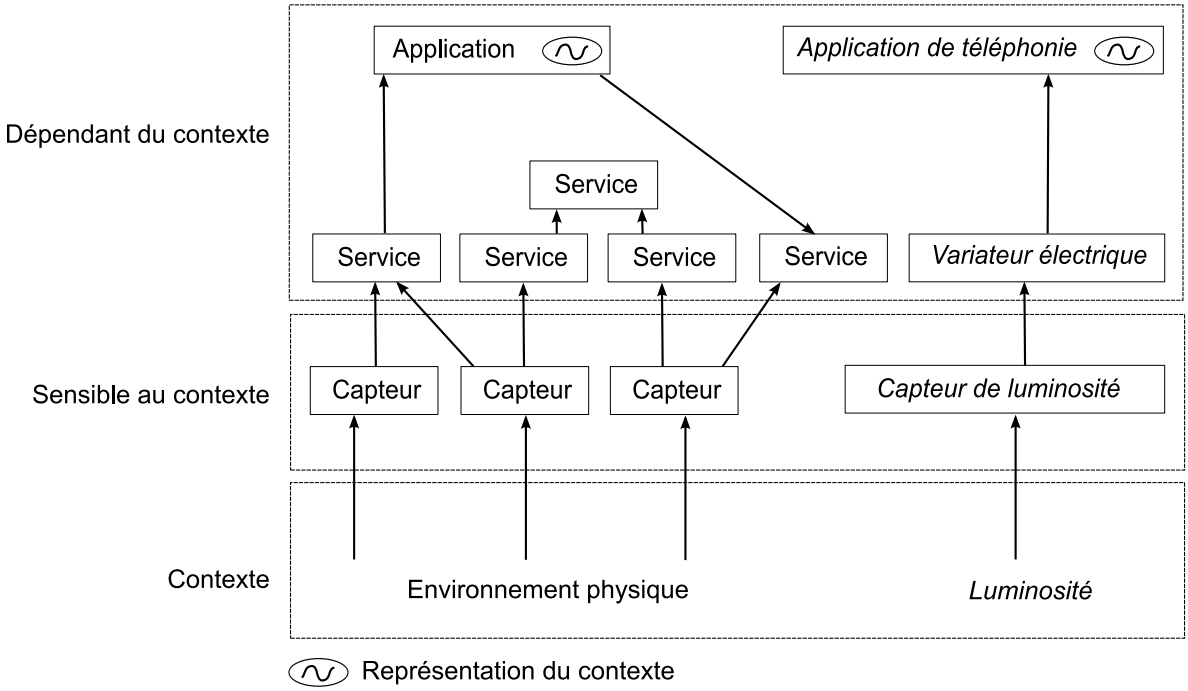


FIGURE 18 – Architecture générale d'un environnement d'informatique diffuse : les capteurs, par exemple, les capteurs de lumière, fournissent des informations sur l'environnement (par exemple la luminosité). Ces informations permettent aux autres services et applications d'évaluer le contexte d'une tâche (par exemple la luminosité d'une partie de l'environnement). Ce sont les applications qui prennent en charge la représentation du contexte dont ils ont besoin (les mentions en italiques sont des exemples, les flèches représentent les flots d'informations).

3.1.2 Service de gestion de contexte centralisé

Dans le cadre d'une architecture basée sur les services ("Service oriented architecture"), la seconde solution consiste à développer un service de gestion de contexte [20] qui permettra de centraliser les

informations et les communiquera aux applications qui les nécessitent (voir la figure 19 et les infrastructures telle que CoOL (voir section 2.4.2), SOCAM (voir section 2.4.3) et celle du "Context Broker" (voir section 2.4.5). Les dispositifs, applications et capteurs sont dans cette infrastructure clients du service de gestion de l'information de contexte. L'avantage de cette vision est de mettre en commun les informations que l'on trouve dans l'environnement, ce qui permet d'agréger facilement les informations récoltées. Par exemple, dans une maison, il est utile d'avoir un système capable de donner la température et la pression. À l'échelle d'une ville, les mêmes informations sont intéressantes mais avec un degré de précision différent. Le défaut d'un tel service de gestion de contexte est de centraliser cette gestion, ce qui est contradictoire avec la notion de contexte attaché à une activité. Il fournit effectivement des informations sur l'environnement d'une activité, et donc des informations de contexte, mais elles ne sont pas contextuelles car elles ne dépendent plus de la tâche ou de la situation courante (telle que prévue par l'application cliente). De plus, un tel système ne permettrait une gestion de contexte efficace que dans un périmètre restreint. Dans le cadre de nos travaux et en collaboration avec le projet européen AMIGO [14], nous avons implémenté une infrastructure de gestion de contexte centralisée [65] utilisant les technologies du web sémantique. Cette expérimentation, nous a montré les limites conceptuelles d'une telle infrastructure qui ne permet pas de gérer les contextes de chaque dispositif et application indépendement des uns et des autres, ainsi que la grande complexification du modèle de contexte et des règles qui le maintiennent consistant à chaque ajout de dispositif.

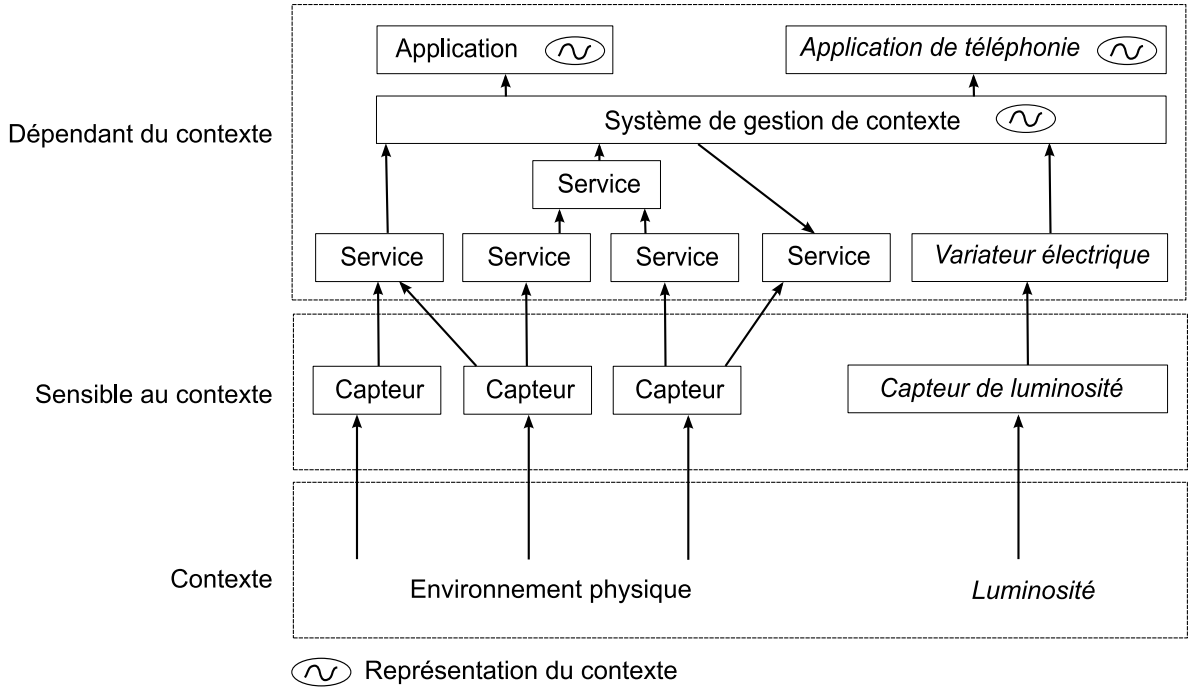


FIGURE 19 – Architecture de gestion d'information de contexte centralisée : les informations de contexte en provenance des capteurs sont agrégées dans un système central où les applications viennent chercher les informations sur l'environnement.

14. http://www.hitech-projects.com/euprojects/amigo/

3.1.3 Service de gestion de contexte distribué

Nous avons adopté la posture qui consiste à définir un contexte en l'associant à une tâche réalisée par un utilisateur (voir section 1.3). Un environnement intelligent met à disposition des utilisateurs des applications de la façon la plus simple et la plus transparente possible. On considère que la réalisation d'une tâche pour l'utilisateur est associée à l'utilisation d'une application et donc que chacune des applications considère un contexte distinct. Ainsi, la solution que nous retenons consiste à considérer que chaque dispositif impliqué dans l'architecture au-dessus de la couche capteurs sera doté d'un composant de gestion de contexte dont le but est de maintenir les informations de contexte pour ses besoins propres ou pour le service des autres (voir la figure 20). Les dispositifs qui le désirent peuvent publier leurs capacités auprès d'annuaires permettant de les localiser, ou l'architecture peut fonctionner sur la base d'un bus auquel se connectent les dispositifs nouvellement introduits dans l'environnement. Un consommateur d'information de contexte, après avoir localisé les producteurs, cherchera à établir des correspondances entre les capacités de chaque producteur d'informations de contexte et ses besoins et ainsi déterminer avec quels producteurs il doit interagir. L'avantage de cette solution est, bien entendu, que de nouveaux dispositifs peuvent être dynamiquement ajoutés ou retranchés de l'environnement sans avoir à réinitialiser les autres éléments de l'architecture. Par ailleurs n'importe quel composant ayant besoin d'information n'a pas à se soucier de à qui il le demande : capteurs, services centralisés et autres applications réagissent de la même manière. Il peut se focaliser sur l'information nécessaire.

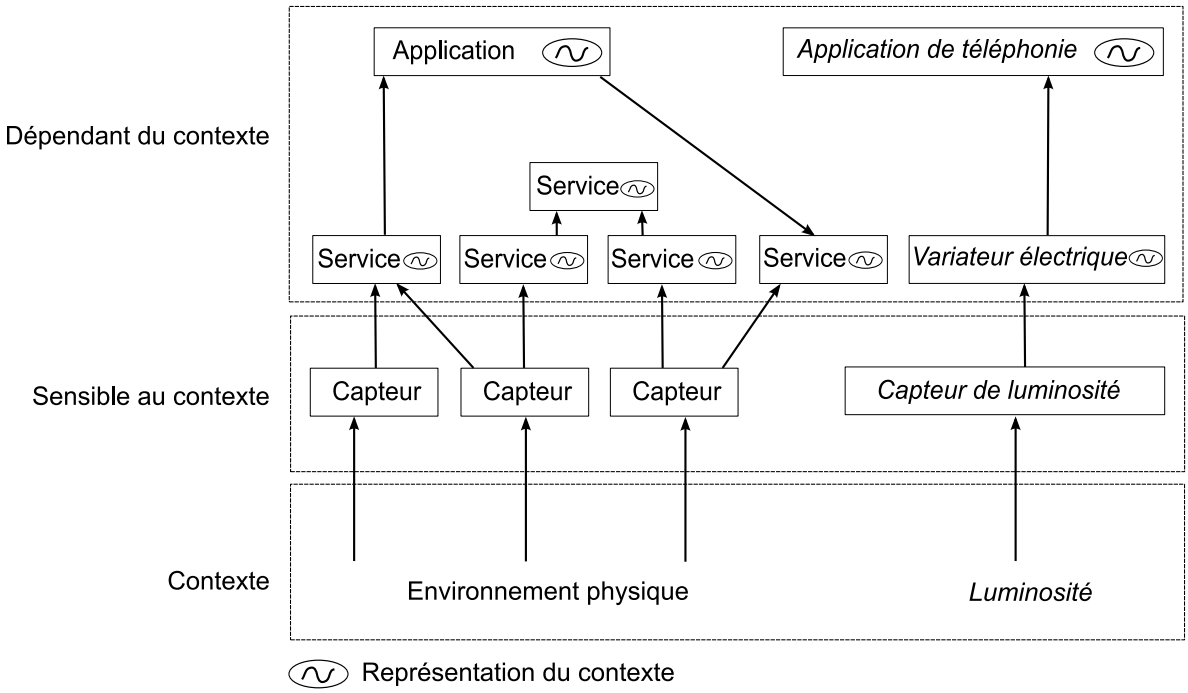


FIGURE 20 – Le contexte, composant de chaque service et application.

La gestion de contexte sera donc implémentée comme une bibliothèque pouvant être embarquée dans n'importe quel dispositif de la figure 18. Elle devra permettre aux dispositifs consommateurs de demander des informations de contexte aux dispositifs producteurs d'informations de contexte. On retrouve ces idées dans les infrastructures de Dey (voir section 2.3.2), Gaia (voir section 2.4.4) et l'architecture des contexteurs (voir section 2.3.3). Avant d'intéragir directement avec les dispositifs producteurs d'informations de contexte, les consommateurs interrogeront préalablement un service d'annuaire que l'on

suppose présent dans l'environnement d'exécution d'intelligence ambiante. Ce service d'annuaire devra caractériser les producteurs d'informations de contexte de façons généraliste, la recherche précise des informations de contexte souhaitées par les consommateurs se fera dans un deuxième temps grâce à un protocole que nous définissons dans la section suivante. Ainsi l'utilisation de notre infrastructure ne demande aucune modification sur l'environnement existant, charge aux fabricants de consommateurs et de producteurs d'information de contexte de réaliser cet enregistrement et l'interrogation auprès d'un service d'annuaire. Cependant cette étape de découverte des producteurs d'informations de contexte n'est pas indispensable au bon fonctionnement de notre infrastructure, car l'interaction entre producteurs et consommateurs peut commencer par la diffusion en "broadcast" du premier message du protocole (voir la section 3.2.1).

3.2 Fonctionnement

Notre infrastructure est construite en fournissant les moyens à chaque dispositif ou service de l'environnement de gérer les informations relatives à son contexte. Pour leur permettre de diffuser ces informations sur le réseau ou de rechercher des informations qui les intéressent sur l'environnement, nous proposons un protocole d'interaction pour exploiter cette modélisation sémantique de l'information de contexte dans l'architecture répartie proposée.

3.2.1 Protocole d'interaction

Les applications doivent pouvoir demander les éléments de contexte qui les concernent à des capteurs ou à des services fournissant des informations agrégées. Il est donc nécessaire de proposer un protocole permettant à tout dispositif de tirer le meilleur parti des services disponibles. Ce protocole doit répondre aux objectif de dynamisme et de minimalité de l'infrastructure. En particulier, vis-à-vis de la seconde exigence, les services minimaux doivent être d'identifier un service, savoir quelles informations de contexte il fournit et s'adresser à lui pour obtenir cette information. Pour cela, le composant de gestion de contexte devra implémenter quelques primitives (le premier élément est la requête, le second le type de la réponse) :

- `Id()`→`URI` : l'identificateur du service fourni ;
- `Cl(URI)`→`URI` : la classe du service identifié ;
- `Desc(URI)`→`OWL` : la description de l'information que le composant peut fournir : $Desc(u) = O'$
- `Req(SPARQL)`→affectations `RDF` : les tuples des valeurs de variables qui satisfont la requête : $Req(q) = S_1, \ldots, S_n$

Producteur de contexte Soit un ensemble d'ontologies O et un ensemble d'alignements A (en théorie tous les alignements possibles entre les ontologies de O). Un producteur de contexte c est caractérisé par quatre fonctions :

- $Id_c = c$
- $Cl_c \in uri$:
- $Desc_c \subseteq \cup_{o \in O} o$:
- $Req_c : SPARQL \rightarrow 2^{O^n}$:

Id peut paraître inutile et Cl est généralement nécessaire pour obtenir $Desc$, par conséquent, il ne sera pas utilisée dans ce qui suit. Mais dans la pratique, ces deux opérations se révèlent utiles (voir chapitre 5). Nous ne supposons pas que les réponses aux requêtes proviennent d'une ontologie unique.

Signature d'une requête La signature $\sigma(q)$ d'une requête σ est l'ensemble des types qui apparaissent dans cette requête (par conséquent $\sigma(q) \subseteq \cup_{o \in O} o$).

On peut remarquer que d'autres standards auraient pu être utilisés. Notamment, utiliser WSDL [81] dans la primitive Desc(URI) pour décrire les capacités du service de contexte est envisageable, ainsi qu'utiliser SOAP et XML [81] dans la primitive de requête Req(SPARQL). Mais notre vision s'attache plus à représenter l'information de contexte que les fonctionnalités d'un service contextuel. On verra dans le chapitre 4, l'intérêt d'utiliser RDF. D'autre part, ces protocoles sont lourds et ne répondent pas à la minimalité voulue.

Ces primitives permettent d'abord d'identifier les différents dispositifs présents dans l'environnement. L'identificateur permettra ensuite de s'adresser au dispositif identifié. Mais il peut aussi permettre de le caractériser plus finement. Si cet identificateur est un URI, une application pourra retrouver sur le réseau une description de l'objet ainsi identifié. La seconde primitive permet de caractériser l'objet identifié comme le membre d'une classe particulière d'une ontologie. En principe, cette classe devrait être accessible à partir du réseau et trouver sa définition devrait permettre d'obtenir une description détaillée du dispositif identifié : on peut s'attendre, par exemple, à ce que les fabricants de ces dispositifs publient leurs caractéristiques (et en particulier les requêtes que l'on peut leur adresser) sur le réseau. Les deux premières primitives sont les deux seules primitives obligatoires. On peut noter qu'elles n'ont rien d'extravagant : elles sont déjà la base des protocoles RFID/EPC qui permet d'obtenir des informations détaillées à partir d'une étiquette RFID. Cependant, pour les appareils fonctionnant en mode purement local, c'est-à-dire sans avoir accès à Internet, il est bon de pouvoir fournir sa description (ou plutôt celles des informations de contexte qu'il est possible d'obtenir). La primitive `Desc` permet d'obtenir cette information dans un langage du type OWL. La quatrième primitive permet de poser des requêtes auxquelles le dispositif sait répondre et d'obtenir les informations nécessaires. Ainsi, tout dispositif pourra :

1. rechercher dans son environnement des services capables de lui donner les informations de son contexte ;
2. obtenir les caractéristiques des services retournés (par exemple, la précision d'une mesure) ;
3. s'adresser au service choisi pour obtenir l'information.

L'intérêt de cette approche est de ne pas avoir à modifier les dispositifs pour de nouvelles applications et de ne pas avoir à modifier les applications pour tirer parti de nouveaux dispositifs. En éxecutant le protocole, les applications découvriront les nouveaux dispositifs ainsi que les informations de contexte qu'ils produisent. Pour interagir et obtenir des informations sur l'environnement, ils doivent uniquement implémenter le protocole. Nous verrons plus tard que nous fournissons cette implémentation.

3.2.2 Exemple

Si l'on revient à l'appareil téléphonique de l'exemple, il recherchera dans l'environnement les services disponibles :

Il enverra la primitive `Id()` à tous les dispositifs déclarés comme producteurs d'informations de contexte auprès de l'annuaire local.

Deux services retourneront leur URI : `http://panasonic.com/3d23455651p/127765543`, `http://legrand.com/sdh/com/jsd67898200`.

Le premier peut être aisément identifié sur le réseau comme étant un téléviseur de la classe `http://panasonic.com/3d23455651p/`. Il permet aux applications de savoir son état et de le piloter.

Le second est la centrale de climatisation identifiable en l'interrogeant : `Cl(http://legrand.com/sdh/com/jsd67898200)`

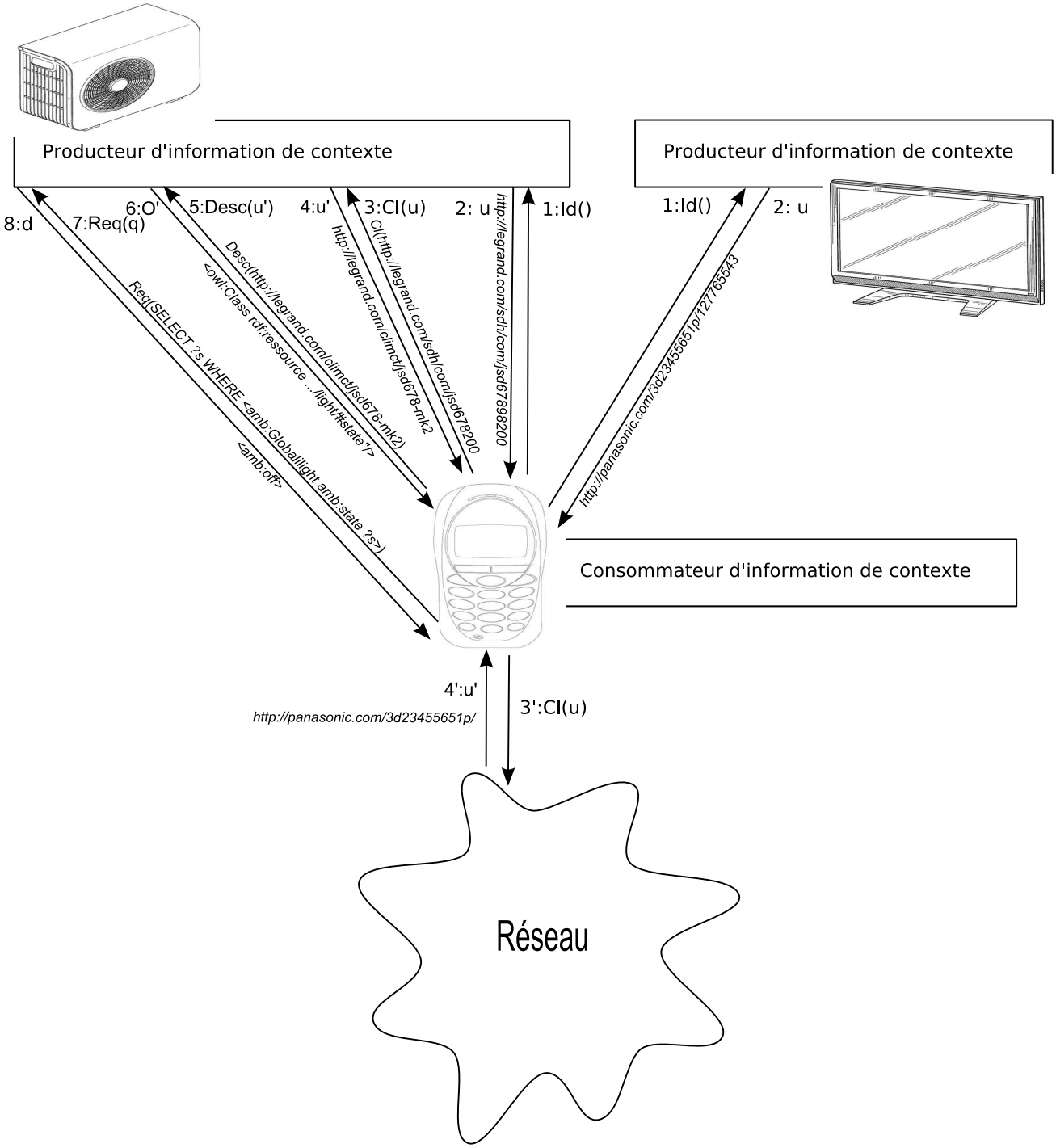


FIGURE 21 – Interaction. Les mentions en italiques dénotent des exemples

Il répondra par `http://legrand.com/climctl/jsd678-mk2`. La centrale de climatisation est capable de donner des informations de haut niveau correspondant à sa fonction (comme l'inutilité de rafraîchir une pièce inoccupée).

La centrale de climatisation est aussi disposée à communiquer de l'information que ses capteurs ont

agrégée. En particulier, elle est capable d'identifier que toutes les lumières sont éteintes. Par exemple, l'appareil télephonique cherchera à obtenir la description des informations de contexte produites en envoyant :

`Desc(http://legrand.com/climctl/jsd678-mk2)`

La centrale de climatisation retournera :

`<owl:Class rdf:resource="http://ambient.org/light/#GlobalLight"/>`,
`<owl:DatatypeProperty rdf:resource="http://ambient.org/light/#state"/>`.

Ainsi l'appareil télephonique posera la requête SPARQL :

`Req(SELECT ?s WHERE <amb:GlobalLight amb:state ?s>)`.

Ce à quoi la centrale de climatisation répondra par :

`{ <amb:off>}`.

La réponse indique que toutes les lumières sont éteintes, l'utilisateur dort vraisemblablement, il est inutile donc de le réveiller. Il est intéressant de noter que dans ce cas précis le consommateur et les producteurs d'informations de contexte utilisent la même ontologie `(http://ambient.org/light)` pour représenter leurs modèles de contexte. La requête du consommateur exprimée selon son modèle de contexte pourra trouver des réponses dans le modèle du producteur.

3.3 Bilan

Dans cette section, nous avons exposé une infrastructure de gestion pour les informations de contexte dans les environnements d'intelligence ambiante de telle sorte que chacun des dispositifs, capteurs et applications identifiés comme producteurs ou consommateurs d'informations de contexte puissent s'y intégrer facilement, en profiter et ainsi diffuser les informations qu'ils ont recueillies dans l'environnement ou y trouver celles dont ils ont besoin.

Les travaux de modélisation et de définition du contexte qui l'associent essentiellement à la situation ou à l'activité de l'utilisateur dans l'environnement, nous ont guidé dans nos choix architecturaux. Comme défini dans la section 1.3, nous modélisons le contexte comme l'ensemble des informations de contexte qu'il soit produit ou recherché par un service, une application ou un capteur. Chaque acteur de l'environnement évolue dans son propre contexte. Cette reflexion, nous a amené à une architecture dédiée à la gestion des informations de contexte décentralisée où, dans le souci de supporter la dynamicité et l'ouverture de l'environnement, chaque agent informatique de l'environnement embarque une bibliothèque dédiée à la gestion, la diffusion et la recherche de ses propres informations de contexte, ainsi que sa propre représentation du contexte.

Nous avons défini un protocole qui permet à chaque consommateur d'information de contexte de découvrir quand il le souhaite les producteurs d'informations de contexte présents dans l'environnement, puis d'obtenir auprès d'eux des informations sur l'environnement à tout instant. En outre, la distribution de la gestion des informations de contexte dans chacun des dispositifs, applications ou capteurs et la représention des informations de contexte libre de toute contraintes dans les ontologies utilisées, leur permet d'entrer dans l'environnemment sans risquer d'interférer avec le bon fonctionnement de l'environnement mais simplement de l'enrichir.

La proposition d'architecture telle qu'elle a été exposée dans cette section atteint les objectifs de minimalité notamment grâce à l'utilisation des technologies standards du web sémantique et du développement d'un protocole simple et facilement implémentable en utilisant des technologies différentes. Nous avons développé un canevas de développement Java sous forme d'interface que nous avons implémentée dans une solution multi-agents (voir section 5). Cependant toutes les autres implémentations du

canevas Java ou plus généralement du protocole assure à un dispositif ou à une application de profiter de l'infrastructure.

Nous n'avons pas encore montré comment notre infrastructure atteint l'objectif d'ouverture qui caractérise la faculté d'accueillir tous les capteurs, dispositifs et applications sans avoir de connaissance a priori sur leurs besoins ou leurs capacités en termes d'information de contexte. Dans le chapitre suivant, nous exposons plus précisement notre modélisation des informations de contexte à partir d'ontologies hétérogènes en utilisant OWL et RDF. Nous détaillons également les technologies du web sémantique qui nous permettent de réconcilier des ontologies et de trouver des correspondances entre ces modèle de contexte développés potententiellement indépendamment.

Chapitre 4

Représentation des informations de contexte dans un monde ouvert

Sommaire

4.1 Modèle pour l'information de contexte 75
4.1.1 Modèle de contexte 76
4.1.2 Exemple 76
4.2 Modèle de contexte extensible 76
4.2.1 Utilité des ontologies 77
4.2.2 Ontologies génériques 77
4.2.3 Exemple 78
4.3 Exploiter des ressources hétérogènes 79
4.3.1 Service d'alignement d'ontologies 80
4.3.2 Exemple 81
4.4 Bilan 82

Dans notre architecture, les acteurs informatiques de l'environnement gèrent leurs informations de contexte et exhibent sur le réseau un modèle de ces informations. Ils évoluent chacun dans un contexte distinct défini comme l'ensemble des informations utiles à leur exécution, incluant aussi les informations qui caractérisent l'environnement et qu'ils peuvent fournir à d'autres acteurs. Nous ne souhaitons pas les restreindre dans l'expression de ces modèles de contexte. Pour que les capteurs, dispositifs et applications puissent exprimer la connaissance de contexte selon leur propre vision du monde, ils utiliseront les ontologies de leurs choix pour décrire leurs informations de contexte. Dans une première partie de ce chapitre, nous détaillerons comment cette modélisation du contexte sera implémentée et utilisée par la bibliothèque de gestion d'information de contexte et quels en sont les avantages. Cependant, cette seule modélisation ne suffit pas pour que l'infrastructure accueille des capteurs, dispositifs et applications hétérogènes, car il n'existe pas de correspondance a priori entre leurs différents modèles assurant leur compatibilité. Dans la deuxième partie de ce chapitre, nous présentons la technologie d'alignement d'ontologies qui permet de trouver des correspondances entre deux ontologies, et la mise en place de cette technologie dans notre infrastructure.

4.1 Modèle pour l'information de contexte

Nous introduisons ici le modèle simple que nous allons utiliser pour représenter les informations de contexte nécessaires aux applications d'informatique diffuse. Ces modèles se basent sur les langages du

web sémantique (voir section 2.4.1). Leur utilisation dans le contexte des applications qui nous occupent sera justifié par la suite (voir section 4.2).

4.1.1 Modèle de contexte

Le modèle de contexte utilisé à ce stade est très simple : un contexte sera un ensemble d'assertions RDF. Les éléments de contexte seront donc considérés comme du RDF. L'avantage d'utiliser un langage de la généralité de RDF est que l'on peut exploiter une interface de très haut niveau et que les informations de contexte stockées et communiquées peuvent être simplement considérées comme des triplets. L'interopérabilité est alors simplement garantie en considérant que les composants gérant le contexte sont des consommateurs et des producteurs de RDF. Ceci n'est cependant pas suffisamment précis et il est souhaitable de n'extraire que l'information utile des différentes sources d'informations de contexte disponibles. Pour cela on pourra utiliser un langage comme RDQL [74] ou SPARQL [64] permettant de poser (ou de s'abonner à) des requêtes précises aux composants. Pour pouvoir poser les bonnes requêtes aux bons composants, il est nécessaire pour les composants de rendre public les requêtes auxquelles ils sont capables de répondre. Cela peut être réalisé par la publication des classes d'objets et les propriétés sur lesquelles le composant peut répondre. Le langage OWL est tout à fait adapté pour spécifier (par le biais d'une ontologie) ces éléments. On ne considérera pas, à ce stade, de réification des contextes ni d'opérations de manipulation de contextes (mais ceci devra pouvoir s'insérer ultérieurement dans l'architecture proposée). En l'état, ces manipulations seront prises en charge par l'architecture qui permettra aux dispositifs d'acquérir leur contexte en combinant ceux des autres dispositifs.

4.1.2 Exemple

Une partie des informations concernant la chambre d'hôtel, fournies par le téléviseur, peut être représentée par l'ensemble de triplet suivant :

```
⟨self   ami:location  #1345⟩
⟨#1345    rdf:type    ami:HotelRoom⟩
⟨#67   ami:location   #1345⟩
⟨#67     rdf:type     tv:tvset⟩
⟨#67     tv:state     tv:on⟩
⟨#67    tv:volume     tv:muted⟩
⟨#67   tv:channel     tv:arte⟩
```

Ils signifient que l'application est localisée (sémantiquement) dans une chambre d'hôtel contenant par ailleurs un poste de télévision diffusant Arte avec le volume coupé. On remarque le typage des entités (tvset, HotelRoom) ainsi que l'assemblage de vocabulaires utilisant des espaces de nom différents (rdf, ami, tv).

4.2 Modèle de contexte extensible

Si l'on peut ajouter des composants à tout moment, il n'est pas clair cependant qu'ils soient si facilement exploitables. Il n'y a pas de raison, a priori, que les composants proposés ainsi que les nouvelles applications soient réellement interopérables. En effet, chaque nouveau capteur fournira plus de précision ou des informations qui ne peuvent avoir été complètement prévues à l'origine. De même, les applications ne peuvent connaître tous les types de capteurs disponibles. On montre comment exploiter les langages de description d'ontologies pour résoudre ce problème.

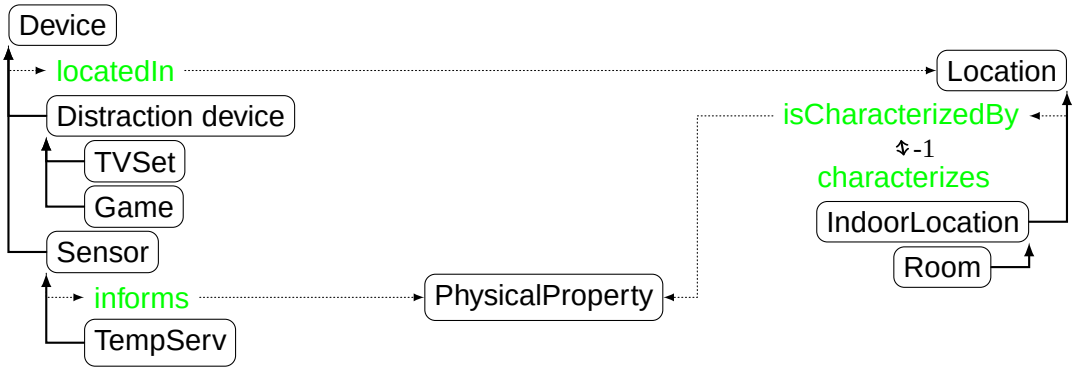


FIGURE 22 – Exemple d'ontologie de concepts d'ordre général (les classes sont représentées par des rectangles arrondis, les propriétés sont représentées par des flèches pointillées et les flèches pleines entre deux classes représentent des relations de sous-classes). Cette ontologie fournit des classes pour les dispositifs et la localisation ainsi que des propriétés comme "locatedIn".

4.2.1 Utilité des ontologies

Les techniques de représentation de connaissance que l'on trouve transposées dans le langage OWL permettent de toujours préciser un concept ou une propriété sans avoir à remettre en cause ceux qui existaient au préalable. Ainsi, il n'est nullement nécessaire, pour accomplir sa mission de savoir que le téléviseur est un téléviseur 16/9 avec disque dur intégré si l'application veut simplement savoir si l'utilisateur est occupé à regarder la télévision (ou s'il y a une activité dans la pièce). Mais rien ne doit non plus empêcher le fabricant de l'appareil de le déclarer ainsi. Il fera cela en caractérisant précisément le téléviseur produit dans l'ontologie des appareils de télévision, des dispositifs de distraction et des appareils consommateurs d'électricité. Dans l'application donnée en exemple, l'important est que la classe du téléviseur soit une sous-classe de "dispositifs de distraction" pour que l'application en déduise une activité. Par contre, si l'application a pour but de trouver rapidement une chaîne pour que l'utilisateur ne manque pas son émission préférée, alors il est important de connaître les capacités de l'appareil et de l'interroger sur ce qu'il est à même de réceptionner.

Ainsi, l'utilisation d'ontologies pour caractériser les situations, permet d'accueillir de nouveaux appareils dont les possibilités ne peuvent être connues au départ et des nouvelles applications tirant parti de ces possibilités. Les applications ont intérêt à être le plus générales possible sur les informations dont elles ont besoin (la température de la pièce, l'activité d'un dispositif de distraction) alors que les systèmes de gestion de contexte doivent être le plus précis possible sur ce qu'ils produisent. Cela permettra aux applications les plus pointues d'en tirer le meilleur parti.

4.2.2 Ontologies génériques

L'essentiel est de disposer d'ontologies suffisamment génériques pour pouvoir prendre sous leurs chapes les différentes notions impliquées dans les applications d'informatique diffuse : ressources, acteurs, lieux, dates, activités, permissions, etc. Il existe plusieurs ontologies de ce type (par exemple [20, 82]). On verra dans la section suivante comment réconcilier ces différentes ontologies. Ces ontologies sont en général peu profondes car elles ont été conçues pour faire fonctionner la machinerie des applications d'informatique diffuse. Pour chacun des domaines, il est nécessaire de développer une description plus précise des informations exploitables. Ainsi, [40] propose une description sémantique de quatre modèles différents (sémantique, géométrique, théorie des graphes et modèles ensemblistes) pour

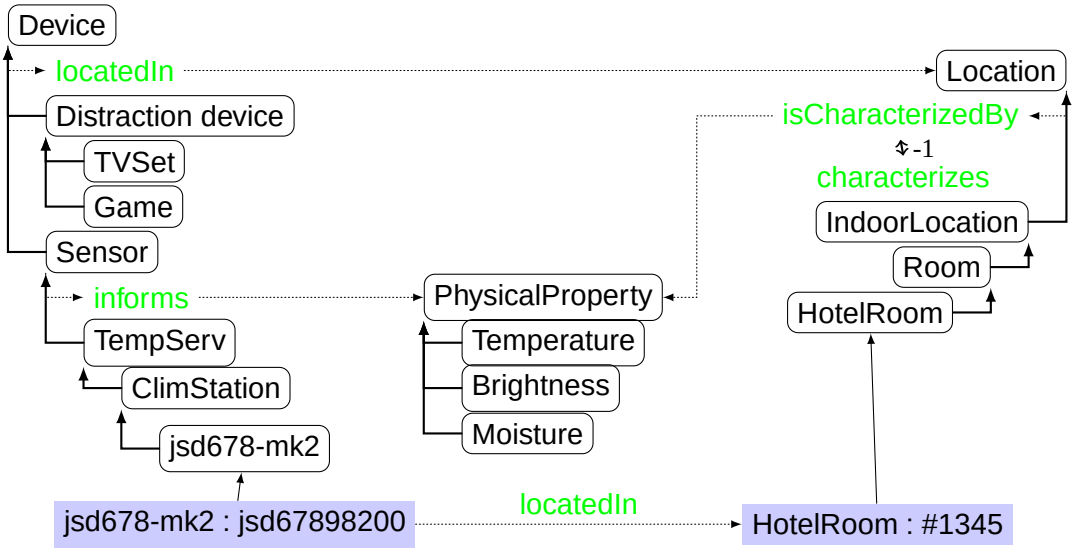


FIGURE 23 – Ontologies spécifiques aux objets de l'environnement. Elles raffinent l'ontologie générale de la figure 22 avec de nouveaux concepts (les instances sont représentées par des rectangles gris et liés à leur classe par des fines flèches).

représenter des informations de localisation. Ces informations peuvent être représentées selon un modèle ou un autre, et le système saura trouver les correspondances entre les différents modèles. Nous développons aussi des modèles des propriétés physiques de l'environnement accessibles aux applications. De tels modèles peuvent être utilisés par les applications pour caractériser les informations qui leur sont nécessaires. Les dispositifs, pour leur part, utiliseront les caractérisations les plus précises possibles pour pouvoir être exploitées par les applications.

4.2.3 Exemple

Poursuivons l'exemple précédent en supposant que l'application désire connaître la température dans la pièce. L'ontologie de haut niveau (voir figure 23) lui permet de caractériser son besoin : la température est une propriété physique de la pièce, cette propriété peut être obtenue par un capteur de type TempServ, comme un thermomètre, situé dans la même pièce. L'information nécessaire est exprimée par le graphe de la figure 24. Ce graphe correspond à un ensemble de triplets RDF (et ainsi à une requête à poser à l'environnement).

La figure 25 présente l'information utilisée pour trouver un dispositif capable de donner l'information de contexte correspondant à la température de la pièce. La température elle-même sera obtenue avec le protocole présenté précédemment. Comme on peut le constater, les informations sont décrites beaucoup plus précisément que dans la requête. La pièce (concept provenant d'un modèle sémantique de localisation [40]) est devenue une chambre d'hôtel. Quant au capteur de température, il est la centrale de climatisation qui, disposée à fournir cette information, est décrite comme un serveur de température (elle peut, bien entendu, être décrite comme beaucoup d'autres choses).

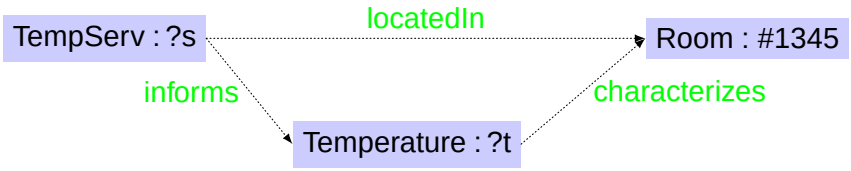


FIGURE 24 – Le graphe requête correspondant à l'information nécessaire à une application qui recherche dans l'environnement la température de la pièce. Les points d'interrogation à la place des instances introduisent des variables libres qui correspondent aux informations que les applications recherchent dans l'environnement (la température dans la pièce). Ces informations sont exprimées en fonction des ontologies spécifiques de la figure 23.

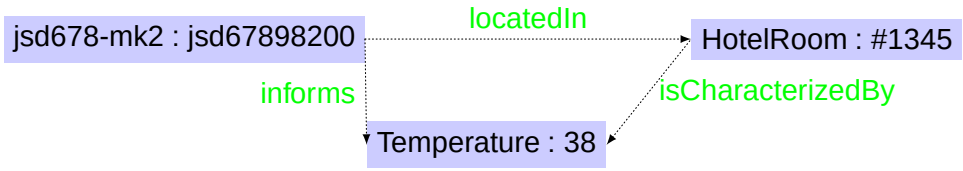


FIGURE 25 – Le raffinement et l'instanciation de l'ontologie entrent effectivement en correspondance avec les besoins de l'application exprimés à la figure 24. Ces informations fournies par les producteurs d'informations de contexte ne sont pas des instanciations du graphe de requête de la figure 24, mais en utilisant les ontologies de la figure 22 et de la figure 23 elles répondent à la requête.

4.3 Exploiter des ressources hétérogènes

Le système de gestion de contexte proposé permet d'introduire dans l'environnement de nouveaux dispositifs en étendant l'ontologie de telle sorte que les anciennes applications puissent en tirer parti. Ceci fonctionne si les acteurs ont pour référence les mêmes bases ontologiques. Il n'y a pas vraiment de raison pour cela. Au contraire, deux fabricants de capteurs peuvent avoir de légitimes raisons d'utiliser des ontologies différentes pour décrire leurs composants. Il n'est par ailleurs pas concevable de supposer une ontologie unique dans laquelle tout est intégré. Ceci entraine un problème lorsqu'il s'agit d'apparier les contextes produits et les informations de contexte nécessaires à une application.

[22] offre trois ontologies de bases découpées selon les concepts "Où", "Qui" et "Quand" pour décrire les informations de contexte partagées entre les applications et pour assurer l'interopérabilité. Les applications peuvent étendre les trois ontologies de base pour modéliser leur propre contexte, mais l'interopérabilité entre leurs ontologies s'appuiera sur les relations existantes entre les trois ontologies fournies. On peut également citer GUMO [46] qui fournit aussi des ontologies pour les environnement d'intelligence ambiante. [1] développe une ontologie de contexte en OWL où le contexte est composé des centres d'intérêt de l'utilisateur. Elle peut être utilisée pour adapter la présentation d'une page web par exemple. Le W3C à mis au point une recommandation [15] pour unifier les descriptions sémantiques devant la diversité des dispositifs utilisés dans les environnement d'intelligence ambiante. L'existence d'un grand nombre d'ontologies qui se rapporte à notre domaine montre bien le besoin grandissant d'un moyen pour réconcilier ces ontologies.

Trois approches sont possibles pour assurer l'interopérabilité dans les environnements d'informatique diffuse :

15. http ://www.w3.org/TR/2008/PR-DDR-Simple-API-20080917/

– La standardisation a priori des ontologies utilisées : comme indiqué plus haut, ceci est difficile à obtenir, par ailleurs ce n'est pas souhaitable car cela mettrait un frein à l'évolution nécessaire des ontologies et de la technologie ;
– La maintenance de médiateurs entre ontologies : que ce soit sur un mode distribué ou centralisé [86], on peut imaginer que les correspondances entre ontologies puissent être librement mises à disposition (que ce soit par les fabricants de capteurs, les développeurs d'application ou les utilisateurs eux-mêmes) et que les applications puissent les exploiter si elles doivent mettre en correspondance deux ontologies ;
– Un service d'alignement d'ontologie : il permet lorsque le besoin s'en fait sentir de trouver la correspondance entre deux ontologies.

Ces trois solutions ne sont pas incompatibles et peuvent fonctionner de concert. S'accorder sur les ontologies de haut niveau est souhaitable et possible et laisser les niveaux plus spécifiques se développer librement permet de prendre en compte rapidement l'évolution d'un domaine.

4.3.1 Service d'alignement d'ontologies

Nous préconisons la mise en œuvre d'un (ou de plusieurs) service(s) d'alignement d'ontologies. Le rôle d'un tel service est d'aider des agents (en l'occurrence ici les gestionnaires de contexte) à trouver les correspondances entre les différentes ontologies auxquelles ils sont confrontés. Ces services sont à même d'offrir les opérations suivantes :
– Trouver une ontologie proche d'une ontologie particulière ;
– Stocker (et retrouver) des alignements déjà obtenus entre différentes ontologies ;
– Calculer dynamiquement la correspondance entre deux ontologies ;
– Traduire les requêtes et réponses aux requêtes vers des gestionnaires de contexte utilisant des ontologies différentes.

De même que les ontologies, les services d'alignement devront être accessibles aux applications et aux gestionnaires de contexte par le biais du réseau. Les services d'alignement utilisent et adaptent à leurs propres besoins une interface fonctionnelle permettant de manipuler explicitement les alignements entre ontologies développées dans le contexte du web sémantique [35].

Le protocole final qui inclut les étapes d'alignement d'ontologie est :
– `Id()`$rightarrow$`URI` : l'identificateur du service fourni ;
– `Cl(URI)`$rightarrow$`URI` : la classe du service identifié ;
– `Desc(URI)`$rightarrow$`OWL` : la description de l'information que le composant peut fournir : $Desc(u) = O'$
– `Req(SPARQL)`$rightarrow$affectations `RDF` : les tuples des valeurs de variables qui satisfont la requête : $Req(q) = S_1, \ldots, S_n$
– `Align(`URI_1`, `URI_2`)`$rightarrow$`A` : À la demande d'alignement entre deux ontologies, le serveur répond par l'identificateur d'un alignement.
– `Translate(A,Req)`$rightarrow$`Req` : Le service client, consommateur d'information de contexte, demande au serveur d'alignement la traduction de la requête qu'il souhaite envoyer au consommateur d'information de contexte. Il demande la traduction d'une requête exprimée selon son propre modèle en fonction de l'alignement obtenu à l'étape précédente, c'est-à-dire l'ensemble des correspondances entre son modèle d'information de contexte et celui du producteur d'information ciblé. La réponse du serveur est une traduction de la requête, passée en paramètre, exprimée selon le modèle du producteur d'information de contexte.
– `Translate(inv(A),Rep)`$rightarrow$`Rep` : De la même façon que décrit ci-dessus, un service demande la traduction, vers son propre modèle d'information de contexte, de la réponse qu'il reçoit d'un producteur en l'associant à l'inverse de l'alignement précédent.

Serveur d'alignement Un serveur d'alignement s est caractérisé par une fonction $match_c : O \times O \rightarrow A$.

Situation À un instant donné, notre système est dans une $\langle C, S \rangle$ caractérisée par un ensemble de producteurs de contexte C et un ensemble de serveurs d'alignement S.

Réponse à une requête d'application Soit une requête de contexte q d'une application dans une situation $\langle C, S \rangle$, l'ensemble de réponses à q est caractérisé par :

La définition (procédurale) ci-dessous suppose que les requêtes sont le plus souvent élémentaires et qu'elles peuvent être composées au niveau de leurs réponses. Ainsi, il est possible de :

1. pour chaque $e \in \sigma(q)$ si $\exists c \in C$ tel que $e \in Desc_c$ alors $q := q[e/Req_c(e)]$
2. pour chaque $e \in \sigma(q)$ si $\exists c \in C$ tel que $e' \in Desc_c$ tel que $\exists s \in S$ tel que $\langle e, e' \rangle \in match_s(o, o')$ alors $q := q[e/Req_c(e')]$
3. si $\sigma(q) = \emptyset$ alors $eval(q)$ sinon échec

Dans ce mémoire, l'opération τ (pour `Translate`) permet d'obtenir la traduction d'une requête en regard d'un alignement. Ainsi, l'action n°2 peut s'écrire $q := \tau(match_s(o, o'), q)$. Elle est aussi utilisée pour convertir des réponses et donc toutes les correspondances doivent être unifiées dans un alignement a et les résultats devront être $\tau(a^{-1}, eval(q))$ (mais c'est encore mieux de faire $\tau(a^{-1}, eval(\tau(a, q)))$.

Étant donné cet algorithme, on peut attendre deux résultats théoriques compte tenu de ces définitions :

correction qu'il fournisse une réponse valide ;

complétude que s'il existe une réponse, il la trouvera.

Il convient également de noter que si plusieurs réponses existent, alors une seule sera trouvée.

4.3.2 Exemple

Considérons le même exemple que précédemment avec une centrale de climatisation utilisant une ontologie particulière du domaine de la climatisation (`http://clim.org/ontologies/devices.owl/v1.1`) et l'appareil téléphonique utilisant une ontologie modélisant les situations dans lesquelles se trouve un humain (`http://psycho.illinois.edu/~andersen/attitudes/onto.rdf`). On comprend bien pourquoi, a priori, ces deux modèles du monde ont peu de raisons d'être les mêmes. Ceci dit, notre téléphone a juste besoin de savoir si une personne est trop occupée pour pouvoir la déranger. Comme on trouve des climatisations en de nombreux endroits et des appareils téléphoniques dans presque toutes les poches, il y a de fortes chances que ces deux ontologies aient déjà été mises en correspondance. Cette fois-ci, l'appareil téléphonique va pouvoir s'adresser au service d'alignement en ces termes :

```
Align( http://clim.org/ontologies/devices.owl/v1.1,
http://psycho.illinois.edu/~andersen/attitudes/onto.rdf)
```

Le service d'alignement lui envoie alors l'identificateur de l'alignement entre ces deux ontologies : $\longrightarrow$ `al23889`

À noter qu'ici, le service répond directement par un alignement sans que l'on sache s'il a été calculé ou stocké. Le protocole permet de spécifier les paramètres, le type d'algorithme à utiliser ou s'il faut chercher en priorité l'alignement dans la bibliothèque. La réponse retournée n'est pas l'alignement mais un identificateur permettant au service de le retrouver si nécessaire. L'application peut demander à traduire la requête à poser au gestionnaire de contexte de la centrale de climatisation.

```
Translate(al24889, "SELECT ?s WHERE <psy:room psy:lighting ?s>")
```

⟶ SELECT ?s WHERE <amb:GlobalLight amb:state ?s>

Il pourra poser la requête retournée à la centrale et demander l'interprétation du résultat :

Translate(inv(al24889), { <amb:off> })

⟶ { <psy:dark> }

Alignement entre l'ontologie clim.org et l'ontologie human situation au format XML
La représentation graphique de cet alignement peut se voir sur la figure 27.

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<rdf:RDF xmlns="http://knowledgeweb.semanticweb.org/heterogeneity/alignment"
       xml:base="http://knowledgeweb.semanticweb.org/heterogeneity/alignment"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
<Alignment>
  <xml>yes</xml>
  <level>0</level>
  <type>**</type>
  <onto1>http://clim.org/ontologies/devices.owl/v1.1</onto1>
  <onto2>http://psycho.univ.edu/attitudes/onto.rdf</onto2>
  <uri1>http://clim.org/ontologies/devices.owl/v1.1</uri1>
  <uri2>http://psycho.univ.edu/attitudes/onto.rdf</uri2>
  <map>
    <Cell>
      <entity1 rdf:resource="http://clim.org/ontologies/devices.owl/v1.1l#GlobalLight"/>
      <entity2 rdf:resource="http://psycho.univ.edu/attitudes/onto.rdf#Lighting"/>
      <relation>=</relation>
      <measure rdf:datatype="http://www.w3.org/2001/XMLSchema#float">1.0</measure>
    </Cell>
    <Cell>
      <entity1 rdf:resource="http://clim.org/ontologies/devices.owl/v1.1l#Temperature"/>
      <entity2 rdf:resource=http://psycho.univ.edu/attitudes/onto.rdf#AverageTemperature"/>
      <relation>=</relation>
      <measure rdf:datatype="http://www.w3.org/2001/XMLSchema#float">1.0</measure>
    </Cell>
  </map>
</Alignment>
</rdf:RDF>
```

4.4 Bilan

Acheminer un flot d'informations caractérisant l'environnement des producteurs d'informations de contexte hétérogènes vers des consommateurs d'informations de contexte hétérogènes dans un environnement d'intelligence ambiante est le problème que nous avons résolu. Notre approche s'intéresse spécifiquement aux environnements ouverts où les applications et les dispositifs doivent pouvoir entrer ou sortir sans en altérer le fonctionnement. Ils doivent pouvoir profiter les uns des autres, être interopérables, de façon dynamique. Enfin, pour que cette solution soit adoptable par les industriels puis utilisée dans notre quotidien, il faut faciliter son utilisation et la rendre accessible aux plus grand nombre avec le minimum d'efforts à fournir.

Le chapitre 3 propose une infrastructure où chaque dispositif ou application qu'il soit consommateur ou producteur d'informations de contexte embarque un composant de gestion de l'information de contexte, ou bien, s'il ne possède pas les ressources informatiques pour embarquer ce composant, ce

dernier est embarqué dans un proxy qui le représente sur le réseau. Ce composant lui permet dans un premier temps de représenter et de manipuler l'information de contexte qu'il produit ou qu'il recherche en utilisant un modèle écrit en OWL et RDF (voir chapitre 4). En utilisant un protocole prédéfini par notre modèle, il peut identifier un dispositif ou une application, obtenir sa description sous forme de classe OWL et lui envoyer des requêtes SPARQL.

L'infrastructure proposée, par sa conception sous forme de bibliothèque implémentable dans chacun des acteurs sensibles au contexte, par l'utilisation d'un protocole restreint et de standards pour représenter les informations de contexte, atteint l'objectif de minimalité. Son utilisation n'oblige pas les fabricants à utiliser des technologies propriétaires, mais elle contraint seulement les acteurs d'un environnement d'informatique diffuse à embarquer une implémentation du protocole défini dans les sections 3.2 et 4.3 ainsi qu'à l'utilisation des standards du web sémantique, ce qui en fait une solution facile à adopter. D'autre part, le fonctionnement du protocole défini et le choix d'une architecture distribuée assurent le dynamisme de cette infrastructure. Les producteurs d'informations de contexte participent à l'infrastructure de gestion de contexte sans autre effort que l'enregistrement auprès de l'annuaire de service local, et les consommateurs d'informations de contexte doivent seulement, pour leur part, dérouler le protocole pour obtenir les informations de contexte auprès des producteurs enregistrés. Que ce soit dans le cas de l'apparition ou de la disparition d'un producteur d'information de contexte, le comportement de l'infrastructure n'est pas modifié et c'est au seul consommateur d'information de prendre la responsabilité de réarmer le protocole.

Pour être totalement ouvert et supporter l'hétérogénéité des environnements intelligents, nous avons pris le parti de ne pas restreindre les consommateurs et les producteurs dans leurs modélisations des informations de contexte. Ils peuvent utiliser le vocabulaire de n'importe quelle ontologie, pour peu qu'elle soit accessible par le réseau, pour décrire les informations de leur contexte (voir 4.2). Le but est qu'ils puissent modéliser le monde selon leurs propre vision du monde. Ainsi, le composant de gestion, dans le cas d'un consommateur d'information de contexte, utilise un serveur d'alignement d'ontologie pour obtenir les correspondances entre ses besoins et les capacités des consommateurs qu'il contacte. L'utilisation des technologies du web sémantique assurent que l'infrastructure supporte l'ouverture et la dynamicité des environnements d'intelligence ambiante.

Maintenant que nous avons dessiné un patron de construction pour réaliser une infrastructure qui atteint les objectifs de dynamicité, d'ouverture et de minimalité, il nous reste à savoir si elle peut être mise en œuvre de manière réaliste. Nous décrirons dans le chapitre suivant l'implémentation de ce patron, puis dans le dernier chapitre nous présenterons l'expérimentation de cette implémentation à travers la réalisation d'un démonstrateur d'environnement d'intelligence ambiante.

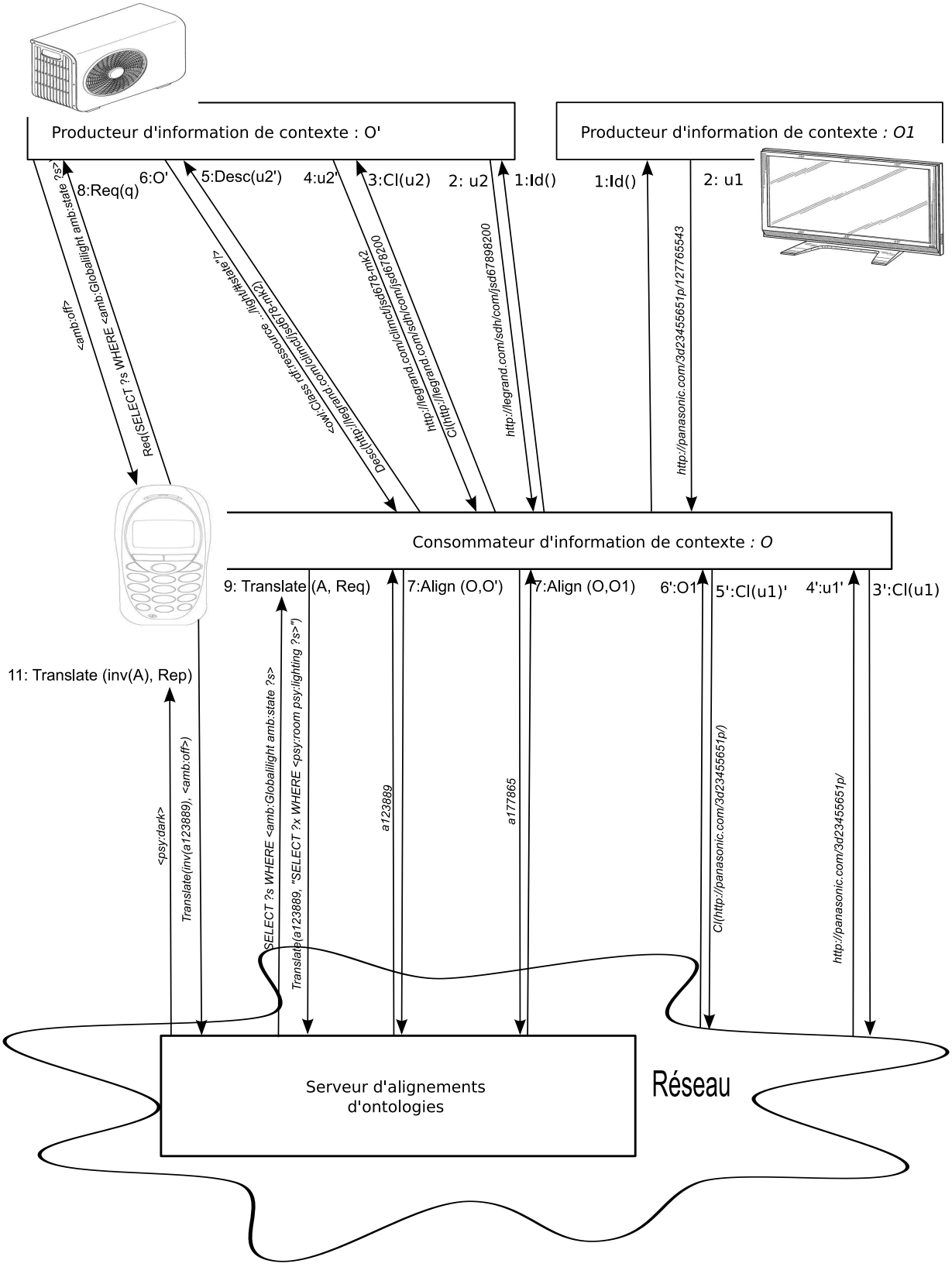


FIGURE 26 – Interaction. Les mentions en italiques dénotent des exemples.

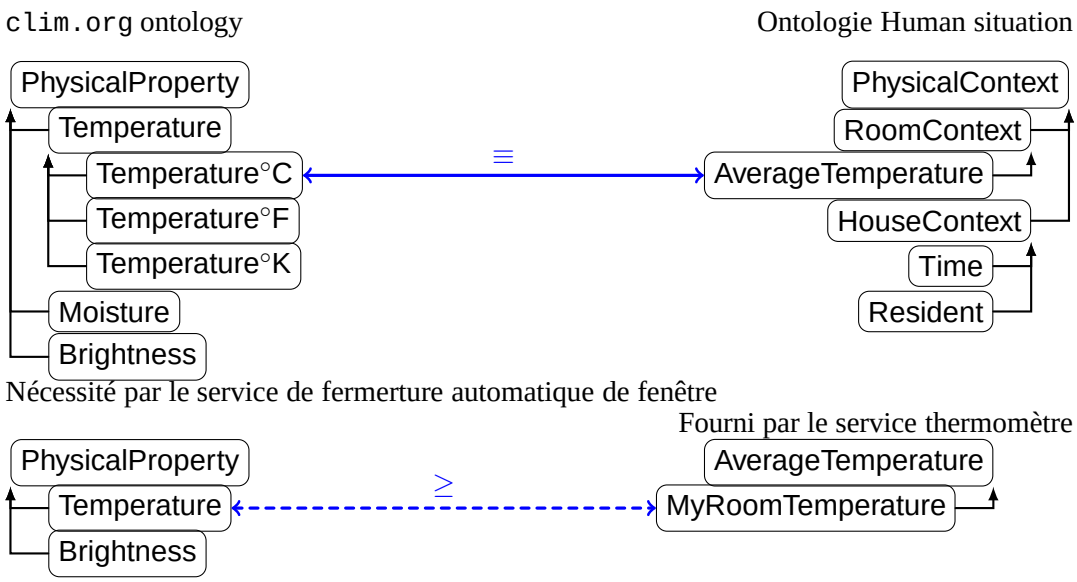


FIGURE 27 – Dans le but d'inférer des relations entre le modèle des informations de contexte dont il a besoin et le modèle des informations de contexte du producteur, le service de fermeture automatique de fenêtre demande au serveur d'alignement les correspondances entre son ontologie de base (`clim.org`) et une ontologie utilisée par le producteur d'information de contexte (l'ontologie Human situation). La relation d'équivalence retournée par le serveur d'alignement permet de déduire que la Température est plus générale que MyRoomTemperature (et donc que cette dernière peut être utilisée comme information de température).

Chapitre 5

Mise en œuvre

Sommaire

5.1 Infrastructure multi-agents 87
5.1.1 Une bibliothèque pour les applications et dispositifs sensibles au contexte 87
5.1.2 Implémentation multi-agent 88
5.2 Gestion des modèles sémantiques d'informations de contexte 91
5.2.1 Manipulation des modèles 91
5.2.2 Serveur d'alignement d'ontologies 92
5.3 Bilan 93

Le modèle conceptuel présenté précédemment est un guide pour remplir les critères d'ouverture, de dynamicité et de minimalité de notre infrastructure, mais il faut encore que les technologies d'implémentation mises en œuvre n'aillent pas à l'encontre de cette réalisation. L'utilisation d'un protocole simple combiné aux langages de représentation standardisés que sont OWL et RDF et de la technologie d'alignement d'ontologies permet d'assurer l'ouverture et la dynamicité de l'environnement. Le critère de minimalité doit assurer que le plus grand nombre de dispositifs et d'applications puissent profiter de cette infrastructure en imposant le minimum d'efforts à leurs développeurs pour s'y intégrer. Pour s'assurer que cet objectif sera atteint, nous fournissons à ces développeurs des interfaces de développement en Java aussi bien pour intégrer un composant consommateur qu'un producteur d'information de contexte. De plus, nous fournissons une implémentation de ces interfaces sous forme de système multi-agent. Nous présenterons ensuite les outils utilisés pour la manipulation des ontologies, de OWL et RDF, ainsi que l'utilisation d'un serveur d'alignement d'ontologies.

5.1 Infrastructure multi-agents

Nous présentons ici la bibliothèque permettant de créer des applications et dispositifs sensibles au contexte ainsi que son diagramme de classe. Nous la déclinerons dans une implémentation multi-agents qui utilise le système JADE.

5.1.1 Une bibliothèque pour les applications et dispositifs sensibles au contexte

Afin de proposer une plateforme interopérable aux dévelopeurs de dispositifs et d'applications sensibles au contexte, nous fournissons un ensemble d'interfaces Java qui définissent une API (voir la figure 28). Nous avons implémenté cette API de façon à ce qu'elle soit embarquée dans une application ou un dispositif.

En définissant des interfaces, nous permettons le développement d'autres implémentations de cette API en assurant l'interopérabilité entre elles. Mais, l'utilisation de ces interfaces assure que les dispositifs profiteront du protocole de découverte et de l'utilisation d'un modèle d'information de contexte sémantique. L'interface *ContextManager* décrit le composant embarqué dans une application sensible au contexte. Il est identifié par un URI et peut être instancié sous la forme d'un objet *ContextConsumer*, *ContextProducer*, ou des deux objets à la fois. Ces interfaces sont respectivement des consommateurs d'informations de contexte comme des applications intelligentes et des producteurs d'informations de contexte tels que des capteurs. Chacune de ces interfaces possède des méthodes pour manipuler les objets *ContextDescription* qui sont les descriptions sémantiques des informations de contexte recherchées ou produites. Les descriptions sont identifiées par des URI. Elles peuvent être mises à jour pendant l'exécution et une méthode est proposée pour trouver un ensemble de correspondances entre deux objets *ContextDescription*. Cette méthode retourne un objet *Alignment* que l'infrastructure obtient en envoyant une requête à un serveur d'alignement (voir la section 4.3.1).

Un *ContextConsumer* commence par demander l'alignement au serveur d'alignement entre sa propre ontologie, contenue dans un objet *ContextDescription*, et la description sémantique des informations de contexte d'un objet *ContextProducer* qu'il a obtenu après l'exécution du protocole (voir la section 3.2). Ensuite, en utilisant un objet *Alignment*, représentant l'ensemble des correspondances entre ces deux ontologies, le consommateur d'information de contexte décide s'il veut sélectionner ce producteur d'information de contexte. Puis, il peut créer un objet *QueryMediator* pour traduire les requêtes selon l'objet *Alignment*.

Les interfaces *ContextConsumer* et *ContextProducer* fournissent également des méthodes pour communiquer en utilisant une interface *Message* et pour exécuter le protocole. À chaque phase de protocole correspond une méthode afin d'envoyer le message correspondant (voir figure 29). De plus, nous fournissons des méthodes pour s'abonner et se désabonner à un annuaire local de service. L'interface *ContextConsumer* fournit des méthodes pour créer, gérer et mettre à jour une liste des instances *ContextProducer* connus, et une méthode doit être implémentée pour déterminer la compatibilité entre un objet *ContextConsumer* et un objet *ContextProducer*. Une méthode pour évaluer une requête SPARQL est nécessaire dans l'interface *ContextProducer* et elle doit retourner un objet *Result*.

5.1.2 Implémentation multi-agent

Notre infrastructure a été implémentée en utilisant la plateforme de dévelopement agent en Java, JADE [16] (Java Agent DEvelopment). Chaque *ContextManager* est un agent JADE et les interfaces *ContextConsumer* et *ContextProducer* sont implémentées sous forme de comportement d'agent qui sont des processus indépendants à l'intérieur d'un même agent. L'interface *Message* est implémentée en utilisant le système de messagerie de la plateforme JADE. Les messages échangés entre les agents peuvent ainsi être observés en utilisant l'utilitaire JADE sniffer (voir une photo d'écran de l'interface sur la figure 30).

Aujourd'hui l'infrastructure de gestion de l'information de contexte que nous avons réalisée est fonctionnelle, cependant il faut multiplier les tests dans des environnements variés pour assurer sa robustesse et prouver son passage à l'échelle.

Pour faciliter l'interopérabilité, une bonne chose serait de calculer des alignements entre les ontologies communément utilisées dans les environnements d'informatique diffuse telles que les ontologies du projet Amigo [17], [43] ou [77], ou des ontologies d'ordre plus général comme SUMO ou CYC, et de les publier sur le web. Ainsi, ces alignements seraient exploitables par un serveur d'alignement.

16. http ://jade.tilab.com/
17. http ://amigo.gforge.inria.fr/home/index.html

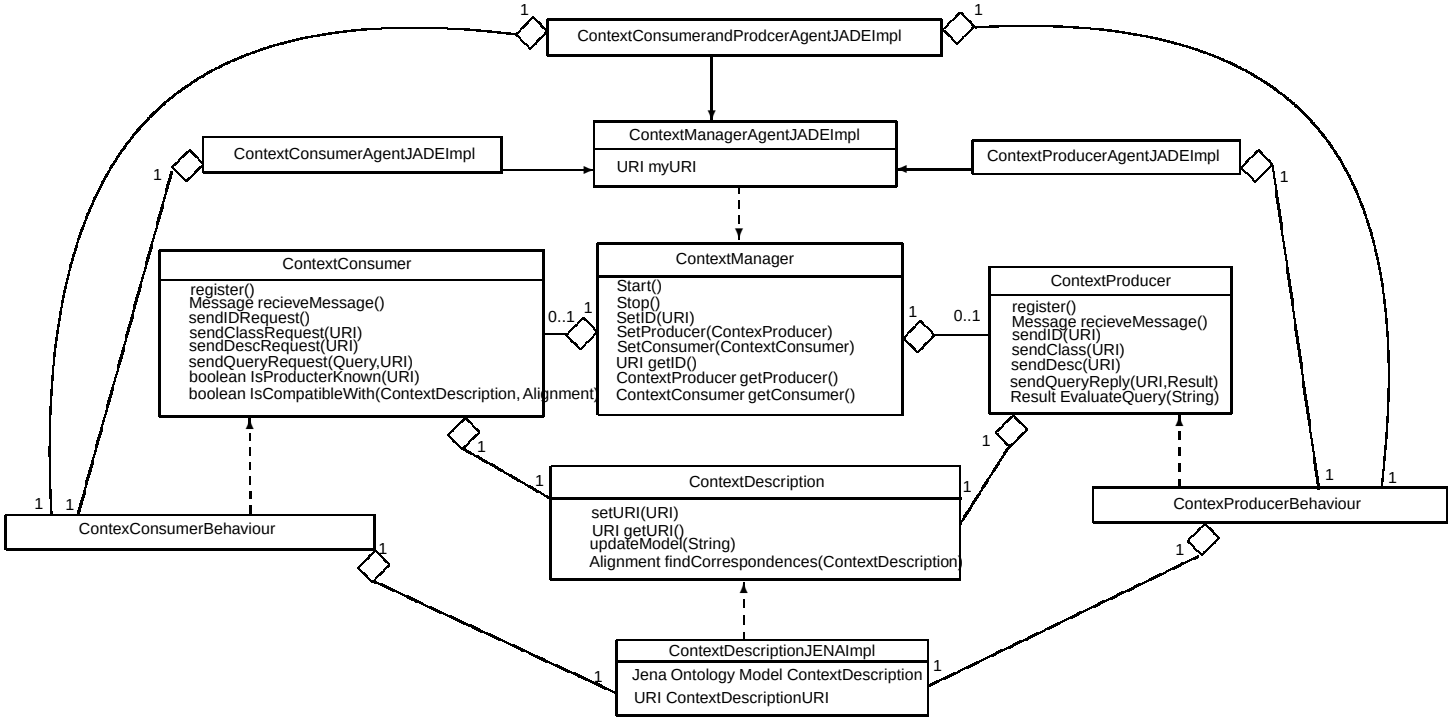


FIGURE 28 – Diagramme de classe des principales classes de l'API. Nous fournissons des implémentations simples d'agents producteurs, consommateur et producteur, et consommateurs, qui peuvent être embarquées dans un dispositif ou dans une application. Les lignes pleines correspondent à une *extension* et les lignes pointillées à une *implémentation*.

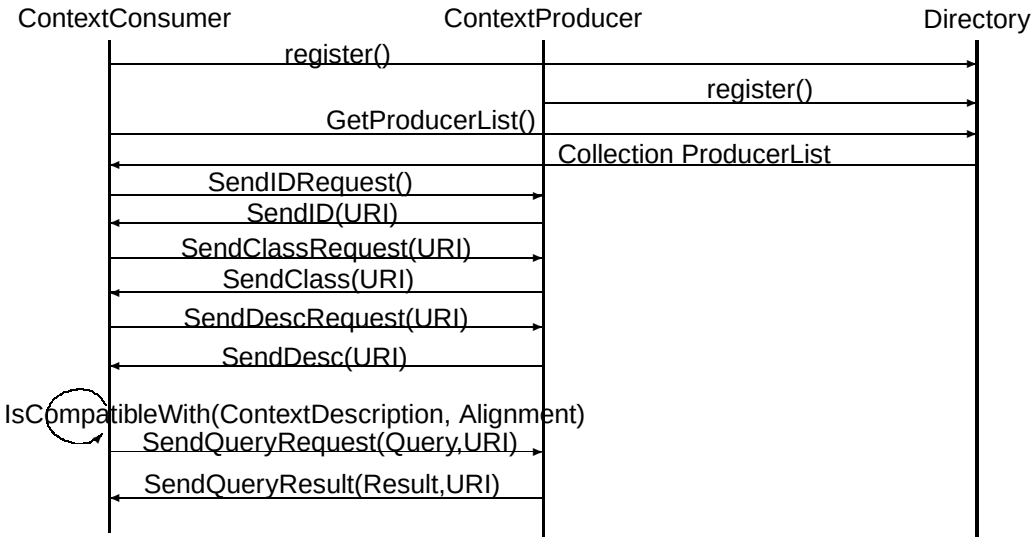


FIGURE 29 – Diagramme d'activité correspondant à l'exécution du protocole.

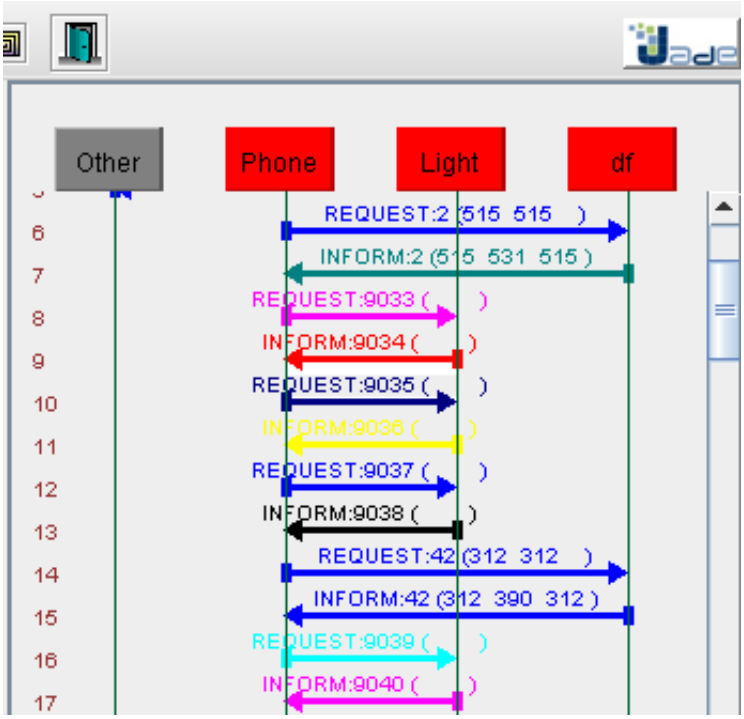


FIGURE 30 – L'interface de renifleur JADE. L'agent téléphone (Phone) est un consommateur d'information de contexte, l'agent luminosité (Light) est un producteur d'information de contexte et l'agent DF est l'annuaire JADE. Le consommateur commence par interroger l'agent annuaire pour connaître tous les agents producteurs de l'environnement. Puis, il interroge tous les producteurs pour connaître leur ID, l'URI de leur classe et de leur modèle sémantique d'information de contexte. S'il trouve une compatibilité entre ces modèles et le sien, il finit par leur envoyer une requête SPARQL.

5.2 Gestion des modèles sémantiques d'informations de contexte

La description de la bibliothèque de gestion d'informations de contexte décrite dans la section précédente (voir 5.1) présente la classe *Context Description* comme la classe utile à la représention sémantique des modèles de contexte des producteurs et des consommateurs. Comme on le voit sur le diagramme de classe de la figure 28, les actions principales possibles sur une description de contexte sont :

- Construire l'objet de description sémantique de contexte à partir de son URI.
- Obtenir l'URI dont est tirée la description sémantique de contexte.
- Obtenir la description sémantique de contexte sous la forme d'une ontologie OWL et RDF.
- Poser des requêtes SPARQL sur cette description et obtenir les réponses sous forme de triplets RDF.
- Obtenir les correspondances entre ce modèle de contexte et un autre modèle de contexte, sous forme d'alignement au format RDF.

Nous présentons les outils spécifiques à la réalisation de ces actions et à la manipulation des ontologies que nous avons utilisées. Les bibliothèques JENA sont utilisées pour transformer des ontologies OWL en objets JAVA. Pour obtenir les correspondances entre les ontologies, nous utilisons un serveur d'alignement d'ontologies présenté dans un deuxième temps.

5.2.1 Manipulation des modèles

Les ontologies sont sous forme OWL et RDF. Les manipulations que nous réalisons avec ces ontologies sont les suivantes :

- Construire des ontologies sous forme d'objets JAVA à partir de fichiers de description d'ontologies en langage OWL et RDF.
- Manipuler ces objets et parcourir ces ontologies pour pouvoir retrouver un objet à partir de son URI, ses propriétés ou les relations entre deux objets.
- Sérialiser des objets OWL ou RDF pour les transporter sur un réseau.
- Construire des requêtes SPARQL sous forme d'objets JAVA.
- Poser des requêtes SPARQL et obtenir des réponses sour forme d'objet JAVA. Ces requêtes étant posées à travers le réseau, il faut aussi pouvoir les sérialiser ainsi que leur réponses.
- Obtenir des correspondances entre plusieurs ontologies sous forme d'objets JAVA.

Pour réaliser ces manipulations, nous avons choisi la bibliothèque JAVA JENA [18], car elle offre une grande partie des fonctionnalités recherchées. Seule la dernière fonctionnalité n'est pas offerte par la bibliothèque JENA. Pour remplir cette dernière fonctionnalité, nous utilisons le serveur d'ontologie, developpé par l'équipe EXMO de l'INRIA Rhône-Alpes présenté dans la section 4.3.1.

Ainsi, à l'aide de la bibliothèque JENA, les agents hebergés par les dispositifs ou les applications de l'environnement commenceront par construire l'ontologie représentant leurs besoins ou leurs capacités à partir d'un fichier qu'ils trouveront sur le réseau, à partir d'un URI, ou sur leurs propres ressources. À partir des objets ainsi créés, ces agents pourront publier leurs classes d'appartenance OWL pour répondre à la performative `Cl(URI)`. Ils pourront aussi fournir la description complète de l'information de contexte qu'ils manipulent pour répondre à la performative `Desc(URI)`. Ils pourront aussi manipuler des URIs et charger les descriptions des informations de contexte d'un autre dispositif à partir d'un URI qui leur est fourni. Quand un consommateur aura récupéré les descriptions des différents producteurs d'information de contexte, il pourra solliciter un serveur d'alignement (voir section 4.3.1) pour obtenir des correspondances entre sa description et celles des producteurs. Ainsi, il pourra choisir le producteur qui lui semble approprié, construire une requête SPARQL et lui envoyer par l'intermédiaire de

18. `http://jena.sourceforge.net/`

la performative `Req(SPARQL)`. Il recevra des informations en retour qu'il pourra ajouter à sa base de connaissances.

5.2.2 Serveur d'alignement d'ontologies

Nous l'avons vu dans la partie 4.3.1, obtenir des correspondances entre des ontologies est une des clefs de notre système pour intégrer des dispositifs hétérogènes en considérant les environnements d'intelligence ambiante comme ouverts et dynamiques. Ses fonctions principales sont les suivantes :

- Trouver une ontologie proche d'une ontologie particulière ;
- Stocker (et retrouver) des alignements déjà obtenus entre différentes ontologies ;
- Calculer dynamiquement la correspondance entre deux ontologies ;
- Traduire les requêtes et réponses aux requêtes vers des gestionnaires de contextes utilisant des ontologies différentes.

Toutes ces fonctionnalités sont supportées par le serveur d'alignement [19], que nous exploitons dans notre implémentation, fourni par la bibliothèque d'alignement [20] d'ontologie de l'équipe EXMO de l'INRIA Rhône-Alpes. Ce serveur est architecturé autour d'un cœur enrichi de plug-ins permettant d'ajouter des protocoles de communication. Afin de l'intégrer dans notre environnement JADE nous avons développé le plug-in correspondant (il est intégré à la livraison standard du serveur d'alignement depuis la version 3.0). Ainsi, il dispose des performatives suivantes pour répondre aux requêtes d'un agent :

- `ALIGN(URI, URI)`→`ID` : Demande au serveur d'alignement, l'alignement des deux ontologies passées en paramètre. La réponse est l'identifiant de cet alignement.
- `RETRIEVE (ID)`→`XML` : Renvoie l'alignement correspondant à l'identifiant passé en paramètre. Le format de l'alignement peut être précisé avec un paramètre supplémentaire. L'alignement peut, par exemple, être exprimé en RDF.
- `TRANSLATE (SPARQL, ID)`→`SPARQL` : Traduit la requête SPARQL passée en paramètre à l'aide de l'alignement dont le deuxième paramètre est l'identifiant.
- `STORE (ID)` : Indique au serveur d'enregistrer l'alignement dont l'identifiant est en paramètre.
- `CUT (ID)`→`ID` : Permet d'effectuer un seuillage de l'alignement passé en paramètre, par rapport aux valeurs de confiance associées aux correspondances. La valeur renvoyée est l'identifiant d'un nouvel alignement.

Quand un consommateur d'information de contexte obtient la description des informations d'un producteur, il invoque le serveur d'alignement avec l'URI de son modèle d'information et celui du producteur. Puis, il peut, s'il le désire, effectuer un seuillage sur cet alignement à partir de son identifiant obtenu à l'étape d'avant. Ainsi, il n'obtiendra que les correspondances avec une valeur de confiance élevée (par exemple supérieure à 0.80). Avec la performative `RETRIEVE`, il obtiendra l'alignement au format désiré, XML par exemple, ce qui lui permettra d'évaluer sa compatibilité avec le producteur. Un algorithme naïf est fourni dans notre implémentation pour réaliser cette évaluation. Il se base sur la moyenne des correspondances obtenues dans l'alignement. Enfin, il invoquera `TRANSLATE` avant d'envoyer une requête au producteur ainsi sélectionné en utilisant l'alignement obtenu lors de leurs échange. Il enverra la requête traduite au producteur. Une autre façon de dialoguer avec le producteur sélectionné est d'instancier un objet `mediator`, en utilisant les URIs des deux modèles d'information de contexte, de la bibliothèque d'alignement d'EXMO qui jouera le rôle de médiateur entre les deux agents en traduisant les requête et les réponses.

19. `http://alignapi.gforge.inria.fr/aserv.html`
20. `http://alignapi.gforge.inria.fr/`

5.3 Bilan

L'architecture définie dans le chapitre 3 est concrétisée sous forme d'interfaces Java. Elles permettent aux fabricants de dispositifs, de capteurs et d'applications d'implémenter un composant de gestion d'informations de contexte capable d'utiliser le protocole et les descriptions de contexte sémantique et ainsi profiter de notre infrastructure décentralisée. De plus, nous avons implémenté ces interfaces en utilisant une plateforme multi-agents JADE où chaque composant est un agent indépendant. Cette implémentation utilise les bibliothèques JENA pour construire et manipuler les modèles d'information de contexte sémantique et le serveur d'alignement de l'INRIA pour les réconcilier. Ces interfaces couvrent l'ensemble de notre proposition d'architecture distribuée pour la gestion des informations de contexte dans un environnement d'intelligence ambiante. L'implémentation qui en est fournie offre la totalité des fonctionnalités prévues par l'interface fonctionnelle. L'implémentation est complètement fonctionnelle comme nous l'illustrons dans le chapitre suivant en présentant le démonstrateur d'intelligence ambiante que nous avons réalisé.

Chapitre 6

Expérimentation

Sommaire

6.1 Présentation 95
6.2 Scénario 96
6.3 Applications et dispositifs du démonstrateur 97
6.3.1 Ontologies "utiles" pour l'environnement 97
6.3.2 LiveMountain 99
6.3.3 Service d'information "neige" de Chamrousse 100
6.3.4 Agenda sur PDA 101
6.3.5 Thermomètre Grenoble 101
6.3.6 Bilan 101
6.4 Une infrastructure pour les environnements d'intelligence ambiante 102
6.4.1 Infrastructure de service 102
6.4.2 Infrastructure de gestion d'information de contexte 104
6.4.3 Infrastructure de composition 104
6.5 Correspondances entre ontologies 104
6.5.1 Sans alignement 105
6.5.2 À l'aide d'alignements 105
6.6 Démonstrateur 108
6.6.1 Installation 109
6.6.2 Interaction avec le démonstrateur 111
6.7 Bilan 114

Afin de valider notre contribution et de l'évaluer, nous l'avons utilisée dans une situation se rapprochant le plus possible d'un environnement d'intelligence ambiante complet. En collaboration avec Mathieu Vallée, doctorant et concepteur d'un système de composition de services intelligents, nous avons réalisé un démonstrateur présenté à Ubicomp 2007. Nous en présentons la partie concernée par la gestion des informations de contexte.

6.1 Présentation

Se rapprocher d'un environnement d'intelligence ambiante, c'est construire un environnement dans lequel les utilisateurs interagissent avec les dispositifs et les applications de façon transparente et intuitive. Ils utiliseront les objets dispersés dans l'environnement et non un terminal unique. L'environnement est peuplé de dispositifs intelligents et de capteurs, qui sont conçus pour fournir une fonctionnalité,

mais pas spécialement pour fonctionner ensemble, ni pour être dédiés à une application spécifique. Ces fonctionnalités seront accessibles par un réseau wifi, ce qui permet de construire des applications en aggrégeant les fonctionnalités de base.

De plus, cet environnement devra conserver les problématiques supposées des environnements d'informatique ambiante, que nous résumons comme suit :

- les dispositifs et les capteurs sont *hétérogènes*,
- l'environnement est *dynamique et imprévisible*,
- les besoins des utilisateurs sont *imprécis et évolutifs*.

Ainsi, nous proposons une infrastructure complète pour les environnements d'intelligence ambiante domestique qui a pour but de simplifier la conception des applications d'intelligence ambiante et de les rendre plus robustes aux conditions réelles. Cette infrastructure repose essentiellement sur les travaux développés dans ce travail de thèse en ce qui concerne la gestion de contexte et les travaux développés par Mathieu Vallée concernant la composition flexible et dynamique d'applications présentée dans [79]. Ont participé au développement de ce démonstrateur Mathieu Vallée, un stagiaire, Rémi Dupuis et moi-même. Rémi Dupuis s'est attelé au développement des services et des applications et de l'intégration des capteurs dans l'infrastructure.

Nous présenterons un scénario réaliste d'utilisation dans notre environnement dans la section 6.2. Puis nous détaillerons la mise en place de l'infrastructure en se concentrant sur le rôle de la partie dédiée à la gestion des informations de contexte dans la section 6.4.

6.2 Scénario

Notre démonstrateur est composé de quatre applications principales que nous avons combinées pour créer un scénario d'intelligence ambiante appelé "Allons skier". Il se déroule comme suit :

Un habitant de Grenoble, Tom, aime skier avec ses amis, surtout quand le soleil est au rendez-vous et que les conditions de ski sont favorables. Pour guetter l'apparition des conditions favorables, il utilise l'application LiveMountain. *Cette application affiche une image représentant un paysage de montagne, évoluant en fonction des conditions météo et de la disponibilité de ses amis. Quand le temps est favorable et que son ami Paul est disponible, il peut obtenir de plus amples informations en utilisant l'application* TangibleZoom. *En approchant son PDA de l'image projetée au mur, il a accès aux détails des conditions pour différentes stations de ski. Pour profiter d'une lecture plus confortable, il peut approcher son PDA d'un écran plus grand qui affichera des informations détaillées. S'il est satisfait des conditions affichées, il voudra trouver un moyen de transport pour se rendre dans la station désirée. Pour cela, il utilisera l'application* iRiderNotification *qui intègre un système de covoiturage pair-à-pair, iRider, accessible depuis les environnements domestiques. Il commencera par exprimer ses souhaits en utilisant une page web. Puis iRider recherche et négocie des rendez-vous avec les propriétaires de voitures. S'il y a une proposition intéressante,* iRiderNotification *cherche un moyen pour en informer Tom en utilisant un dispositif de notification. Ce dernier est choisi dynamiquement en fonction de la pertinence de la proposition et du contexte de Tom. Par exemple, si Tom est occupé et que la proposition est jugée peu intéressante, il n'y a pas de nécessité de le déranger, ainsi un dispositif de notification périphérique peut être utilisé. Par contre, quand une proposition très intéressante est disponible,* iRiderNotification *activera une sonnette. Une offre provenant de son ami Paul qui travaille dans une station proche, partira dans une heure. Paul a déjà accepté cette proposition, donc Tom confirme son intérêt et se prépare pour sa journée à la montagne.*

6.3 Applications et dispositifs du démonstrateur

Nous allons présenter les différents dispositifs, capteurs et applications dont nous avons doté notre environnement pour réaliser ce démonstrateur. Ils exhibent sur le réseau des ontologies représentant les informations de contexte qu'ils manipulent. Ces ontologies utilisent d'autres ontologies générique que nous avons mises à disposition sur le réseau. C'est pourquoi, nous allons consacrer un premier temps à présenter ces ontologies générique.

6.3.1 Ontologies "utiles" pour l'environnement

Nous avons conçu six ontologies générique pour permettre aux différents dispositifs de décrire leur modèle de contexte. Dans un souci de simplicité de réalisation et pour la clarté de notre explication, nous avons préféré concevoir nous-même ces ontologies, plutôt que d'utiliser celles disponibles sur Internet. Cependant, nous avons pris un soin particulier à la réalisation de ces ontologies pour qu'elles reflètent l'hétérogénéité des descriptions que l'on peut trouver sur le web.

Ontologie : "Conditions de ski"

La première ontologie (voir figure 31) décrit les conditions météorologiques relatives à la pratique du ski, notamment l'ensoleillement, le niveau d'enneigement des pistes et la température.

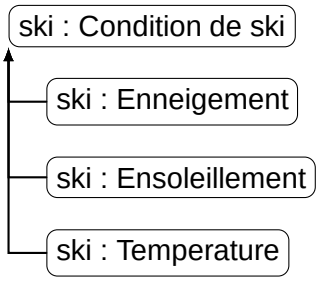


FIGURE 31 – L'ontologie "condition de ski".

Ontologie :"Contexte physique"

La deuxième ontologie (voir figure 32) utilisée dans notre démonstrateur représente le contexte physique en s'attachant aux aspects température, temporel et à la localisation.

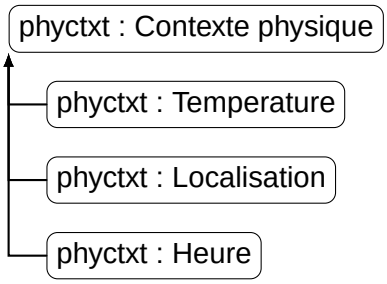


FIGURE 32 – L'ontologie "contexte physique".

Ontologie : "Contexte"

La troisième ontologie (voir la figure 33) décrit le contexte physique et le contexte social comme sous-classe du contexte. Le deuxième étant "décrit par" des relations et "utilise" des activités.

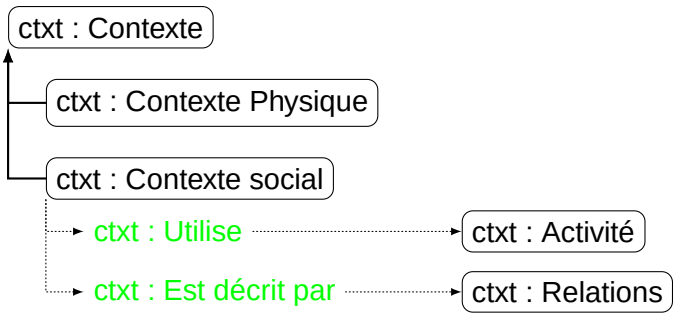


FIGURE 33 – L'ontologie "contexte".

Ontologie : "PDA"

L'ontologie de la figure 34 est typiquement une ontologie qui pourrait servir à décrire les fonctionnalités basiques d'un agenda électronique. Elle nous indique que "calendrier", "notes" et "contacts" sont des sous-classes d'une "application". De plus, on apprend qu'un "calendrier" utilise des "rendez-vous".

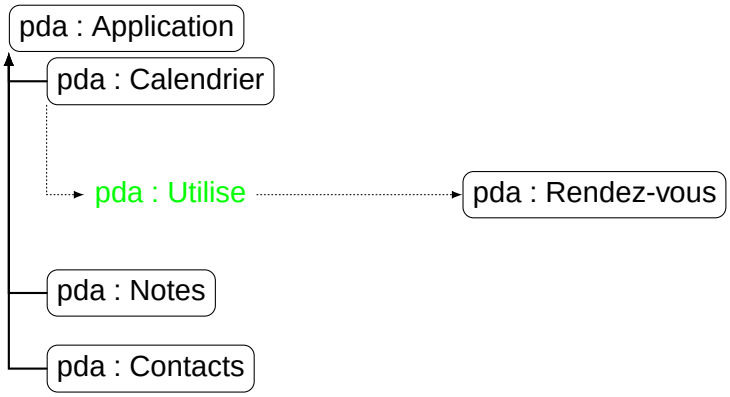


FIGURE 34 – L'ontologie "PDA".

Ontologie "Météo"

L'ontologie "Météo" décrit des conditions météorologiques incluant la température, l'ensoleillement et la pression (voir figure 35).

Ontologie "Ville"

La dernière ontologie générique de l'environnement est une hiérarchie de villes (voir figure 36).

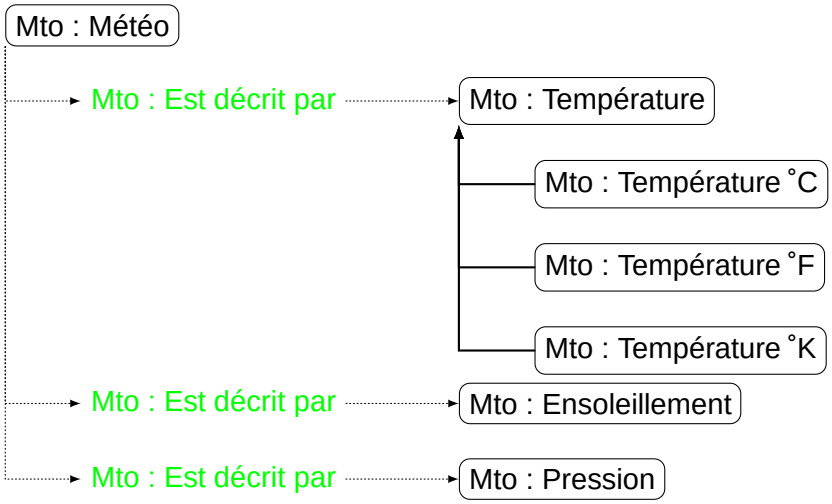


FIGURE 35 – L'ontologie "Météo".

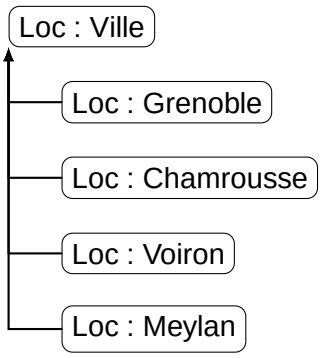


FIGURE 36 – L'ontologie "Ville".

6.3.2 LiveMountain

Tom utilise *Live Mountain* pour organiser ses journées de ski avec Paul. L'application lui permet de surveiller l'apparition des conditions adéquates. Elle prend la forme d'un cadre représentant un tableau, qui évolue en fonction des conditions. LiveMoutain interroge des services web spécialisés dans la météo et les conditions de ski de la station qui intéresse Tom et Paul. S'ils ne sont pas disponibles, l'application essayera de contacter d'autres sources d'information en tenant compte de leur précision et de la confiance qu'elle leur accorde pour représenter ces informations. Ainsi, Tom peut se rendre compte d'un seul coup d'oeil des conditions de ski (voir figure 52).

Lorsqu'il s'aperçoit que le temps s'annonce bien et que son ami Paul a l'intention d'aller en montagne, il peut obtenir plus d'information grâce à l'application *TangibleZoom*. Il lui suffit d'approcher son PDA du tableau pour avoir accès aux détails des conditions dans différentes stations. Pour plus de confort, il peut déplacer son PDA près d'un écran plus grand, sur lequel les informations apparaissent alors. S'il est satisfait des conditions et disponible pour une sortie, l'application *iRider* lui permet de rechercher un moyen de covoiturage pour se rendre en montagne avec Paul. En fonction de l'intérêt des propositions et de sa situation courante, Tom reçoit des propositions de covoiturage sur un dispositif approprié. Lorsqu'une proposition plus intéressante se présente, iRider déclenche l'application *AlertMe*,

qui cherche à alerter Tom de manière graduelle, en tenant compte de l'échéance de la proposition et du contexte d'utilisation. Il s'agit d'une offre de covoiturage proposée par une amie de Paul, qui travaille dans une station proche, et Paul a déjà accepté cette offre. Tom accepte lui aussi et se prépare pour cette journée.

LiveMountain est un consommateur d'information de contexte, implémente la classe *contextConsumer* avec le modèle de contexte de la figure 37. Pour fonctionner, l'application LiveMountain s'intéresse aux informations suivantes sur l'environnement (voir la figure 37) :

- la météo définie par l'ontologie `Mto` de la localisation Chamrousse définie dans l'ontologie `Loc` afin de construire la représentation symbolique qu'il affiche,
- l'activité de l'utilisateur telle qu'elle est définie par l'ontologie `ctxt`, afin de la transmettre aux applications LiveMoutain distantes.
- les conditions de ski définies par l'ontologie `ski` de la localisation Chamrousse de l'ontologie `Loc` ; pour affiner sa représentation symbolique.

Pour fonctionner, l'application TangibleZoom s'intéresse aux informations suivantes sur l'environnement :

- la localisation précise de l'utilisateur afin de savoir quel est le dernier dispositif d'interface qui a été approché,
- la localisation et l'activité des autres utilisateurs présents dans l'environnement pour ne pas les perturber et pouvoir gérer les conflits entre applications.

Pour fonctionner, l'application AlertMe s'intéresse aux informations suivantes sur l'environnement :

- la localisation de l'utilisateur pour savoir quels sont les dispositifs d'interface disponibles dans cette pièce,
- l'activité de l'utilisateur pour pouvoir adapter le niveau d'alerte et la moyen de l'alerte,
- la localisation et l'activité des autres utilisateurs présents dans l'environnement pour ne pas les perturber et pouvoir gérer les conflits entre applications.

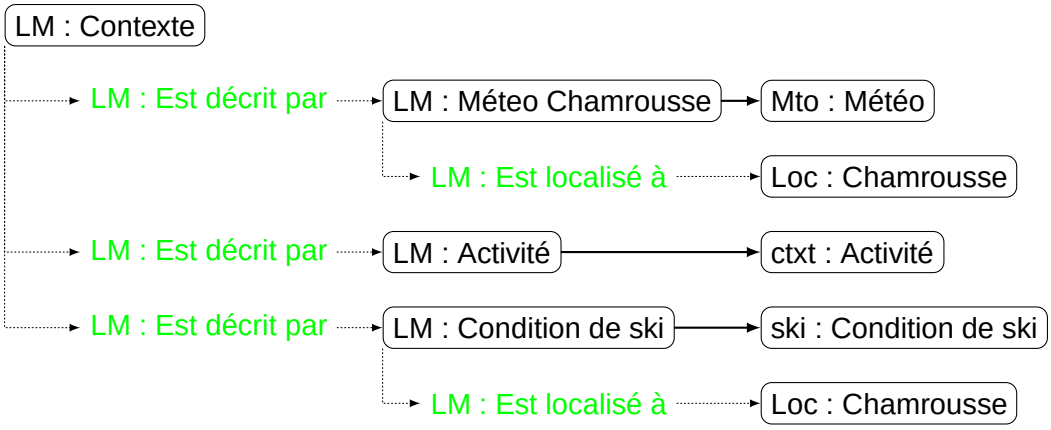


FIGURE 37 – Ontologie représentant les informations de contexte recherchées par l'application "Live Mountain".

6.3.3 Service d'information "neige" de Chamrousse

Un service web accessible depuis le réseau fournit des informations sur les conditions de ski à Chamrousse. Pour les besoins de l'expérimentation, nous avons développé ce service web qui s'exécute sur un serveur local, comme une réplique d'un service web existant. Cela nous permet, à travers une console

d'administration, de simuler les changements dans les conditions de ski. Ce service web agit comme un fournisseur d'information de contexte pour notre environnement, notamment pour l'application Live-Mountain. Il implémente la classe *ContextProducer*. Il est capable de fournir les informations de la figure 38, c'est-à-dire l'enneigement tel qu'il est défini dans l'ontologie `ski` pour la localisation Chamrousse de `Loc`.

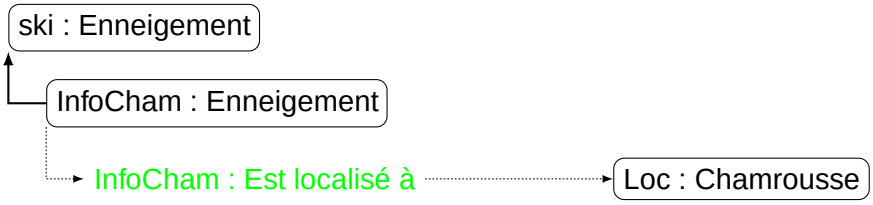


FIGURE 38 – Ontologie représentant les informations de contexte fournies par le service d'information "neige" Chamrousse.

6.3.4 Agenda sur PDA

Le PDA de Tom est fournisseur d'informations de contexte et implémente donc la classe *Context-Producer*. Il peut, notamment fournir son emploi du temps, défini en tant que sous-classe de `pda:Calendrier`, comme le montre la figure 39. Cette information peut donner des indications sur l'occupation de Tom et donc sur la meilleure façon de lui transmettre un message.

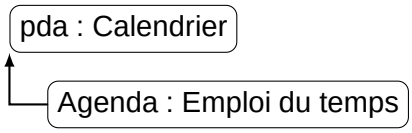


FIGURE 39 – Ontologie utilisée par l'application d'agenda sur le PDA pour représenter les informations de contexte qu'il fournit.

6.3.5 Thermomètre Grenoble

L'environnement du démonstrateur est censé être localisé à Grenoble. Cet environnement est équipé d'un thermomètre qui fournit la température, défini comme sous-classe de `phyctxt:Température`, pour la localisation de Grenoble. On peut voir un détail de l'information fournie par le thermomètre sur la figure 40. Le thermomètre, ou plutôt son représentant sur le réseau implémente la classe *ContextProducer*.

6.3.6 Bilan

Un bilan des dépendances entres les différents modèles d'informations de contexte de chacun des dispositifs est exposé sur la figure 41.

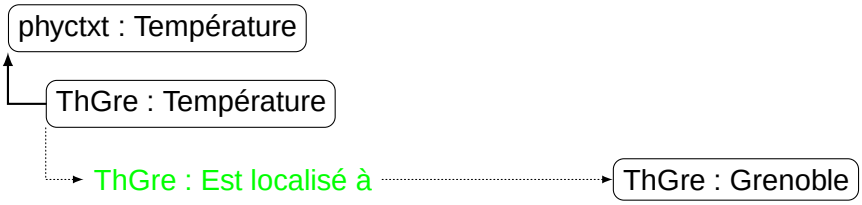


FIGURE 40 – Ontologie utilisée pour représenter les informations de contexte produites par le dispositif "Thermomètre Grenoble".

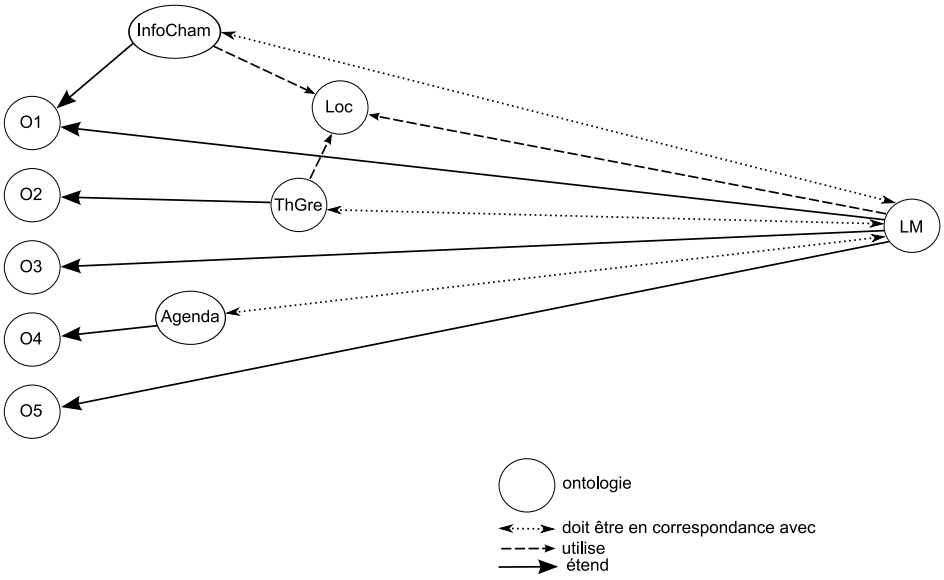


FIGURE 41 – Relation entre les ontologies de l'environnement.

6.4 Une infrastructure pour les environnements d'intelligence ambiante

Notre approche est de construire une architecture générique pour les environnements d'intelligence ambiante. Elle est représenté par la figure 42 qui illustre certains éléments du scénario décrit dans la section 6.2. Elle se décompose en trois éléments, l'infrastructure de services, l'infrastructure de contexte et l'infrastructure de composition, que nous allons décrire un par un.

6.4.1 Infrastructure de service

L'infrastructure de service fournit les services de base pour le bon fonctionnement de notre architecture dans l'environnement. Elle permet la découverte et la communication entre des capteurs et des dispositifs hétérogènes présents dans l'environnement. Par exemple, nous avons représenté sur la figure 53 deux des services pouvant être utilisés pour notifier l'utilisateur : le service `MessagePopup` déclenche une fenêtre popup sur le PDA de Tom et le service `AmbientLamp` utilise l'intensité lumineuse d'une lampe de l'environnement pour attirer l'attention de l'utilisateur. Le service de covoiturage, `iRider`, permet de notifier l'utilisateur à chaque fois qu'une nouvelle proposition est disponible. Différents systèmes de notification seront utilisés en fonction du contexte des utilisateurs. Au sein de cette infrastructure cohabitent trois environnements d'exécution différents : une plateforme J2EE, une pla-

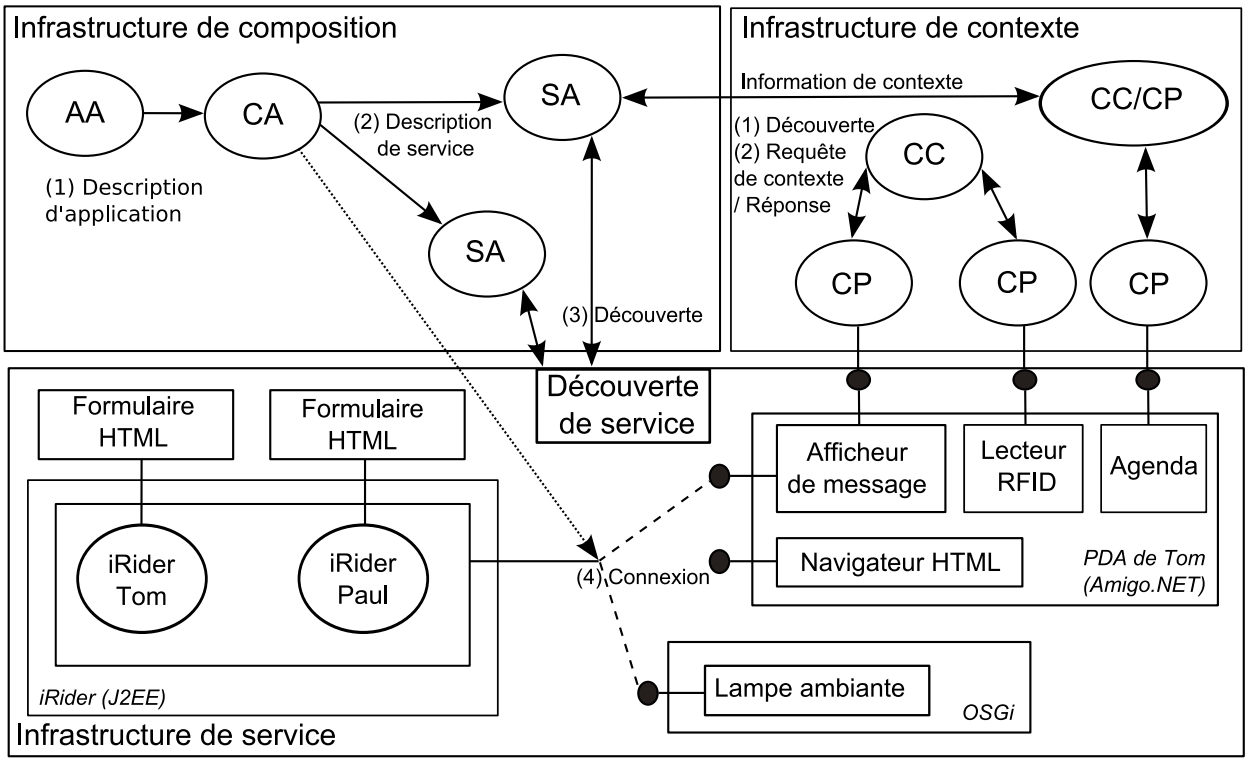


FIGURE 42 – Vue d'ensemble de l'architecture. CP : Producteur d'information de contexte, CC : Consommateur d'information de contexte. AA : Agent Assistant, CA : Agent Composeur, SA : Agent Superviseur.

teforme OSGI et une plateforme Amigo.NET. Chacun de ces environnements héberge une plateforme d'exécution pour des agents JADE. L'interropérabilité fonctionnelle entre les différents services des différents environnements d'exécution est assurée principalement par la communication possible entre ces plateformes multi-agents. En effet, les services fournis par les infrastructures de contexte et de composition sont implémentés sous forme d'agents et chaque dispositif et application possède un représentant sur une des plateformes multi-agents.

Les services peuvent être *source de contexte*, c'est-à-dire qu'ils diffusent de l'information sur l'environnement pouvant servir aux applications pour adapter leur comportement. Dans notre scénario, on peut considérer que le PDA de Tom fait figure de trois sources de contexte différentes. Premièrement, l'*Agenda* fournit des informations sur les occupations de Tom. Deuxièmement, le *Lecteur RFID* en lisant des étiquettes RFID disséminées dans l'environnement, fournit la localisation du PDA et de son porteur, Tom par exemple, ainsi que l'intérêt probable de l'utilisateur du dispositif de l'environnement porteur de ce tag. Enfin, l'application *MessagePopup* est aussi considérée comme une source de contexte. En effet, lorsque Tom clique sur un message popup, on peut considérer que celui-ci l'intéresse.

Une *application de gestion de l'infrastructure* fournit les services nécessaires aux applications pour profiter de l'environnement d'intelligence ambiante. Elle se divise en deux parties : l'*infrastructure de contexte* qui nous intéresse particulièrement et l'*infrastructure de composition*.

6.4.2 Infrastructure de gestion d'information de contexte

L'infrastructure de gestion d'information de contexte est l'implémentation multi-agent de la solution que nous proposons (voir chapitre 3, 4 et 5). Elle a pour but de fournir de l'information sur l'environnement en provencance de producteurs d'informations de contexte (CPs), vers les consommateurs d'informations de contexte (CCs). L'infrastructure s'attaque aux problèmes d'hétérogénéité et de dynamisme en créant des liaisons dynamiques entre les CCs et les CPs qui n'ont pas été conçus pour fonctionner ensemble, et en utilisant les technologies d'alignement d'ontologie [35] pour résoudre le problème d'hétérogénéité. La figure 18 illustre l'utilisation de la solution dans le cadre du démonstrateur. Un CC ayant besoin de la disponibilité de Tom commencera par découvrir tout les CPs de l'environnement *(1)* et obtient, en exécutant le protocole, les informations qu'ils fournissent *(2)*. Ce modèle d'information de contexte est exprimé avec une ontologie OWL [28]. Lorsqu'un modèle d'information de contexte contient un concept inconnu d'un CC, celui-ci demande à un serveur d'alignement d'ontologie [36] qui lui retournera l'ensemble des correspondances entre le modèle et son propre modèle. Si le CC trouve des compatibilités entre les deux modèles, il enverra une requête SPARQL [64], exprimée avec le vocabulaire de l'ontologie de CC et traduite dans le vocabulaire de l'ontologie de CP par le serveur d'alignement. Ce serveur traduira également le résultat de la requête, exprimé sous forme de triplets RDF, de CP vers CC. Les applications, les services et les dispositifs peuvent être CP, CC ou les deux.

6.4.3 Infrastructure de composition

L'infrastructue de composition supporte l'assemblage et la reconfiguration des services en applications. Elle répond aux exigences citées dans la section 2.2.2 grâce à la gestion flexible des applications basée sur des techniques de représentation de connaissances et les systèmes multi-agents [79]. La figure 42 montre le cas de l'application *iRiderNotification* qui gère la redirection des notifications des propositions de covoiturage sur le dispositif approprié. Un agent assistant, AA, déclenche la composition en fournissant la description de l'application à réaliser à l'agent composeur CA *(1)*. Cette description exprime que le service iRider doit être connecté à un service de notification. Ce dernier est décrit de façon abstraite sans aucune référence à un service existant. Pour composer l'application, le CA envoie des requêtes de service à tous les agents superviseurs SA *(2)*. Chaque agent superviseur obtient des descriptions des services disponibles du service d'infrastructure *(3)*, et évalue leur pertinence pour effectuer la tâche voulue dans l'application cible. Afin de choisir le service de notification approprié, un SA doit connaître la disponibilité de Tom. Pour cela, il utilise notre infrastructure de gestion de l'information de contexte. Une fois qu'une composition correcte est trouvée, CA établit une connexion entre les différents services *(4)*. Ainsi construit, le système est complètement adaptatif : quand le contexte change ou quand un service apparaît ou disparaît, l'assemblage de l'application est modifié.

6.5 Correspondances entre ontologies

Pour atteindre l'interopérabilité en cours d'exécution entre les producteurs et les consommateurs d'informations de contexte présents dans l'environnement, l'infrastructure de contexte utilise un serveur d'alignement pour lui fournir les correspondances entre les ontologies décrivant les modèles d'informations de contexte que manipulent les différents dispositifs. Cependant, si deux dispositifs utilisent des modèles qui se référent à la même ontologie, le consommateur d'informations de contexte calculera lui même le degré de correspondance avec le producteur d'informations de contexte. C'est ce premier cas que nous détaillerons. Dans le cas où les dispositifs exhibent des modèles se référant à des ontologies différentes, le consommateur d'informations de contexte utilisera un alignement pour calculer son degré de compatibilité avec le producteur. Ces alignements stockés par le serveur sont soit calculés pendant

l'exécution ou pré-renseignés. Nous allons détailler les alignements utilisés pour le bon fonctionnement de notre environnement dans un deuxième temps.

6.5.1 Sans alignement

Afin de calculer son degré de compatibilité avec un producteur d'informations de contexte dont le modèle utilise une ontologie commune avec celles utilisées pour représenter le modèle d'information du consommateur, ce dernier n'aura pas besoin d'utiliser un service d'alignement, mais recherchera les relations de subsomption ou d'équivalence entre les concepts des différents modèles. Nous illustrons ce cas de figure avec les correspondances entre l'ontologie du service LiveMountain et l'ontologie du service d'information sur Chamrousse (voir figure 43).

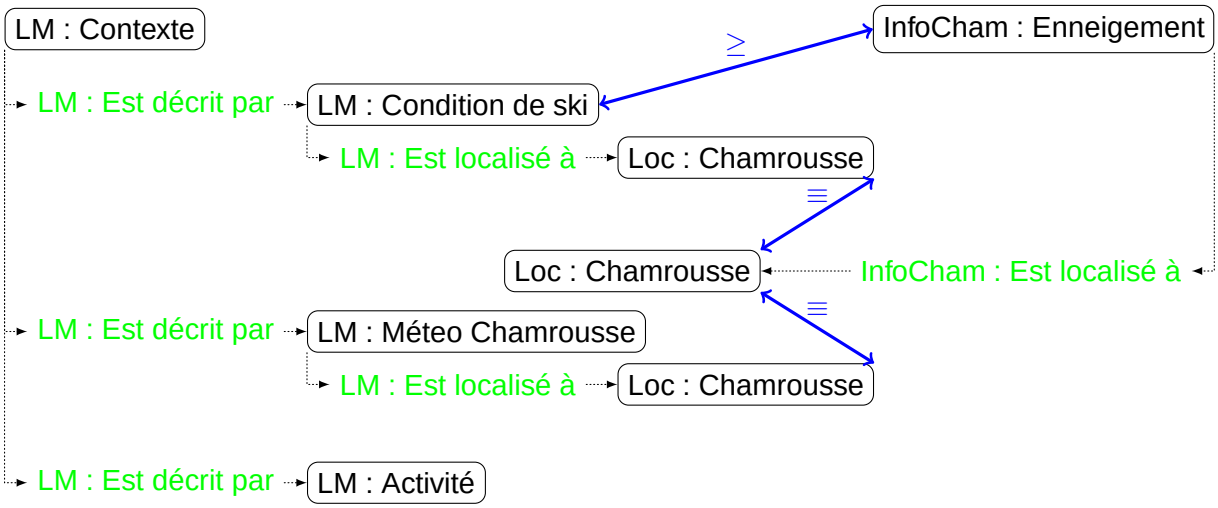


FIGURE 43 – Correspondances entre deux ontologies avec des relations de subsomption et d'équivalence.

6.5.2 À l'aide d'alignements

Dans le cas où l'on cherche des correspondances entre deux concepts provenant de deux ontologies différentes, on utilise un alignement. Les alignements peuvent être engendrés de deux façons :

- Manuellement, c'est-à-dire qu'un utilisateur ou un expert construira l'ensemble des correspondances entre les deux ontologies à aligner.
- Automatiquement, c'est-à-dire que les correspondances entre les deux ontologies sont calculées par une machine. Il existe differents algorithmes pour réaliser ce calcul qui peut se baser par exemple sur des distances syntaxiques ou sémantiques entre les concepts.

En plus de ces deux types d'alignement, nous avons distingué les alignements géographiques qui ont la particularité de faire correspondre des concepts représentant des lieux en fonction de leur proximité géographique.

Alignement manuel

Le premier alignement manuel est entre l'ontologie d'application et l'ontologie de contexte. Il indique qu'un rendez-vous est une sorte d'activité (voir figure 44).

FIGURE 44 – Alignement manuel 1.

Le second alignement manuel est entre le modèle de contexte du service LiveMountain et le modèle de contexte de l'Agenda. Il indique qu'un rendez-vous est une sorte d'activité (voir la figure 45).

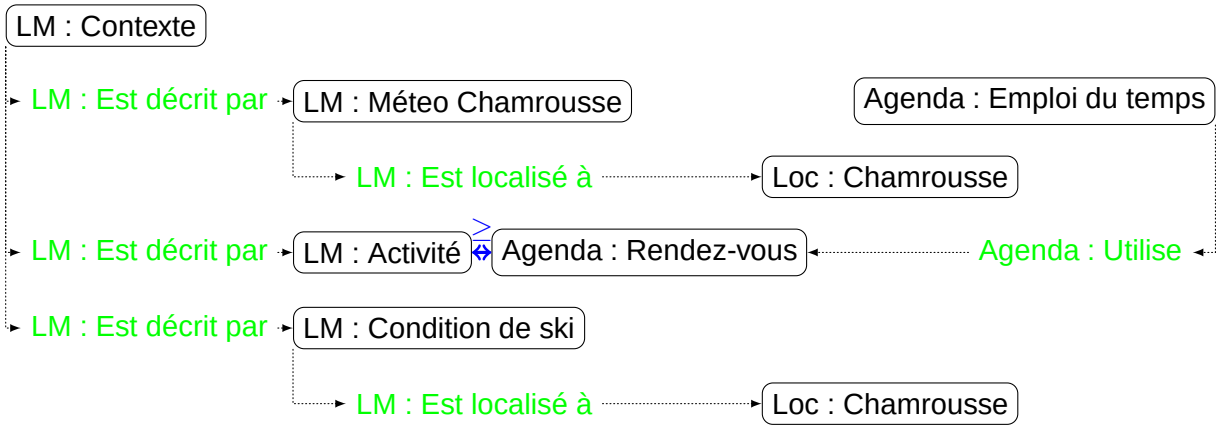


FIGURE 45 – Alignement manuel 2.

Alignement automatique

Les alignements automatiques que nous présentons sont basés sur la distance syntaxique entre les concepts des différentes ontologies. Le premier alignement automatique est entre l'ontologie de météo et l'ontologie de contexte physique. Il indique une équivalence entre les deux concepts températures des deux ontologies (voir la figure 46). Le second alignement automatique est entre le modèle du service LiveMountain et le modèle du service de thermomètre à Grenoble. Il indique une équivalence entre les deux concepts températures des deux ontologies (voir figure 47).

Alignement "géographique"

En utilisant l'alignement géographique de la figure 48 qui représente la proximité de différentes villes aux alentours de Grenoble, le service LiveMountain, consommateur d'information de contexte, peut

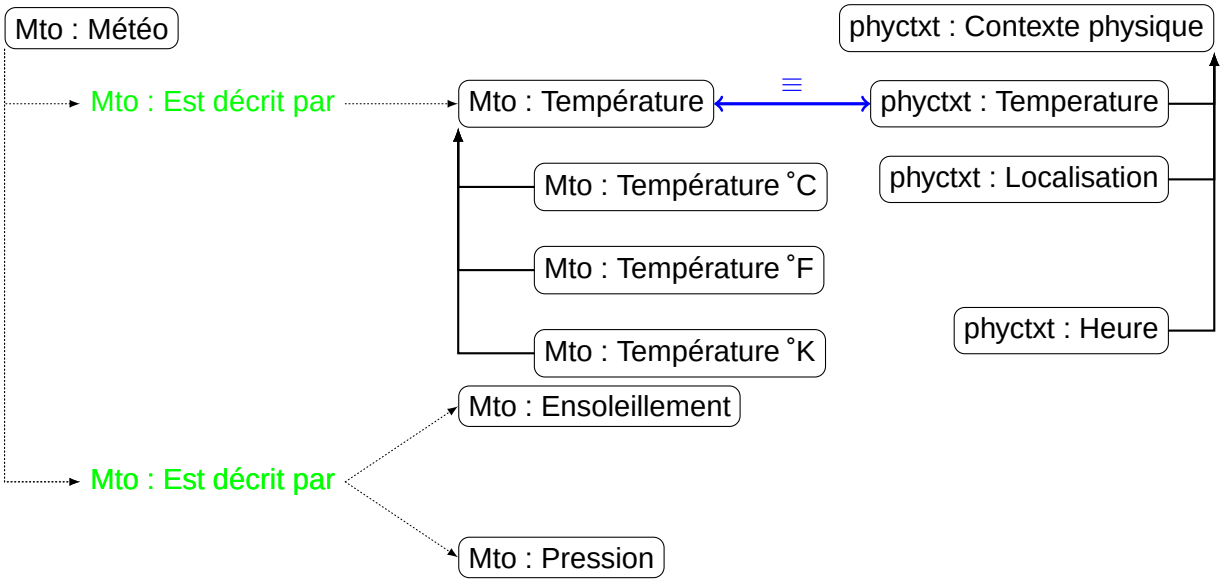


FIGURE 46 – Alignement automatique syntaxique 1.

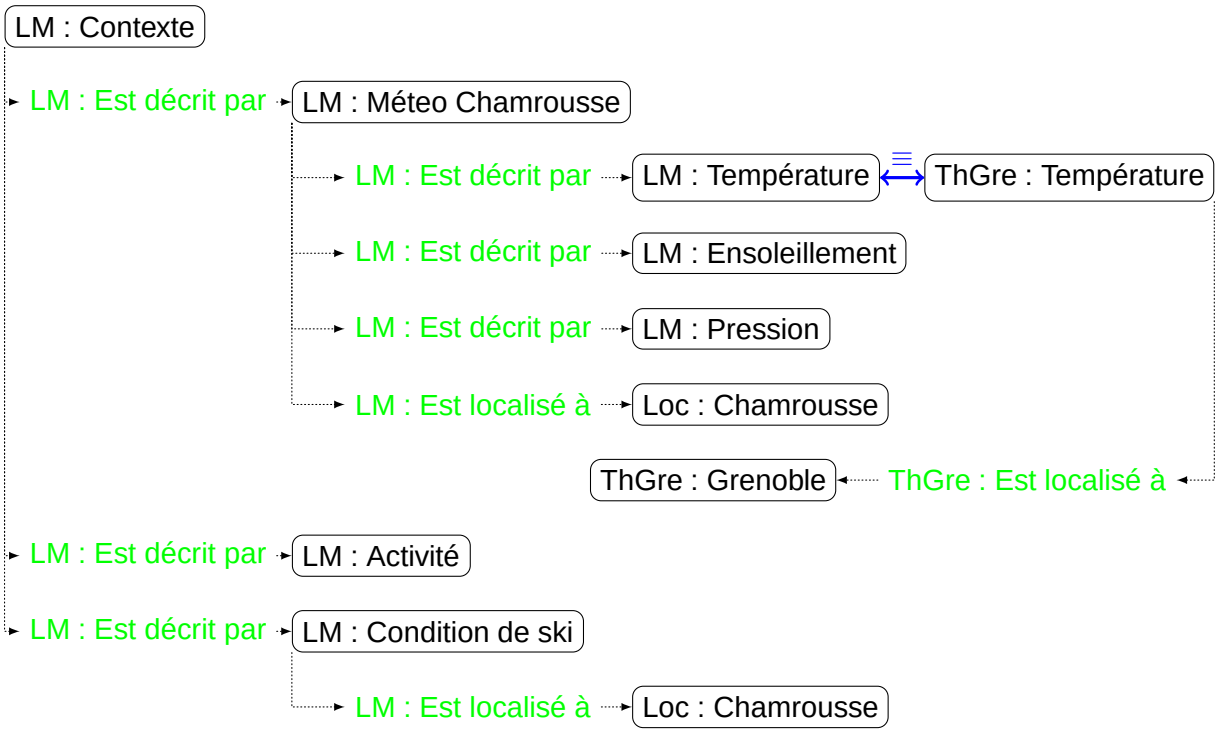


FIGURE 47 – Alignement automatique syntaxique 2.

évaluer la confiance qu'il accorde aux informations provenant du service de température de Grenoble (voir figure 49).

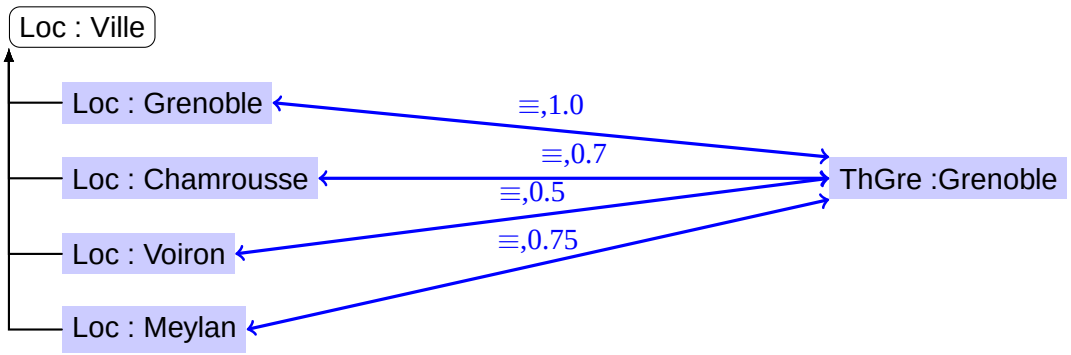


FIGURE 48 – Alignement géographique.

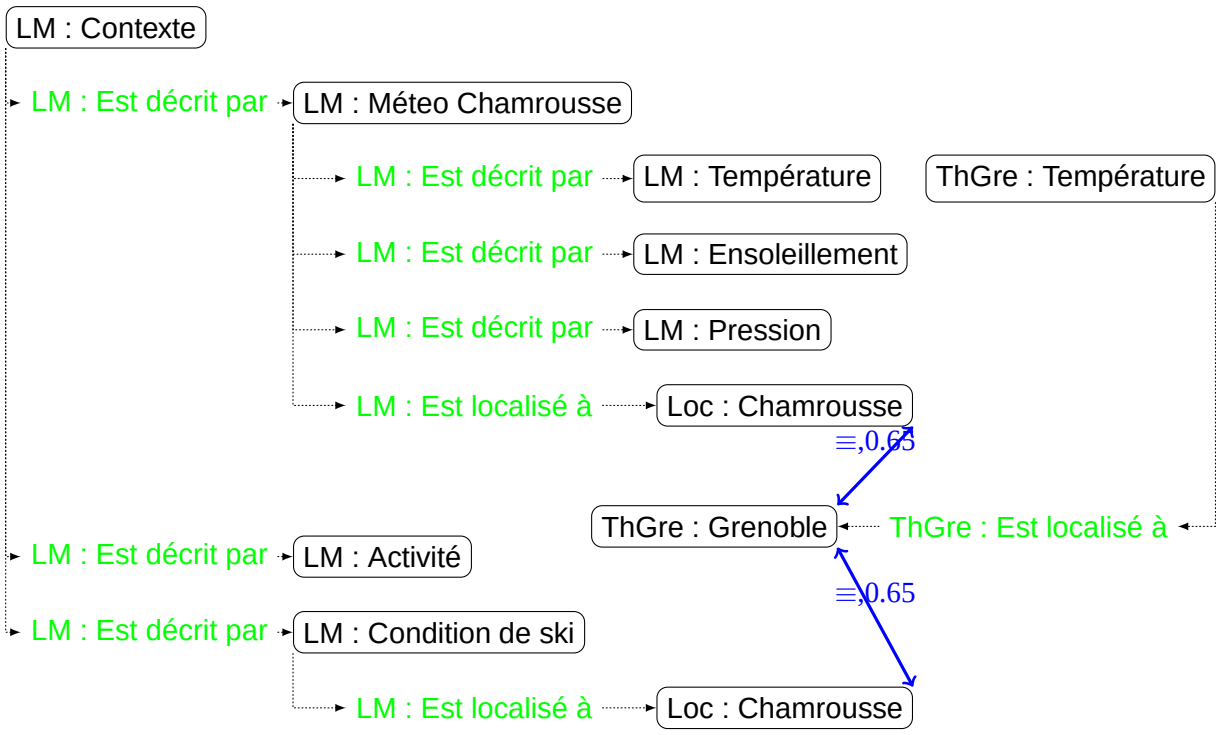


FIGURE 49 – Correspondance géographique.

6.6 Démonstrateur

Ce démonstrateur à été présenté à la conférence Ubicomp 2007 où nous en avons installé une version légère pour des raisons pratiques liées au transport, mais qui offrait tout de même de nombreux moyens d'interaction pour les visiteurs. Son but premier était de montrer une infrastructure capable d'accueillir facilement des capteurs, des dispositifs et des applications indépendantes et de les combiner pour obtenir les comportements espérés. Nous commencerons par décrire l'installation du démonstrateur dans la section 6.6.1. Puis, nous présenterons les interactions que les visiteurs peuvent réaliser avec le démonstrateur dans la section 6.6.2.

6.6.1 Installation

Nous allons présenter l'installation de notre démonstrateur en commençant par décrire son organisation spatiale, puis les dispositifs et les capteurs utilisés. Enfin, nous présenterons les outils de simulations que nous avons inclus dans notre démonstrateur pour le rendre plus riche ; ainsi que les outils d'administration.

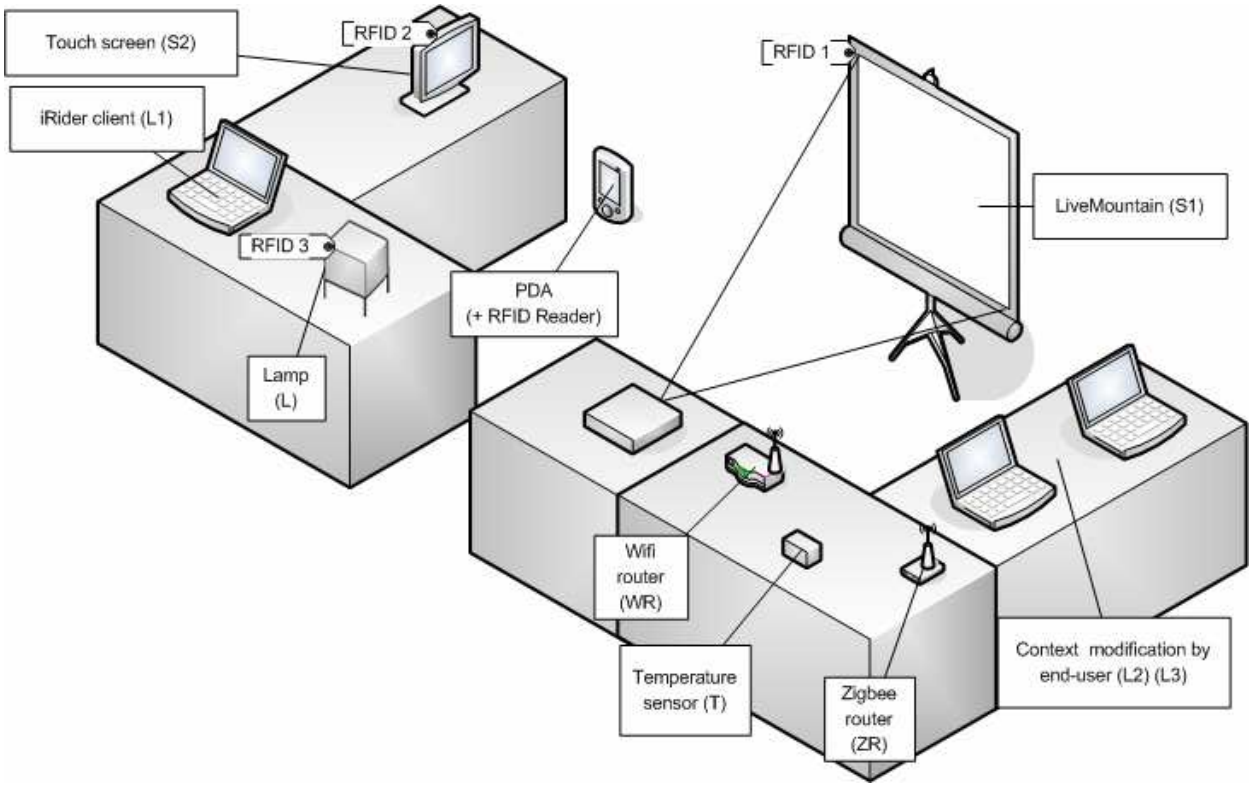


FIGURE 50 – Organisation spatiale du démonstrateur.

Organisation spatiale

La figure 50 montre la disposition spatiale du démonstrateur. Il a été conçu pour former un espace clos et se rapprocher le plus possible d'un environnement domestique. D'un côté, nous avons disposé une surface de projection vidéo. De l'autre côté, nous avons disposé quatre tables pour installer les différents éléments du démonstrateur. Nous les avons regroupés en trois groupes comme suit :

- *Capteurs et dispositifs domestiques* sont supposés être présents dans l'environnement et visibles par l'utilisateur. Les visiteurs peuvent donc les voir, les manipuler et interagir avec eux. Ce sont les éléments (S1), (S2), (L), (T) and (PDA).
- *Les consoles de simulation* représentent les services présents en dehors de l'environnement, présents à l'extérieur ou sur Internet. Les visiteurs en dehors de l'environnement s'en servent pour simuler un changement impactant l'environnement. Ce sont les éléments (L1) and (L2).
- *Les consoles d'administration* sont accessibles depuis l'extérieur de l'environnement et permettent aux visiteurs de surveiller ce qui se passe en détails dans l'infrastructure (L3).

Nous allons maintenant détailler les éléments qui composent chacun des trois groupes.

Capteurs et dispositifs domestiques

Le démonstrateur est composé d'un ensemble de dispositifs intelligents qui, ensemble, forment un environnement intelligent. Nous distinguons trois types de dispositifs dans notre environnement : des écrans intégrés à l'environnement, les petits capteurs et dispositifs et les dispositifs portables.

Les écrans intégrés à l'environnement

L'environnement est équipé de deux écrans :

Un écran mural (S1) affiche la représentaion symbolique proposée par l'application *LiveMountain*. L'utilisateur le verra comme un tableau classique qui a la particularité d'évoluer au cours du temps. Dans le cadre de ce démonstrateur, nous avons projeté cette image à l'aide d'un projecteur relié à un ordinateur. Ce dernier, connecté au réseau, n'exécutait qu'une seule application qui collectait les informations de contexte de l'environnement et affichait l'image correspondante.

Un tableau d'affichage tactile (S2) utilisé pour afficher des messages à l'attention de l'utilisateur. Les messages sont affichés sous forme de pop-up puis ils peuvent être effacés ou détaillés si l'utilisateur les touche. Cet écran peut aussi servir pour visualiser une page Internet associée à un message. Quand il n'y pas de message, l'écran affiche des photos. Pour la réalisation du démonstrateur, le tableau est réalisé avec un écran tactile relié à un ordinateur qui reçoit les messages et les affiche.

Petits dispositifs et capteurs

L'environnement contient plusieurs petits dispositifs et capteurs. Ces objets communicants utilisent le standard de communication sans fil ZigBee. Ils sont connectés au réseau local et autres dispositifs grâce à un routeur ZigBee (ZR) développé dans notre laboratoire. Deux types d'objets communicants sont utilisés dans notre démonstrateur :

La lampe ambiante (L) est une lampe dont on peut contrôler la lumière et la luminosité à distance. Elle est utilisée pour transmettre une information à l'utilisateur de façon "douce". Dans notre démonstrateur, nous l'utilisons pour notifier à l'utilisateur l'arrivée d'un message en modifiant l'éclairage de la lampe en fonction de l'importance du message.

Les capteurs de température (T) envoient périodiquement les valeurs recueillies sur le réseau. Nous les utilisons pour indiquer la température de la station de ski. Dans notre environnement, les utilisateurs peuvent les manipuler et agir ainsi sur ce paramètre.

Les dispositifs personnels portables

Le PDA est le seul dispositif qui ne fait pas partie intégrante de l'environnement puisqu'il est rattaché à un utilisateur. Il a, pour notre démonstrateur, trois utilités :

- Stocker les informations personnelles de l'utilisateur telle que son agenda ou ses préférences. L'utilisateur peut consulter ou modifier ces informations durant la démonstration.
- Fournir une interface pour contrôler ou interagir avec le système. A la différence des autres écran de l'environnement dédiés à une seule application, le PDA peut être utilisé comme un ordinateur classique afin de démarrer une application ou ouvrir une page web par exemple.
- Connecter l'utilisateur à des objets de l'environnement en utilisant des tags RFID. Des tags RFID (RFID1, RFID2, RFID3) sont attachés à des objets de l'environnement. Ainsi, l'utilisateur peut interagir avec ces objets en y approchant le PDA. L'utilisateur accède aux informations relatives à l'objet (une proposition de covoiturage en approchant le PDA de la lampe allumée par exemple),

de plus cette action est interprétée par l'infrastructure comme une information de contexte qui peut être la localisation de l'utilisateur ou son interêt.

Consoles de simulation

Le web service météorologique fournit la météo de la station de ski. Ce service web s'exécute sur le réseau local sur l'ordinateur L2, ainsi les visiteurs peuvent simuler un changement dans les conditions météo grâce à une console de simulation.

Le service, iRider, de covoiturage permet à l'utilisateur d'enregistrer sa recherche de covoiturage et d'être notifié des propositions correspondantes. Ce service s'exécute sur l'ordinateur L1 et est accessible via notre réseau local. En utilisant une page internet, le visiteur peut enregistrer une nouvelle proposition de covoiturage et s'apercevoir que la notification qu'elle engendre sera différente en fonction du contexte de l'utilisateur dans l'environnement et sa compatibilité avec la recherche de l'utilisateur.

Consoles d'administration

Le but principal de ce démonstrateur est de présenter le rôle de l'infrastructure utilisée. Pour offrir la possibilité aux visiteurs de comprendre son fonctionnement, nous avons utilisé plusieurs consoles d'administration qui permettent principalement de :

- Parcourir les différentes descriptions sémantiques utilisées pour représenter l'information de contexte et pour l'infrastructure de composition.
- Montrer la découverte et la sélection des différents producteurs d'informations par les consommateur d'informations.
- Montrer le comportement de l'infrastructure de composition pendant la découverte et la composition de services.

Désignation	Type	OS	Processeur	Mémoire	Connectivité
PDA	PDA	Windows CE	500 Mhz	128 Mo	Wifi + RFID
L1	Portable	Windows XP	1,6 Ghz	1 Go	Wifi
L2	Portable	Windows XP	1,6 Ghz	1 Go	Wifi
L3	Portable	Windows XP	1,6 Ghz	1 Go	Wifi
S1	Portable	Windows XP	1,6 Ghz	512 Mo	Wifi
S2	Portable	Windows XP	1,6 Ghz	512 Mo	Wifi

TABLE 6.1 – Caractérisiques des dispositifs du démonstrateur

6.6.2 Interaction avec le démonstrateur

Ce démonstrateur a été conçu pour offrir le maximum d'interactivité aux visiteurs avec l'environnement (voir figure 50) sans le guider dans une utilisation scénarisée afin de tester l'aspect dynamique des infrastructures mises en œuvre. Notre environnement est divisé en trois espaces où les visiteurs peuvent interagir avec une partie du démonstrateur.

L'espace LiveMountain

L'application LiveMountain affiche continuellement de l'information dans la périphérie de l'attention de l'utilisateur. Il est composé d'un service de présentation de l'information et trois sources d'informations de contexte. Il affiche un tableau représentant un paysage de montagne et symbolisant les conditions

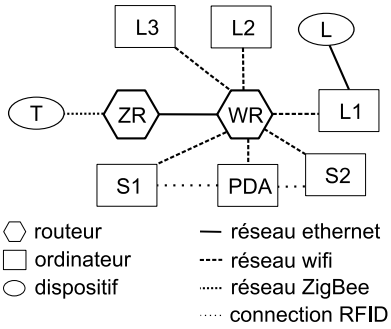


FIGURE 51 – Connectivité des élements du démonstrateurs.

de ski. Il affiche aussi le désir et la disponibilité de l'utilisateur local qu'il infère à partir d'informations du contexte de l'utilisateur qu'il interprète. Les sources de contexte sont potentiellement redondantes : un service web fournissant les conditions météo et l'enneigement, des capteurs de température et de luminosité extérieure, les agendas partagés des utilisateurs...

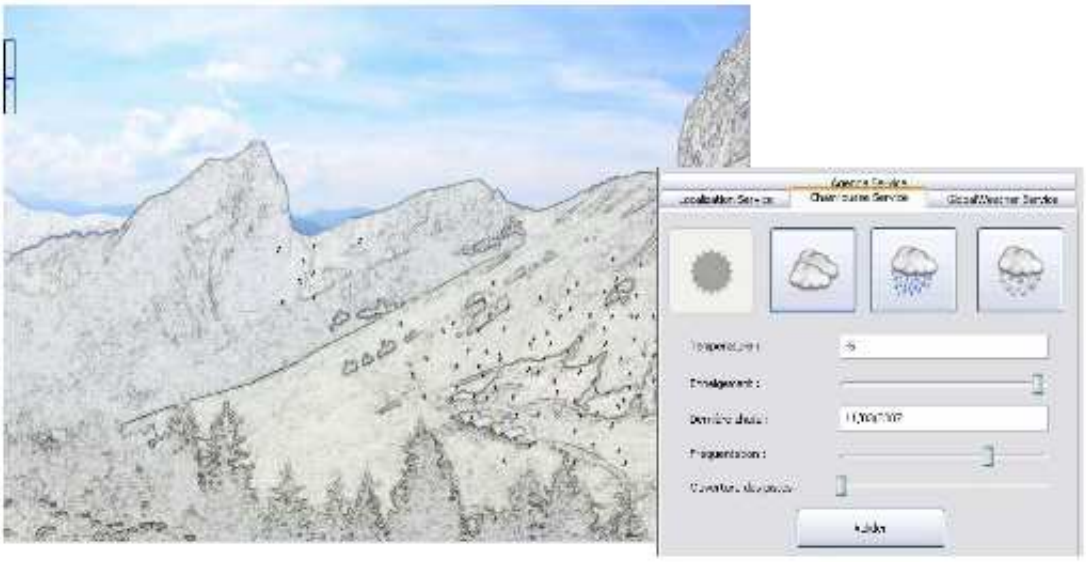


FIGURE 52 – Le tableau de LiveMountain et la console de simulation pour l'information météorologique.

Pour illustrer l'interaction de l'infrastructure de contexte et de l'application LiveMoutain, le démonstrateur est équipé d'une console de simulation pour modifier les informations des sources d'informations de contexte. Les visiteurs peuvent ainsi :

- Démarrer ou arrêter les producteurs d'informations de contexte et regarder les modifications du comportement de LiveMountain.
- Modifier les informations produites par les sources d'information de contexte et se rendre compte des modifications de la représentation que fournit LiveMoutain.

La figure 52 nous montre le tableau affiché par LiveMoutain et la console de simulation correspondant à cette situation.

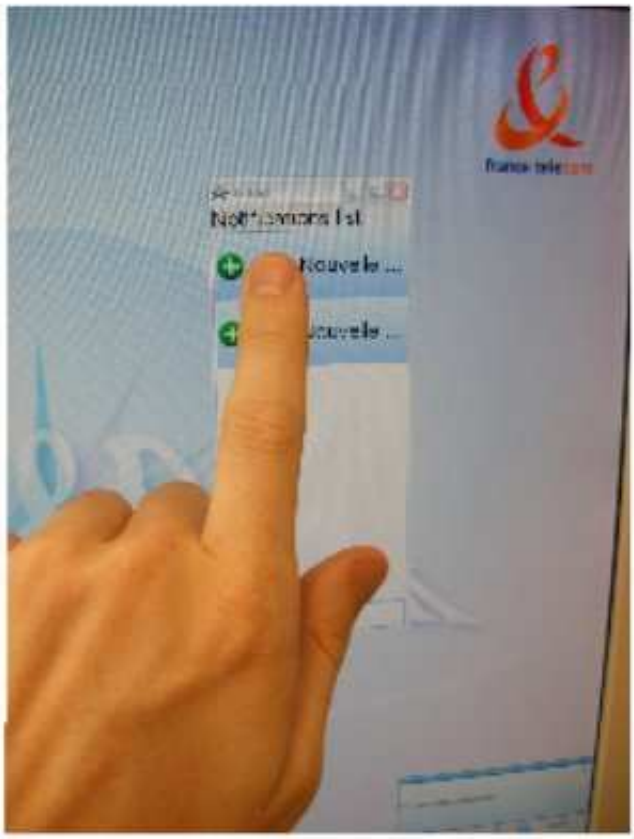
L'espace iRiderNotification

L'application iRiderNotification utilise une application pair-à-pair de covoiturage appelé iRider et un service de notification diffuse pour les propositions de covoiturage. iRider permet d'enregistrer des propositions de covoiturage. L'utilisateur en est informé par différents dispositifs de notification disséminés dans l'environnement qui sont dynamiquement choisis par l'infrastructure de composition en fonction du contexte de l'utilisateur et la pertinence des propositions par rapport aux activités et aux préférences de l'utilisateur.

Afin d'illustrer le rôle des infrastructure de contexte et de composition dans l'application iRiderNotification, les visiteurs ont accès à une version locale de l'application iRider qui leur permet de :

- Enregister leurs préférences et souhaits de covoiturage.
- Fournir une proposition de covoiturage, et observer comment l'environnement la notifie à l'utilisateur en fonction de sa pertinence.

Les visiteurs peuvent aussi agir sur le contexte de Tom, par exemple sa disponibilité, pour influencer le choix du dispositif de notification.



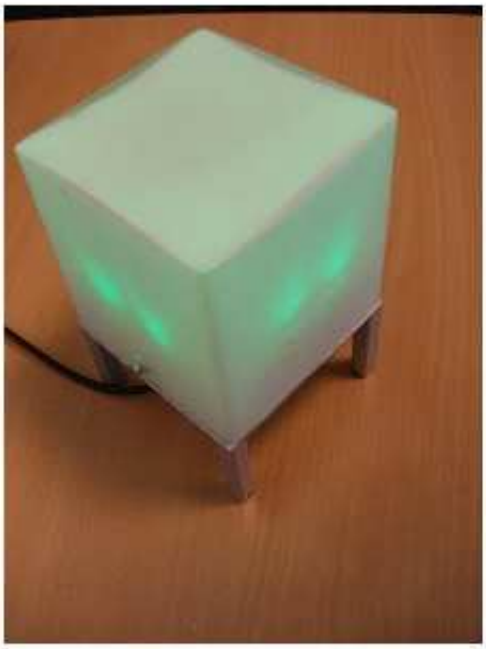


FIGURE 53 – Deux dispositifs de notification : Un pop-up de notification (à gauche) et la lampe ambiante (à droite).

FIGURE 54 – Le PDA equipé avec un lecteur de tag RFID et le tag associé à la lampe ambiante.

L'espace TangibleZoom

L'application TangibleZoom offre une manipulation diffuse de l'information. En utilisant un dispositif mobile que l'utilisateur déplace d'un dispositif de notification vers un dispositif d'affichage sur lequel il peut regarder le détail de l'information qui lui est notifiée. Cette application nécessite la détection de la proximité, que nous réalisons grâce aux tags RFID placés à proximité des dispositifs dans l'environnement et du PDA muni d'un lecteur de tag, pour assurer la composition dynamique entre dispositifs, services de notification et le dispositif mobile.

Les visiteurs expérimentent l'application TangibleZoom comme s'ils habitaient l'environnement. Ils portent le PDA et scannent le tag RFID associé avec le tableau de LiveMountain (RFID1), l'écran tactile (RFID2) ou la lampe ambiante (RFID3). Ainsi, le PDA affiche une page web relative au dispositif scanné et à l'application qui utilise ce dispositif. Par exemple, la figure 54 nous montre la page web qui apparaît sur le PDA après avoir scanné le tag associé à la lampe ambiante.

6.7 Bilan

Dans un premier temps, la réalisation de ce démonstrateur a permis d'évaluer la difficulté pour un fabricant d'utiliser l'implémentation de notre solution et de l'embarquer dans un capteur, un service ou une application. Il apparaît que pour les dispositifs limités en puissance de calcul comme le sont les capteurs de température de notre démonstrateur, il faut qu'ils aient un représentant dans une plateforme multi-agent accessible sur le réseau. Cependant ce n'est pas un obstacle insurmontable, et malgré les multitudes d'environnements d'éxécution utilisés dans notre démonstrateur (J2EE, OSGi, Amigo.NET et ZigBee), nous avons réussi facilement à implémenter un composant de gestion de contexte pour tous les consommateurs et producteurs d'informations de contexte de l'environnement puisque le développement de la totalité du démonstrateur nous a demandé seulement un peu plus d'un mois de travail (pour une équipe réduite de trois développeurs).

Dans un deuxième temps, cette expérimentation a permis de tester le fonctionnement et la robustesse de notre proposition dans un environnement réel très peu contraint. Nous n'avons pas relevé de problème majeur pendant l'utilisation du démonstrateur qui s'est révélé entièrement fonctionnel. Au démarrage du démonstrateur, la première exécution du protocole par les consommateurs d'informations de contexte s'achève par leur connexion prévue avec le ou les producteurs prévus. Les producteurs d'informations de contexte peuvent être désactivés "à chaud" et les consommateurs en découvrent de nouveaux sans aucune

intervention de notre part. Le démonstrateur a toujours assuré l'ensemble de ses fonctionnalités. Cependant, nous n'avons pas pu tester l'arrivée de nouveaux dispositifs et applications dans l'environnement. Le travail préparatoire qui a consisté à munir chaque consommateur et chaque producteur d'un modèle de contexte représentant une vision du monde différente pour chacun d'eux, permet de confirmer que l'infrastructure supporte la découverte et prend en compte les dispositifs et les applications hétérogènes.

Conclusion

Nous avons proposé une représentation et une gestion de l'information de contexte pour les environnements d'informatique diffuse. L'objectif de nos travaux est de considérer spécifiquement l'aspect dynamique de la gestion de contexte dans ce type d'environnement ouvert et changeant (des utilisateurs portant des dispositifs entrent dans l'environnement, se déplacent...). L'aspect dynamique requiert aussi d'atteindre des objectifs d'ouverture pour construire notre infrastructure, c'est-à-dire la capacité à enrichir l'environnement avec de nouveaux dispositifs, typiquement quand des nouveaux dispositifs avec des caractéristiques jusqu'alors inconnues sont disponibles. De plus, nous nous sommes efforcés de proposer une infrastructure qui ne nécessite qu'un minimum d'efforts dans son utilisation en utilisant des technologies standards et ne demandant que peu d'adaptations.

Nous avons attaqué ce problème en fournissant une infrastructure pour aider les développeurs à la construction de dispositifs et d'applications sensibles aux informations de contexte. Cette infrastructure se base sur :

Une architecture distribuée à base de composants qui permet l'ajout de nouveaux dispositifs fournissant des informations de contexte aux applications, à n'importe quel instant de l'exécution ;

Une représentation du contexte qui, en utilisant les technologies du web sémantique (RDF, OWL et SPARQL), assure l'interopérabilité entre des composants développés indépendamment grâce au caractère d'ouverture de ces technologies et leur capacité à incorporer de nouvelles descriptions ;

Un protocole minimal afin d'acquérir les informations de contexte en utilisant SPARQL, un langage de requête pour le web sémantique ;

Des correspondances entre les ontologies afin de faire face aux descriptions sémantiques hétérogènes.

Nous avons montré comment ces technologies interagissent dans le but de faire face aux problèmes d'hétérogénéité et de dynamisme. Cette proposition à été implémentée en utilisant des technologies toutes déjà disponibles. La première perspective de recherche est de tester et de montrer comment notre infrastructure remplit l'objectif de passage á l'échelle. La force principale de notre proposition réside dans le juste assemblage de ces éléments.

1 Comparaisons avec l'existant

Comme cela à été exposé dans la partie 2, il existe de nombreuses infrastructures de gestion de l'information de contexte. Dey et ses collaborateurs [31] ont précédemment travaillé sur une architecture générique pour aider les développeurs d'environnements sensibles au contexte dans leurs tâches. Le Context Toolkit qu'ils proposent est composé d'un ensemble de composants d'abstractions de l'information de contexte utilisés pour construire des applications sensibles au contexte. Ce toolkit se révèle un outil puissant et pratique pour les développeurs d'environnements sensibles au contexte puisqu'il couvre tous les aspects, de la transformation jusqu'au transport du flux d'informations de contexte des capteurs jusqu'aux effecteurs.

Malheureusement, on ne trouve pas d'explication concernant comment les informations de contexte utilisées par une application sont modélisées ,ni comment ces modèles sont utilisés dans le processus de découverte d'un composant d'abstraction du contexte lors de la construction d'un flux d'informations de contexte. Cette solution semble adaptée à la construction d'un environnement que les développeurs connaissent a priori, au contraire de notre infrastructure qui est conçue pour les environnements inconnus à leur conception et plus ouverts, c'est-à-dire, des environnements où les capteurs et actionnneurs interagissent en décrivant leurs besoins et leurs capacités.

Un système plus proche de notre travail est le travail sur les contexteurs [24]. Ils proposent une librairie d'abstractions logicielles modélisant la relation entre les variables composant une situation et celles observées par le système. Les contexteurs fournissent des informations de contexte et permettent de les combiner de façon distribuée. L'approche fournie par les contexteurs est très puissante pour construire des applications sensibles au contexte. Elle permet aux développeurs de construire une chaîne de contexteurs qui s'étend des valeurs numériques observables jusqu'aux situations et à l'identification d'un contexte pertinent. L'ajout dynamique de dispositifs sans arrêter le système ou l'infrastructure n'est pas explicité. À l'instar du Context Toolkit, cette approche manque d'ouverture et de dynamisme pour être utilisée dans un environnement inconnu et imprévisible.

L'intergiciel PACE [47] est une autre approche sous forme de toolkit. Elle offre la gestion du contexte et des préférences à travers l'utilisation d'une boîte à outils. Cette infrastructure prête à l'emploi pour construire des applications sensibles au contexte ne permet pas de lier dynamiquement de nouveaux modèles de contexte à ceux déjà utilisés. Ainsi, les applications existantes dans l'environnement ne peuvent pas tirer parti des nouvelles applications qui entrent dans l'environnement, à moins que ces dernières n'utilisent des ontologies pré-existantes dans l'environnement pour décrire les informations de contexte qu'elles utilisent. Par contre, des ontologies développées indépendamment peuvent être utilisées.

Ces trois boîtes à outils imposent l'utilisation de leur intergiciel complet pour implémenter des applications sensibles au contexte et construire des environnements intelligents. Dans notre infrastructure, nous avons suivi une autre approche en adoptant le plus possible des standards disponibles. Par conséquent, l'approche que nous proposons peut être adoptée par différents outils de manière interopérable.

Il y a plusieurs propositions pour étendre les languages du web sémantique afin de contextualiser les assertions [12, 45, 53]. Les dispositifs et les applications tels que nous les avons décrits dans nos travaux peuvent utiliser l'une ou l'autre de ces propositions. L'utilisation de C-OWL [12] pour représenter les informations de contexte se rapproche fortement de notre proposition dans l'utilisation de correspondances entre ontologies. Alors que C-OWL permet de gérer statiquement plusieurs contextes en utilisant OWL, notre proposition se concentre autour de l'aspect dynamique.

En ce qui concerne l'utilisation de OWL pour représenter les informations contextuelles, [82] introduit une ontologie de haut niveau pour les informations de contexte dans les environnements d'intelligence ambiante, même si les auteurs ne font pas de proposition pour l'étendre dynamiquement. Son interêt est le même que ce qui est présenté dans la section 4.2 : pouvoir étendre des ontologies. Une approche similaire est proposée dans le projet CoBra [20]. Cette infrastructure peut accueillir des dispositifs qui entrent dans l'environnement grâce à son intergiciel "context broker", mais ne permet en aucun cas d'étendre son ontologie pour incorporer de nouvelles informations de contexte.

Nous finirons par un dernier exemple d'utilisation d'ontologie dans les environnements d'intelligence ambiante avec le projet Gaia [67]. L'infrastructure est un système multi-agents où les agents sont des sources de contexte, des producteurs de contexte ou des synthétiseurs de contexte. Un serveur d'ontologies enregistre les ontologies utilisées pour établir des liens terminologiques entre les modèles de contexte des agents. Même si des agents peuvent enrichir ces ontologies avec de nouveaux concepts, il n'y a pas de proposition pour assurer l'interopérabilité entre ces concepts nouvellement introduits.

2 Discussion

Le travail que nous avons présenté ici porte une attention particulière aux propriétés suivantes des infrastructures de gestion d'information de contexte qui nous semblent indispensables pour la construction d'applications d'informatique diffuse :

L'ouverture est la première exigence aussi appelée *soutien pour l'hétérogénéité* dans [47]. L'utilisation des technologies du web sémantique dans notre infrastructure lui permet d'accepter l'introduction de dispositifs, de services et d'applications inconnus et imprévus pendant l'exécution, sans arrêt ou reconfiguration. L'utilisation de technologies standards encourage l'intégration de nos travaux dans les applications, d'autant plus que ces technologies sont conçues comme des technologies ouvertes. De plus, l'utilisation de standards fournit d'autant plus d'aide aux développeurs qu'ils peuvent profiter des outils développés pour ces technologies, par exemple des systèmes de raisonnement.

Le dynamisme , ou flexibilité, recouvre la faculté à prendre en compte l'hétérogénéité des dispositifs, des applications et aussi bien des représentations de contexte pendant l'exécution. Ceci est une nouvelle fois rendu possible à l'aide des technologies du web sémantique, notamment grâce à la possibilité de toujours étendre une représentation par une autre représentation, et grâce à la technologie d'alignement d'ontologie pour rapprocher des représentations.

Les obligations minimales qui se rapproche fortement de la notion de *facilité de deploiement et de configuration* de [47], est assurée par l'utilisation de technologies standards et l'utilisation d'un protocole d'interaction simple qui ne nécessite pas d'opérations coûteuses en ressources de calcul. Le but est de rendre notre infrastructure la plus facilement intégrable à d'autres technologies en assurant les fonctionnalités d'une infrastructure de gestion d'information de contexte. Les technologies utilisées sont déjà recommandées par le W3C et sont largement utilisées ; ce qui ne devrait pas obliger les industriels à fournir un effort insurmontable pour les adopter. Ceci rejoint également l'utilisation de *standards* exigé par [18].

Les exigences concernant la représentation du contexte introduites dans [18] appelle à une representation *structurée*, *interchangeable*, *uniforme*, *extensible* et *standardisée*. Ces exigences sont toutes satisfaites par l'utilisation des technologies du web sémantique, notamment RDF ou OWL. La propriété supplémentaire, *décomposable*, n'est pas étudiée ici.

[47] ou [23] présentent des exigences pour les infrastructures de gestion de contexte que nous n'avons pas directement considérées dans nos travaux, mais en s'appuyant sur des technologies standardisées, notre solution peut répondre à ces exigences :

Mobilité [47] Nous n'avons pas fourni de disposition spéciale pour supporter la mobilité des dispositifs. Cependant, notre bibliothèque propose un protocole fonctionnel pour tous les dispositifs atteignables sur le réseau. Notre implémentation en utilisant une plateforme d'exécution multi-agents supporte la mobilité des dispositifs, l'adressage et le transport de messages d'une plateforme à une autre plateforme d'exécution comme nous l'avons utilisé dans notre démonstrateur (6). De plus, des travaux existent pour interfacer la plateforme JADE avec d'autres systèmes multi-agent ou le monde de l'internet.

Scalabilité [47] Nous n'avons pas assez testé notre infrastructure pour prouver son passage à l'échelle. Cependant en utilisant des technologies simples et légères, nous pensons qu'elle supportera le déploiement d'applications dans un environnement d'informatique diffuse domestique. La disponibilité des serveurs d'alignement peut apparaître comme une menace à ce déploiement. La réalisation du démonstrateur a déjà prouvé que l'on peut déployer un nombre non négligeable d'applications et de dispositifs en utilisant notre infrastructure.

Respect de la vie privé [47, 23] et *Sécurité* [23]. À la création des gestionnaires d'informations de contexte de leurs dispositifs ou de leurs applications, les développeurs choisissent les informations de contexte qu'ils veulent voir exhibées sur le réseau et transmises aux autres applications et dispositifs ce qui assure le respect de la vie privée. Par contre, aucune méthode n'a été mise en place pour assurer une sécurité élaborée.

Traçabilité et contrôle [47], ou *Historique* [23], L'historique des informations de contexte peut se révéler très utile pour les environnements d'informatique diffuse, notamment pour inférer le futur à partir du passé, et adapter son comportement aux habitudes et aux préférences des utilisateurs. Nous ne gérons pas l'historisation des informations de contexte, mais les développeurs peuvent associer à notre bibliothèque des librairies pour stocker des descriptions OWL et RDF dans des bases de données relationnelles. Le flux d'information circulant entre les producteurs d'information de contexte et les consommateurs est cachée des utilisateurs. Par contre, les développeurs qui utilisent notre implémentation JADE utiliseront le "sniffer" fourni par la plateforme d'exécution pour effectuer le debuggage (voir figure 30).

Tolérance aux pannes [47] L'ouverture et le caractère distribué de notre infrastructure est une bonne base pour obtenir un système tolérant aux pannes. En effet, si un producteur d'informations de contexte s'arrête, le consommateur d'information peut lui trouver simplement un remplaçant en exécutant le protocole. Le seul composant centralisé de l'infrastructure est le serveur d'alignement. La redondance de ce composant permettrait de surmonter cette faiblesse, et cela est possible.

Il y a tout de même des limitations à notre approche. Ainsi, comme la plupart des solutions utilisant des ontologies, elle est plus flexible mais certainement plus lente. Par exemple, une application basée sur le web sémantique et utilisant notre infrastructure nécessitera des opérations de raisonnement, de recherche de correspondances entre les ontologies et de calcul de réponses à des requête, ce qui peut se révéler être des opérations très complexes. Heureusement, la compléxité d'une telle solution ne constitue pas un obstacle insurmontable d'autant qu'il existe des solutions spécifiques à ces problèmes. Par exemple, [9] propose une technique de compilation des ontologies afin d'accélerer le processus de réponse aux requêtes, ce qui est appelé "matching in context". L'utilisation de méthodes de raisonnements approximatifs est une autre solution qui peut aussi être utilisée dans les environnements d'informatique diffuse : c'est-à-dire proposer un comportement pour aider l'utilisateur dans sa tâche, mais pas forcément le comportement optimal.

Bibliographie

[1] F. Abel and J. Brase. Using semantic web technologies for context-aware information providing to mobile devices. Technical report, University of Hannover, 2004. http ://www.kbs.uni-hannover.de/Arbeiten/Publikationen/2004/abel$_b rase_K BS_h annover.pdf$.

[2] G. D. Abowd, C. G. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton. Cyberguide : a mobile context-aware tour guide. *ACM Wireless Networks*, 3 :421–433, 1997.

[3] R. Aipperspach, B. Hooker, and A. Woodruff. The heterogeneous home. In *UbiComp '08 : Proceedings of the 10th international conference on Ubiquitous computing*, pages 222–231, New York, NY, USA, 2008. ACM.

[4] Z. Aleksovski, Kateten Warner, and F. van Harmelen. Exploiting the structure of background knowledge used in ontology matching. In I. F. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, and L. Aroyo, editors, *The Semantic Web - ISWC 2006, 5th International Semantic Web Conference, ISWC 2006, Athens, GA, USA, November 5-9, 2006, Proceedings*, volume 4273 of *Lecture Notes in Computer Science*. Springer, Nov. 2006.

[5] O. Alliance. *OSGi Service Platform : The OSGi Alliance*. IOS Press,US, 2007.

[6] I. Amundson, M. Kushwaha, X. Koutsoukos, S. Neema, and J. Sztipanovits. Oasis : A service-oriented middleware for pervasive ambient-aware sensor networks. *Pervasive and Mobile Computing Journal on Middleware for Pervasive Computing*, October 2006.

[7] M. Anne, J. L. Crowley, V. Devin, and G. Privat. Localisation intra-bâtiment multi-technologies : Rfid, wifi et vision. In *UbiMob '05 : Proceedings of the 2nd French-speaking conference on Mobility and uibquity computing*, pages 29–35, New York, NY, USA, 2005. ACM.

[8] M. Beigl. Memoclip : A location-based remembrance appliance. *Personal Ubiquitous Computing*, 4(4) :230–233, 2000.

[9] S. Ben Mokhtar, A. Kaul, N. Georgantas, and V. Issarny. Efficient semantic service discovery in pervasive computing environments. In *Proc. 7th Middleware conference*, volume 4290 of *Lecture notes in computer science*, pages 240–259, Melbourne (AU), 2006.

[10] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5) :34–43, 2001.

[11] G. Bieber, L. Architect, and I. Ci. Introduction to service-oriented programming, 2001. http ://www.openwings.org.

[12] P. Bouquet, F. Giunchiglia, F. V. Harmelen, L. Serafini, and H. Stuckenschmidt. C-OWL : Contextualizing ontologies. In *Proc. 2nd International Semantic Web Conference*, pages 164–179, 2003.

[13] O. Brdiczka, J. Maisonnasse, P. Reignier, and J. L. Crowley. Detecting small group activities from multimodal observations. *International Journal of Applied Intelligence*, 2007. IN PRESS.

[14] D. Brickley and L. Miller. Foaf vocabulary specification, 2005. http ://xmlns.com/foaf/0.1/.

[15] J. Brotherton, K. Truong, and G. Abowd. Supporting capture and access interfaces for informal and opportunistic meetings, 1999. http ://www.cc.gatech.edu/gvu/reports/1999/.

[16] P. J. Brown. The stick-e document : a framework for creating context-aware applications. In *Proceedings of EP'96, Palo Alto*, pages 259–272. also published in it EP–odd, January 1996.

[17] P. J. Brown, J. D. Bovey, and X. Chen. Context-aware applications : from the laboratory to the marketplace. *IEEE Personal Communications*, 4(5) :58–64, October 1997.

[18] S. Buchholz, T. Hamann, and G. Hübsch. Comprehensive structured context profiles (CSCP) : Design and experiences. In *Proc. 2nd IEEE Conference on Pervasive Computing and Communications Workshops (PERCOMW)*, pages 43–47, Washington (DC US), 2004.

[19] M. Chalmers. A historical view of context. *Computer supported cooperative work*, 13(3) :223–247, 2004.

[20] H. Chen, T. Finin, and A. Joshi. An ontology for context-aware pervasive computing environments. *Knowledge engineering review*, 18(3) :197–207, 2004.

[21] H. Chen, T. Finin, and A. Joshi. *The SOUPA Ontology for Pervasive Computing*. Whitestein Series in Software Agent Technologies. Springer, July 2005.

[22] A. K. Clear, S. Knox, J. Ye, L. Coyle, S. Dobson, and P. Nixon. Integrating multiple contexts and ontologies in a pervasive computing framework. volume 210, pages 20–25, Riva Del Garda, Italy, 28/08/2006 2006. CEUR Workshop Proceedings, CEUR Workshop Proceedings.

[23] J. Coutaz, J. Crowley, S. Dobson, and D. Garlan. Context is key. *Communications of the ACM*, 48(3) :49–53, 2005.

[24] J. Coutaz and G. Rey. Foundations for a theory of contextors. In C. Kolski and J. Vanderdonckt, editors, *CADUI*, pages 13–34. Kluwer, 2002.

[25] J. Crowley, J. Coutaz, G. Rey, and P. Reignier. Perceptual components for context aware computing. In *UbiComp '02 : Proceedings of the 4th international conference on Ubiquitous Computing*, pages 117–134, London, UK, 2002. Springer-Verlag.

[26] I. F. Cruz, S. Decker, J. Euzenat, and D. L. McGuinness, editors. *The Emerging Semantic Web, Selected papers from the first Semantic web working symposium, Stanford University, California, USA, July 30 - August 1, 2001*, volume 75 of *Frontiers in Artificial Intelligence and Applications*. IOS press, 2002.

[27] J. de Kleer. An assumption-based TMS. *Artificial Intelligence*, 28(2) :127–162, 1986.

[28] M. Dean and G. Schreiber. OWL Web Ontology Language Reference. Recommendation, W3C, February 10 2004. http ://www.w3.org/TR/owl-ref/.

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

[30] A. Dey. Understanding and using context. *Personal Ubiquitous Computing*, 5(1) :4–7, 2001.

[31] A. Dey, D. Salber, and G. Abowd. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction*, 16 :97–166, 2001.

[32] P. Dourish. Seeking a foundation for context-aware computing. *Human-Computer Interaction*, 16 :2–3, 2001.

[33] P. Dourish. What we talk about when we talk about context. *Personal Ubiquitous Comput.*, 8(1) :19–30, 2004.

[34] J. Euzenat. Research challenges and perspectives of the semantic web. *IEEE Intelligent Systems*, 17(5) :86–88, 2002.

[35] J. Euzenat. An API for ontology alignment. In S. McIlraith, D. Plexousakis, and F. van Harmelen, editors, *Proc. 3rd International Semantic Web Conference (ISWC)*, volume 3298 of *Lecture Notes in Computer Science*, pages 698–712, Hiroshima (JP), 2004.

[36] J. Euzenat. Alignment infrastructure for ontology mediation and other applications. In M. Hepp, A. Polleres, F. van Harmelen, and M. Genesereth, editors, *Proc. 1st ICSOC international workshop on Mediation in semantic web services, Amsterdam (NL)*, pages 81–95, 2005.

[37] J. Euzenat and P. Shvaiko. *Ontology matching*. Springer, Heidelberg (DE), 2007.

[38] D. Fensel, J. A. Hendler, H. Lieberman, and W. Wahlster, editors. *Spinning the Semantic Web : Bringing the World Wide Web to Its Full Potential [outcome of a Dagstuhl seminar]*. MIT Press, 2003.

[39] G. Fischer. Articulating the task at hand and making information relevant to it. *Hum.-Comput. Interact.*, 16(2) :243–256, 2001.

[40] T. Flury, G. Privat, and F. Ramparany. OWL-based location ontology for context-aware services. In *Proceedings Artificial Intelligence in Mobile Systems*, pages 52–58, 2004.

[41] J. Fogarty, S. E. Hudson, C. G. Atkeson, D. Avrahami, J. Forlizzi, S. Kiesler, J. C. Lee, and J. Yang. Predicting human interruptibility with sensors. *ACM Trans. Comput.-Hum. Interact.*, 12(1) :119–146, 2005.

[42] S. Gibet, A. Braffort, C. Collet, F. Forest, R. Gherbi, and T. Lebourque. Gesture in Human-Machine Communication : capture, analysis-synthesis, recognition, semantics . In *Gesture Workshop'96 , York, Royaume-Uni, 19/03/1996*, pages 89–95, http ://www.springerlink.com/, mars 1996. Springer-Verlag OK.

[43] T. Gu, X. H. Wang, H. K. Pung, and D. Q. Zhang. An ontology-based context model in intelligent environments. In *In Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference*, pages 270–275, 2004.

[44] R. Guha. *Contexts : A Formalization and Some Applications*. PhD thesis, Stanford university, Stanford (CA US), 1995.

[45] R. Guha, R. McCool, and R. Fikes. Contexts for the semantic web. In *International Semantic Web Conference*, pages 32–46, 2004.

[46] D. Heckmann, E. Schwarzkopf, J. Mori, D. Dengler, and A. Krner. The user model and context ontology gumo revisited for future web 2.0 extensions. In *Proc. 3rd Contexts and ontologies workshop*, pages 37–46, Roskilde (DK), 2007.

[47] K. Henricksen, J. Indulska, T. McFadden, and S. Balasubramaniam. Middleware for distributed context-aware systems. In *Proc. International Symposium on Distributed Objects and Applications (DOA)*, volume 3760 of *Lecture Notes in Computer Science*, pages 846–863, Orlando (FL US), 2005.

[48] J. R. Hobbs and F. Pan. Time ontology in owl. World Wide Web Consortium, Working Draft WD-owl-time-20060927, September 2006. http ://www.w3.org/TR/2006/WD-owl-time-20060927.

[49] I. Horrocks and J. A. Hendler, editors. *The Semantic Web - ISWC 2002, First International Semantic Web Conference, Sardinia, Italy, June 9-12, 2002, Proceedings*, volume 2342 of *Lecture Notes in Computer Science*. Springer, 2002.

[50] H. Ishii and B. Ullmer. Tangible bits : towards seamless interfaces between people, bits and atoms. In *CHI '97 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 234–241, New York, NY, USA, 1997. ACM Press.

[51] L. Kagal, T. W. Finin, and A. Joshi. A policy based approach to security for the semantic web. In *International Semantic Web Conference*, pages 402–418, 2003.

[52] M. Kallmann and D. Thalmann. Direct 3d interaction with smart objects. In *VRST '99 : Proceedings of the ACM symposium on Virtual reality software and technology*, pages 124–130, New York, NY, USA, 1999. ACM.

[53] O. Khriyenko and V. Terziyan. Context description framework for the semantic web. In *Proceedings Context 2005 Context representation and reasoning workshop, Paris (FR)*, 2005.

[54] G. Klyne and J. Carroll. Resource description framework (RDF) : Concepts and abstract syntax. Recommendation, W3C, 2004.

[55] L. Leahu, P. Sengers, and M. Mateas. Interactionist ai and the promise of ubicomp, or, how to put your box in the world without putting the world in your box. In *UbiComp '08 : Proceedings of the 10th international conference on Ubiquitous computing*, pages 134–143, New York, NY, USA, 2008. ACM.

[56] D. B. Lenat and R. V. Guha. *Building Large Knowledge-Based Systems ; Representation and Inference in the Cyc Project*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[57] K. Lyytinen and Y. Yoo. Introduction : Issues and challenges in ubiquitous computing. *CACM*, 45(12) :62–65, Dec. 2002.

[58] J. McCarthy. Notes on formalizing contexts. In T. Kehler and S. Rosenschein, editors, *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 555–560, Los Altos, California, 1986. Morgan Kaufmann.

[59] J. Pascoe. Adding generic contextual capabilities to wearable computers. In *The Second International Symposium on Wearable Computers*, pages 92–99, Pittsburgh, October 1998. IEEE Computer Society. Online proceedings available from http ://iswc.gatech.edu/.

[60] J. Pascoe, N. Ryan, and D. Morse. Using while moving : Hci issues in fieldwork environments. *ACM Trans. Comput.-Hum. Interact.*, 7(3) :417–437, September 2000.

[61] F. Perich. Mogatu bdi ontology, 2004. http ://mogatu.umbc.edu/bdi/.

[62] S. Peters and H. E. Shrobe. Using semantic networks for knowledge representation in an intelligent environment. In *PERCOM '03 : Proceedings of the First IEEE International Conference on Pervasive Computing and Communications*, page 323, Washington, DC, USA, 2003. IEEE Computer Society.

[63] S. Powers. *Practical RDF*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2003.

[64] E. Prud'hommeaux and A. Seaborne. SPARQL query language for RDF. Working draft, W3C, 2006. http ://www.w3.org/TR/rdf-sparql-query/.

[65] F. Ramparany, J. Euzenat, T. H. F. Broens, A. Bottaro, and R. Poortinga. Context management and semantic modelling for ambient intelligence. Technical Report TR-CTIT-06-52, Enschede, April 2006.

[66] D. A. Randell, Z. Cui, and A. Cohn. A spatial logic based on regions and connection. In B. Nebel, C. Rich, and W. Swartout, editors, *KR'92. Principles of Knowledge Representation and Reasoning : Proceedings of the Third International Conference*, pages 165–176, San Mateo, California, 1992. Morgan Kaufmann.

[67] A. Ranganathan and R. Campbell. A middleware for context-aware agents in ubiquitous computing environments. In *Proc. 4th Middleware conference*, volume 2672 of *Lecture notes in computer science*, pages 143–161, Rio de Janeiro (BR), 2003.

[68] A. Ranganathan, R. E. McGrath, R. H. Campbell, and M. D. Mickunas. Use of ontologies in a pervasive computing environment. *Knowledge engineering review*, 18(3) :209–220, 2003.

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

[70] N. S. Ryan, J. Pascoe, and D. R. Morse. Enhanced reality fieldwork : the context-aware archaeological assistant. In V. Gaffney, M. van Leusen, and S. Exxon, editors, *Computer Applications in Archaeology 1997*, British Archaeological Reports, Oxford, October 1998. Tempus Reparatum.

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

[72] B. N. Schilit, N. Adams, R. Gold, M. Tso, and R. Want. The PARCTAB mobile computing system. In *Proceedings Fourth Workshop on Workstation Operating Systems (WWOS-IV)*, pages 34–39. IEEE, October 1993.

[73] A. Schmidt. Implicit human computer interaction through context. *Personal Technologies*, 4(2-3) :191–199, June 2000.

[74] A. Seaborne. RDQL – a query language for RDF. Submission, W3C, 2004.

[75] L. Serafini and P. Bouquet. Comparing formal theories of context in ai. *Artif. Intell.*, 155(1-2) :41–67, 2004.

[76] T. Strang and C. Linnhoff-Popien. A context modeling survey. In *Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/England*, 2004.

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

[78] H. Stuckenschmidt, F. van Harmelen, P. Bouquet, F. Giunchiglia, and L. Serafini. Using C-OWL for the alignment and merging of medical ontologies. In *KR-MED*, pages 88–101, 2004.

[79] M. Vallée, F. Ramparany, and L. Vercouter. Flexible composition of smart device services. In *Pervasive 2006*, Dublin, Ireland, May 2006.

[80] A. van Dam. Post-WIMP user interfaces. *Communications of the ACM*, 40(2) :63–67, 1997.

[81] A. E. Walsh, editor. *Uddi, Soap, and Wsdl : The Web Services Specification Reference Book*. Prentice Hall Professional Technical Reference, 2002.

[82] X. H. Wang, D. Q. Zhang, T. Gu, and H. K. Pung. Ontology based context modeling and reasoning using OWL. In *PERCOMW '04 : Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*, page 18, Washington, DC, USA, 2004. IEEE Computer Society.

[83] A. Ward, A. Jones, and A. Hopper. A new location technique for the active office. *Personal Communications, IEEE [see also IEEE Wireless Communications]*, 4(5) :42–47, 1997.

[84] M. Weiser. The computer for the twenty-first century. *Scientific American*, 256(3) :94 – 104, Sept. 1991.

[85] T. Winograd. Architectures for context. *Human-Computer Interaction*, 16(2–4) :401–419, Dec. 2001.

[86] W. Xue, H. Pung, P. P. Palmes, and T. Gu. Schema matching for context-aware computing. In *UbiComp '08 : Proceedings of the 10th international conference on Ubiquitous computing*, pages 292–301, New York, NY, USA, 2008. ACM.

[87] S. Zaidenberg. *Apprentissage par renforcement de modèles de contexte pour l'informatique ambiante*. PhD thesis, Groupe Grenoble INP, 46, avenue Félix Viallet - 38031 Grenoble Cedex 1 - France, oct 2009.

Abstract

In a pervasive computing environment, the environment itself is the interface between services and users. Using context information coming from sensors, location technologies and agregation services, applications adapt their run-time behaviour to the context in which users evolve (e.g., physical location, social or hierarchical position, current tasks as well as related information). These applications have to deal with the dynamic integration in the environment of new elements (users or devices), and the environment has to provide context information to newly designed applications. We study and develop a dynamic context management system for pervasive application. It is flexible enough to be used by heterogeneous applications and to run dynamically with new incoming devices.

We have designed an architecture in which context information is distributed in the environment. Each device or service implements a context manager component in charge of maintaining its local context. It can communicate with other context manager components : some of them are context information producers, some of them are context information consumers and some of them are both. We have defined a simple protocol to allow a consumer to identify and determine the producer for the information it needs. Context manager components express their context information using an OWL ontology, and exchange RDF triples with each other. The openness of ontology description languages makes possible the extension of context descriptions and ontology matching helps dealing with independently developed ontologies. Thus, this architecture allows the introduction of new components and new applications without interrupting what is working.

Résumé

Les environnements d'intelligence ambiante servent d'interface entre les applications et les utilisateurs. Ces applications doivent prendre en compte le contexte dans lequel les utilisateurs évoluent (le lieu, la position sociale ou hiérarchique ou l'activité par exemple) pour adapter leur comportement. Il doit exister un flux d'informations de l'environnement vers les applications. Ces applications doivent pouvoir prendre en compte dynamiquement l'arrivée de nouveaux éléments dans l'environnement (utilisateurs ou dispositifs), et les informations de contexte en provenance de l'environnement doivent pouvoir parvenir aux applications entrantes ; ces flux d'informations ne peuvent pas être déterminés à l'avance et doivent se construire pendant l'exécution. Les modèles de gestion de l'information de contexte existants ne traitent pas ou peu cet aspect dynamique de l'informatique diffuse.

Nous proposons d'utiliser les technologies du web sémantique pour décrire et rechercher ces informations : l'information de contexte est exprimée en RDF et décrite par des ontologies OWL. Ces technologies, parce qu'elles sont fondées sur l'hypothèse du monde ouvert, assurent l'ouverture du système et la prise en compte de dispositifs hétérogènes. Nous montrons qu'à l'aide d'un protocole simple qui permet à chacun des dispositifs et des applications d'exhiber sur le réseau un modèle des informations de contexte qu'il produit ou qu'il recherche et de s'identifier, toutes les applications de l'environnement satisfont leurs besoins en informations de contexte. De surcroît, l'ouverture des langages de description d'ontologies permet l'extension des descriptions de contexte à tout moment et les technologies d'alignement d'ontologies permettent l'utilisation d'ontologies développées indépendamment.

Nous avons implémenté un composant pour la gestion de l'information de contexte fondé sur ce modèle. Puis nous avons développé une architecture distribuée où les dispositifs et les applications embarquent ce composant et exposent un modèle de l'information de contexte qu'ils recherchent ou produisent. Nous avons montré comment cette architecture permet d'accepter sans interruption de nouveaux composants.